Code No. 9246

FACULTY OF INFORMATICS

B.E. 4/4 (IT) I – Semester (Main)

Examination, Dec. 2014 / Jan. 2015

Subject: Distributed Systems (Elective – II) Time: 3 hours Max. Marks: 75

Note: Answer all questions from Part - A. Answer any FIVE questions from Part - B. PART – A (25 Marks)

1 What is open distributed system? 2

An **Open Distributed System** is made up of components that may be obtained from a number of different sources, which together work as a single **distributed system**. In 1988 the International Standards Organization (ISO) began work on preparing standards for **Open Distributed** Processing (ODP).

2 What is transparency in distributed system? 3

**Transparency**

Goal - hide the fact that its processes and resources are physically distributed across multiple computers systems should be transparent

**Different forms of transparency in a distributed system (ISO, 1995).**

|  |  |
| --- | --- |
| Transparency | Description |
| Access | Hide differences in data representation and how a resource is accessed |
| Location | Hide where a resource is located |
| Migration | Hide that a resource may move to another location |
| Relocation | Hide that a resource may be moved to another location while in use |
| Replication | Hide that a resource is replicated |
| Concurrency | Hide that a resource may be shared by several competitive users |
| Failure | Hide the failure and recovery of a resource |

**Degree of Transparency Issues**

**Timing**

e.g. requesting an electronic newspaper to appear in your mailbox before 7 A.M. local time, as usual, while you are currently at the other end of the world living in a different time zone.

**Synchronization**

e.g. a wide-area distributed system that connects a process in San Francisco to a process in Amsterdam limited by laws of physics - a message sent from one process to the other takes about 35 milliseconds.

· It takes several hundreds of milliseconds using a computer network.

· Signal transmission is not only limited by the speed of light, but also by limited processing capacities of the intermediate switches.

**Performance**

e.g. many Internet applications repeatedly try to contact a server before finally giving up. Consequently, attempting to mask a transient server failure before trying another one may slow down the system as a whole.

**Consistency**

e.g. need to guarantee that several replicas, located on different continents, need to be consistent all the time -  a single update operation may now even take seconds to complete, something that cannot be hidden from users.

**Context Awareness**

e.g. notion of location and context awareness is becoming increasingly important, it may be best to actually expose distribution rather than trying to hide it. -  consider an office worker who wants to print a file from her notebook computer. It is better to send the print job to a busy nearby printer, rather than to an idle one at corporate headquarters in a different country.

**Limits of Possibility**

Recognizing that full distribution transparency is simply impossible, we should ask ourselves whether it is even wise to pretend that we

3 What is distributed objects in RMI? 3

In [distributed computing](https://en.wikipedia.org/wiki/Distributed_computing), **distributed objects**[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)] are objects (in the sense of [object-oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming)) that are distributed across different [address spaces](https://en.wikipedia.org/wiki/Address_space), either in different [processes](https://en.wikipedia.org/wiki/Process_(computing)) on the same computer, or even in multiple [computers](https://en.wikipedia.org/wiki/Computer) connected via a [network](https://en.wikipedia.org/wiki/Computer_network), but which work together by sharing data and invoking methods. This often involves [location transparency](https://en.wikipedia.org/wiki/Location_transparency), where remote objects appear the same as local objects. The main method of [distributed object communication](https://en.wikipedia.org/wiki/Distributed_object_communication) is with [remote method invocation](https://en.wikipedia.org/wiki/Remote_method_invocation), generally by message-passing: one object sends a message to another object in a remote machine or process to perform some task. The results are sent back to the calling object.

Distributed objects were popular in the late 1990s and early 2000s, but have since fallen out of favor.[[1]](https://en.wikipedia.org/wiki/Distributed_object#cite_note-1)

The term may also generally refer to one of the extensions of the basic [object](https://en.wikipedia.org/wiki/Object_(computer_science)) concept used in the context of distributed computing, such as *replicated objects* or *live distributed objects*.

* [*Replicated objects*](https://en.wikipedia.org/wiki/Replication_(computer_science)) are groups of software components (*replicas*) that run a distributed multi-party protocol to achieve a high degree of consistency between their internal states, and that respond to requests in a coordinated manner. Referring to the group of replicas jointly as an *object* reflects the fact that interacting with any of them exposes the same externally visible state and behavior.
* [*Live distributed objects*](https://en.wikipedia.org/wiki/Live_distributed_object) (or simply [*live objects*](https://en.wikipedia.org/wiki/Live_distributed_object))[[2]](https://en.wikipedia.org/wiki/Distributed_object#cite_note-2) generalize the *replicated object* concept to groups of replicas that might internally use any distributed protocol, perhaps resulting in only a weak consistency between their local states. Live distributed objects can also be defined as running instances of distributed multi-party protocols, viewed from the object-oriented perspective as entities that have distinct identity, and that can encapsulate distributed state and behavior.

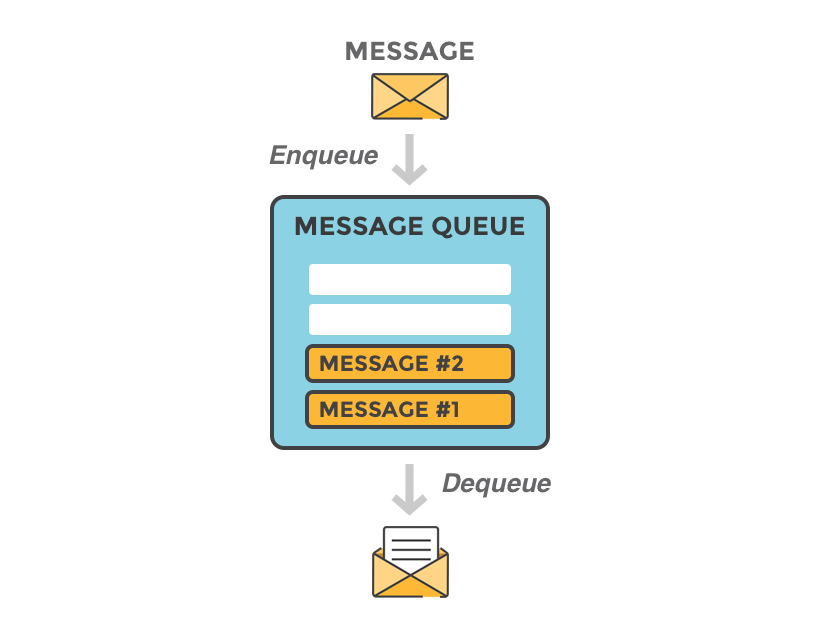
4 Explain the role of message broker in message queuing system. 2

**Message queuing allow applications to communicate by sending messages to each other. The message queue provide a temporary message storage when the destination program is busy or not connected.**

In this blog article I will explain message queueing. I will explain what it is, how it can be used and benefits achieved when using message queues.

A **queue** is a line of things waiting to be handled - in sequential order starting at the beginning of the line. A message queue is a queue of messages sent between applications. It includes a sequence of work objects that are waiting to be processed.

A **message** is the data transported between the sender and and the receiver application, it's essentially a byte array with some headers on top. A message example could be anything that tells one system to start processing a task, information about a finished task or a plain message.



The basic architecture of a **message queue** is simple, there are client applications called producers that create messages and deliver them to the message queue. An other application, called consumer, connect to the queue and get the messages to be processed. Messages placed onto the queue are stored until the consumer retrieves them.

**Message queues**

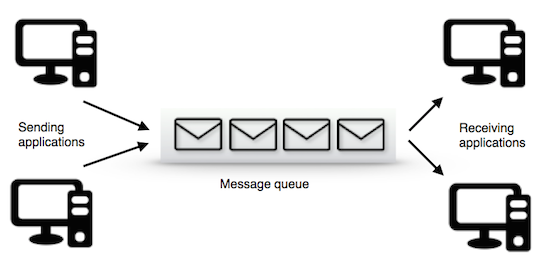
A message queue provide an **asynchronous communications protocol,** a system that puts a message onto a message queue does not require an immediate response to continue processing. Email is probably the best example of asynchronous messaging. When an email is sent can the sender continue processing other things without an immediate response from the receiver. This way of handling messages **decouple** the producer from the consumer. The producer and the consumer of the message do not need to interact with the message queue at the same time.

**DECOUPLING AND SCALABILITY**

Decoupling is used to describe how much one piece of a system relies on another piece of the system. Decoupling is the process of separating them so that their functionality will be more self contained.

A decoupled system are achieved when two or more systems are able to communicate without being connected. The systems can remain completely autonomous and unaware of each other. Decoupling is often a sign of a computer system that is well structured. It is usually easier to maintain, extend and debug.

If one process in a decoupled system fails processing messages from the queue, other messages can still be added to the queue and be processed when the system has recovered. You can also use a message queue to delay processing; A producer post messages to a queue. At the appointed time, the receivers are started up and process the messages in the queue. Messages in queue can be stored-and-forwarded and the message be redelivered until the message is processed.



Instead of building one large application, is it beneficial to decouple different parts of your application and only communicate between them asynchronously with messages. That way different parts of your application can evolve independently, be written in different languages and/or maintained by separated developer teams.

A message queue will keep the processes in your application separated and independent of each other. The first process will never need to invoke another process, or post notifications to another process, or follow the process flow of the other processes. It can just put the message on the queue and then continue processing. The other processes can also handle their work independently. They can take the messages from the queue when they are able to process them. This way of handling messages creates a system that is easy to maintain and easy to scale.

**Message queuing - a simple use case**

Imagine that you have a web service that receives many requests every second, where no request is afford to get lost and all requests needs to be processed by a process that is time consuming.

Imagine that your web service always has to be highly available and ready to receive new request instead of being locked by the processing of previous received requests. In this case it is ideal to put a queue between the web service and the processing service. The web service can put the "start processing"-message on a queue and the other process can take and handle messages in order. The two processes will be decoupled from each other and does not need to wait for each other. If you have a lot of requests coming in a short amount of time, the processing system will be able to process them all anyway. The queue will persist requests if their number becomes really huge.

You can then imagine that your business and your workload is growing and you need to scale up your system. All that is needed to be done now is to add more workers, receivers, to work off the queues faster.

**RabbitMQ**

If you do start to consider a queue-based solution, CloudAMQP offers hosting of the message queue [RabbitMQ.](http://www.rabbitmq.com/)RabbitMQ is open source message-oriented middleware that implements the [Advanced Message Queuing Protocol (AMQP).](http://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol) AMQP have features like queuing, routing, reliability and security. You can read more about CloudAMQP[here.](https://www.cloudamqp.com/docs/index.html)

5 What is stateful server? 2

| Parameters | Stateful | Stateless |
| --- | --- | --- |
| 1. State | A Stateful server remember client data (state) from one request to the next. | A Stateless server keeps nostate information |
| 2. Programming | Statefulserver is harder to code | Statelessserver is straightforward to code |
| 3.Efficiency | MoreBecause clients do not have to provide full file information every time they perform an operation | Less because information needs to be provided |
| 4.Crash recovery | Difficult due to loss of information | Can easily recover from failure. Because there is no state that must be restored |
| 5.Information transfer | Using a Stateful,file server, the client can send less data with each request | Using a stateless file server, the client must,specify complete file names in each request specify location for reading or writing re-authenticate for each request. |
| 6.Extra services | Stateful servers can also offer clients extra services such as file locking, and remember read and write positions | It does not have to implement the state accounting associated with opening, closing, and locking of files. |
| 7.Operations | Open, Read, Write, Seek, Close | Read, Write |
| 8.Example,(Tip: you can draw them outside this two column format) | enter image description here | enter image description here |

6 Define Name resolution.

Name Spaces and Name Resolution

Names are organized into name spaces

A name space can be represented as a labeled, directed graph with two types of nodes

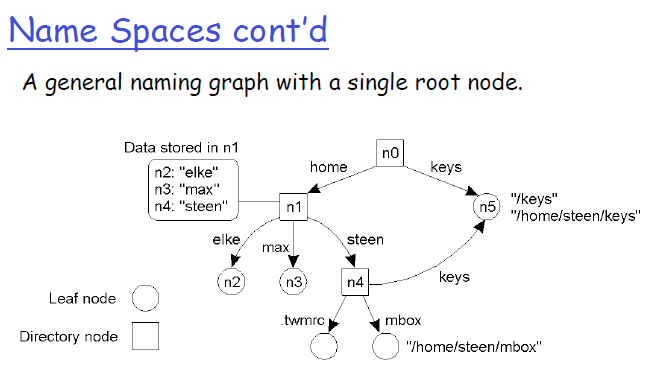
Leaf nodes and directory nodes

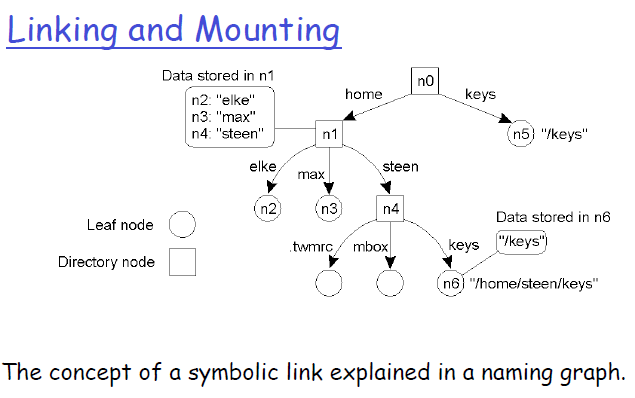
Absolute vs relative path names

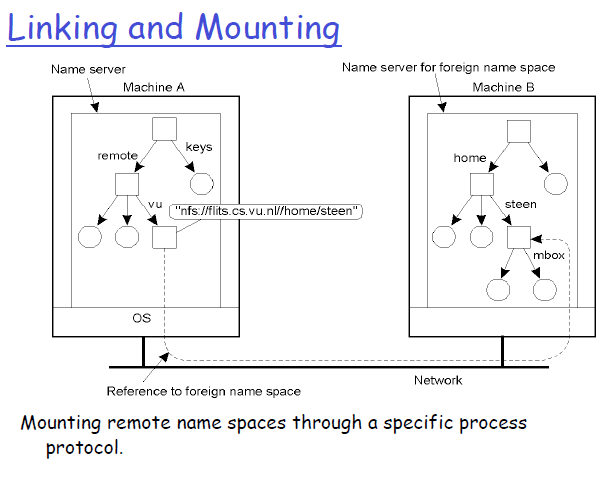
Local names vs global names

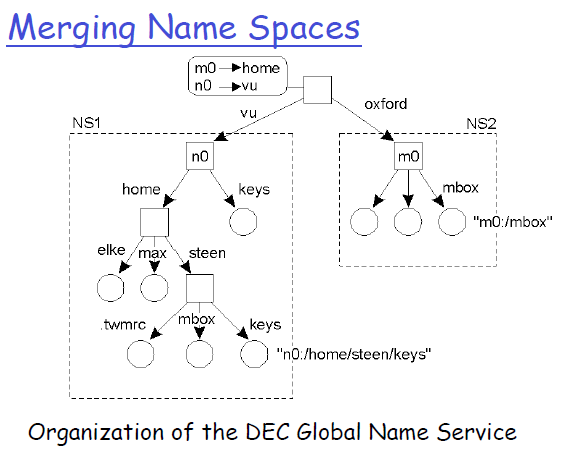
Name Resolution: the process of looking up a name

Closure mechanism: knowing where and how to start name resolution









3

7 What is Globe local objects? Explain their types. 3

**GLOBE**

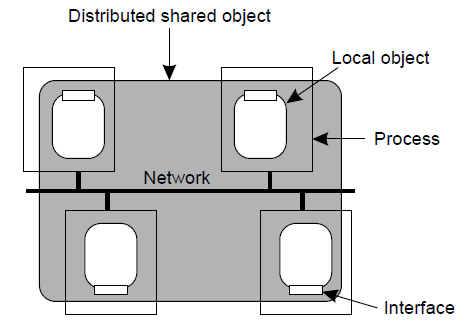
1. Experimental wide-area system currently being developed at Vrije Universiteit

2. Unique for its focus on scalability by means of truly distributed objects.

3. Prototype version up and running across multiplemachines distributed in NL and across Europe and the US.

**Object Model**

**Essence:** A Globe object is a **physically distributed shared object**: the object’s state may be physically distributed across several machines

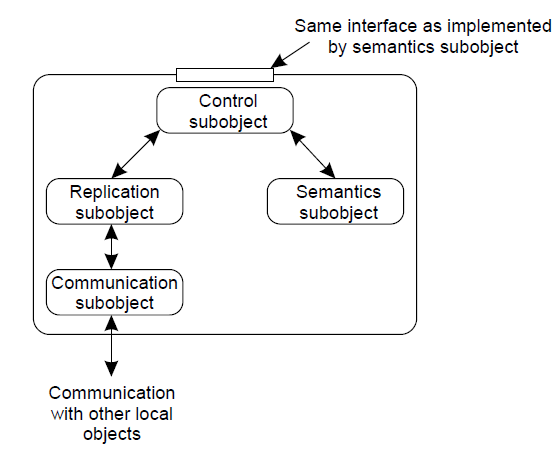


**Local object:** A non-distributed object residing a single address space, often representing a distributed shared object

**Contact point:** A point where clients can contact the distributed object; each contact point is described through a **contact address**

**Object Model**

**Observation:** Globe attempts to separate functionality from distribution by distinguishing different local Sub objects:



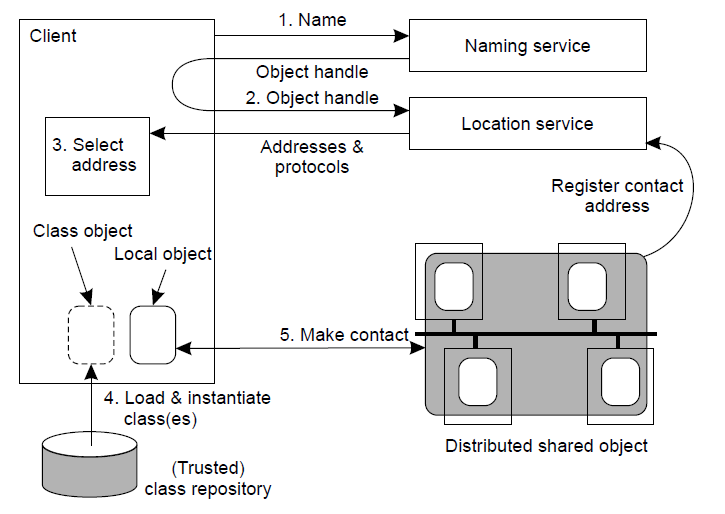
**Semantics sub-object:** Contains the methods that implement the functionality of the distributed shared object

**Communication sub-object:** Provides a (relatively simple), network-independent interface for communication between local objects

**Replication sub-object:** Contains the implementation of an **object-specific** consistency protocol that controls exactly when a method on the semantics sub-object may be invoked

**Control sub-object:** Connects the user-defined interfaces of the semantics sub-object to the generic, predefined interfaces of the replication sub-object

**Client-to-Object Binding**



**Observation:** Globe’s contact addresses correspond to CORBA’s object references

**GLOBE SERVICES**

|  |  |  |
| --- | --- | --- |
| **Service** | **Possible implementation** | **Av?** |
| Collection | Separate object that holds references to other objects | No |
| Concurrency | Each object implements its own concurrency control strategy | No |
| Transaction | Separate object representing a transaction manager | No |
| Event/Notif. | Separate object per group of events (as in DCOM) | No |
| Externalization | Each object implements its own marshaling routines | Yes |
| Life cycle | Separate class objects combined with per-object implementations | Yes |
| Licensing | Implemented by each object separately | No |
| Naming | Separate service, implemented by a collection of naming objects | Yes |
| Property | Separate service, implemented by a collection of directory objects | No |
| Persistence | Implemented on a per-object basis | Yes |
| Security | Implemented per object, combined with (local) security services | Yes |
| Replication | Implemented on a per-object basis | Yes |
| Fault tolerance | Implemented per object combined with fault-tolerant servers | Yes |

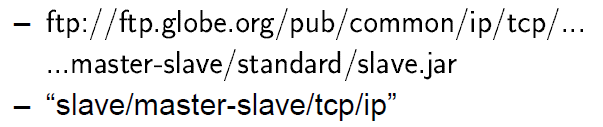
**Object References**

**Essence:** Globe uses location-independent object handles which are to be resolved to **contact addresses** (which describes **where** and **how** an object can be contacted):

1. Associated with a contact point of the distributed object

2. Specifies (for example) a transport-level network address to which the object will listen

3. Contains an **implementation handle**, specifying exactly what the client should implement if it wants to communicate through the contact point:



**Observation:** Objects in Globe have their own objectspecific implementations; there is no “standard” proxy that is implemented for all clients

**NAMING OBJECTS**

**Observation:** Globe separates naming from locating objects (as described in Chapter 04). The current naming service is based on DNS, using TXT records for storing object handles

**Observation:** The location service is implemented as a generic, hierarchical tree, similar to the approach explained in Chapter 04.

**CACHING AND REPLICATION**

**Observation:** Here’s where Globe differs from many other systems:

1. The organization of a local object is such that replication is inherently part of each distributed

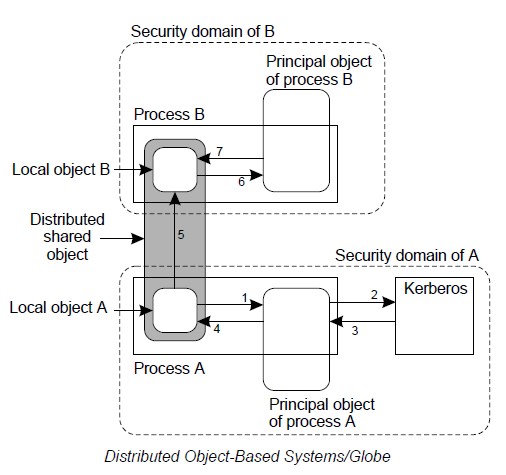
shared object

2. All replication sub-objects have the same interface:

|  |  |
| --- | --- |
| **Method** | **Description** |
| start | Called to synchronize replicas of the semantics subobjects, obtain locks if necessary, etc. |
| send | Provide marshaled arguments of a specific method, and pass invocation to local objects in other address spaces |
| invoked | Called after the control sub-object has invoked a specific method at the semantics subobject |

\_ This approach allows to implement any **object-specific** caching/replication strategy

**Security: Essence:** Additional security sub-object checks for authorized communication, invocation, and parameter values. Globe can be integrated with existing security services:

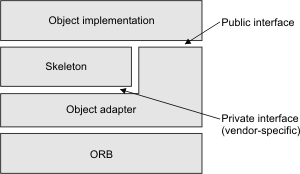


8 Explain the use of portable object adapter in CORBA. 2

An object adapter is the primary way for an object to access ORB services such as object reference generation. A portable object adapter exports standard interfaces to the object.

The main responsibilities of an object adapter are:

* Generation and interpretation of object references.
* Enabling method calling.
* Object and implementation activation and deactivation.
* Mapping object references to the corresponding object implementations.



For CORBA 2.1 and earlier, all ORB vendors implemented an object adapter, which was known as the basic object adapter. A basic object adapter could not be specified with a standard CORBA IDL. Therefore, vendors implemented the adapters in many different ways. The result was that programmers were not able to write server implementations that were truly portable between different ORB products. A first attempt to define a standard object adapter interface was done in CORBA 2.1. With CORBA v.2.3, the OMG group released the final corrected version of a standard interface for the object adapter. This adapter is known as the Portable Object Adapter (POA).

Some of the main features of the POA specification are to:

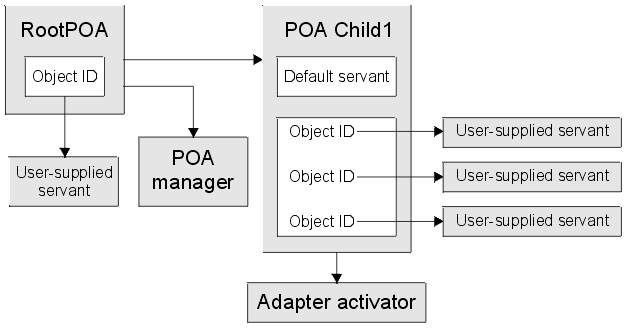
* Allow programmers to construct object and server implementations that are portable between different ORB products.
* Provide support for persistent objects. The support enables objects to persist across several server lifetimes.
* Support transparent activation of objects.
* Associate policy information with objects.
* Allow multiple distinct instances of the POA to exist in one ORB.

For more details of the POA, see the CORBA v.2.3 (formal/99-10-07) specification.

The IBM® ORB supports both the POA specification and the previous proprietary basic object adapter. By default, the RMI compiler, when used with the -iiop option, generates RMI-IIOP ties for servers. These ties are based on the basic object adapter. When a server implementation uses the POA interface, you must add the -poa option to the **rmic** compiler to generate the relevant ties.

To implement an object using the POA, the server application must obtain a POA object. When the server application calls the ORB method resolve\_initial\_reference("RootPOA"), the ORB returns the reference to the main POA object that contains default policies. For a list of all the POA policies, see the CORBA specification. You can create new POAs as child objects of the RootPOA. These child objects can contain different policies. This structure allows you to manage different sets of objects separately, and to partition the namespace of objects IDs.

Ultimately, a POA handles Object IDs and active servants. An active servant is a programming object that exists in memory. The servant is registered with the POA because one or more associated object identities was used. The ORB and POA cooperate to determine which servant starts the operation requested by the client. By using the POA APIs, you can create a reference for the object, associate an object ID, and activate the servant for that object. A map of object IDs and active servants is stored inside the POA. A POA also provides a default servant that is used when no active servant has been registered. You can register a particular implementation of this default servant. You can also register a servant manager, which is an object for managing the association of an object ID with a particular servant.



The POA manager is an object that encapsulates the processing state of one or more POAs. You can control and change the state of all POAs by using operations on the POA manager.

The adapter activator is an object that an application developer uses to activate child POAs.

9 Explain the characteristics of multimedia data. 3

**CHARACTERISTICS OF TYPICAL MULTIMEDIA STREAMS**

It is likely that multimedia applications will remain in the window of scarcity (Say resources sufficient or not) for the foreseeable future. Advances in system performance are likely to be used to improve the quality of multimedia data, to include higher frame rates and greater resolution for video streams or to support many media streams concurrently, for example in **a video conferencing** system. More demanding applications, including virtual reality and real-time stream manipulation (“special effects”) can extend the window of scarcity almost indefinitely.

The term ‘continuous’ refers to the **user’s view of the data**. Internally, continuous media are represented as sequences of discrete values which replace each other over time. For example, the value of an image array is replaced 25 times per second to give the impression of a TV-quality view of a moving scene; a sound amplitude value is replaced 8000 times per second to convey telephone quality speech.

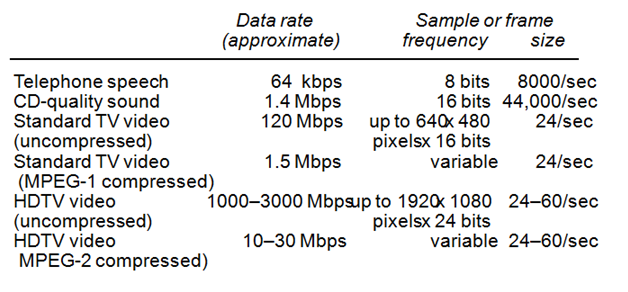
Multimedia streams are said to be time-based (or isochronous) because timed data elements in audio and video streams define the semantics or ‘content’ of the stream. The times at which the values are played or recorded affect the validity of the data. Hence systems that support multimedia applications need to preserve the timing when they handle continuous data.

Compression can reduce bandwidth requirements by factors between 10 and 100, but the timing requirements of continuous data are unaffected. Various compressed data formats such as GIF, TIFF and JPEG for still images and MPEG-1, MPEG-2 and MPEG-4 for video sequences.

For Compression we use of special-purpose hardware to process and despatch video and audio information – the video and audio coders/decoders (Codecs) found on video cards manufactured for personal computers. The compression method used for the MPEG video formats is asymmetric, with a complex compression algorithm and simpler decompression.

**The window of scarcity for computing and communication resources**





**TYPICAL INFRASTRUCTURE COMPONENTS FOR MULTIMEDIA APPLICATIONS**

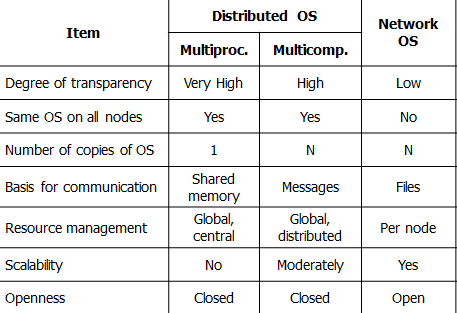


10 What is real-time scheduling? 2

Scheduling of real-time tasks is very different from general scheduling. Ordinary scheduling algorithms attempt to ensure fairness among tasks, minimum progress for any individual task, and prevention of starvation and deadlock. To the scheduler of real-time tasks, these goals are often superficial. The primary concern in scheduling real-time tasks is deadline compliance. Another difference between scheduling ordinary and real-time tasks is the dependence of the latter on the real-time clock. Because of this dependence, Dijkstra’s assertion that preemption is transparent to the task [Dij68] does not hold for real-time systems, making preemptive schedulers of real-time tasks much more complex than their “ordinary” counterparts.

PART – B (50 Marks)

11 a) Explain the main differences between network operating system and distributed operating system.



5 b) Explain the client-server architecture in detail. 5

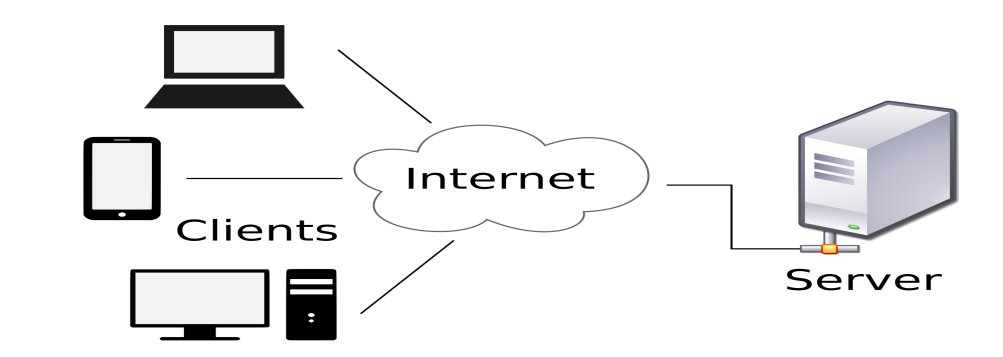
**Client-Server Architectures**

Various hardware and software architectures are used for distributed computing. At a lower level, it is necessary to interconnect multiple CPUs with some sort of network, regardless of whether that network is printed onto a circuit board or made up of loosely coupled devices and cables. At a higher level, it is necessary to interconnect [processes](https://en.wikipedia.org/wiki/Process_(computing)) running on those CPUs with some sort of [communication system](https://en.wikipedia.org/wiki/Communication_system).

Distributed programming typically falls into one of several basic architectures: [client–server](https://en.wikipedia.org/wiki/Client%E2%80%93server), [three-tier](https://en.wikipedia.org/wiki/Three-tier_(computing)), [n-tier](https://en.wikipedia.org/wiki/Multitier_architecture), or [peer-to-peer](https://en.wikipedia.org/wiki/Peer-to-peer); or categories: [loose coupling](https://en.wikipedia.org/wiki/Loose_coupling), or [tight coupling](https://en.wikipedia.org/wiki/Computer_cluster).

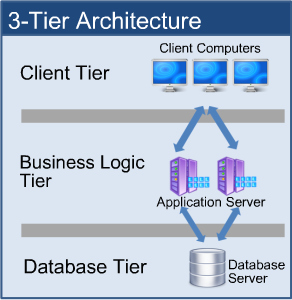
[**Client–server**](https://en.wikipedia.org/wiki/Client%E2%80%93server)

Architectures where smart clients contact the server for data then format and display it to the users. Input at the client is committed back to the server when it represents a permanent change.



[**Three-tier**](https://en.wikipedia.org/wiki/Three-tier_(computing))

Architectures that move the client intelligence to middle tier so that stateless clients can be used. This simplifies application deployment. Most web applications are three-tier.

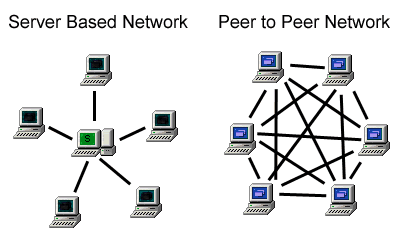


[**n-tier**](https://en.wikipedia.org/wiki/Multitier_architecture)

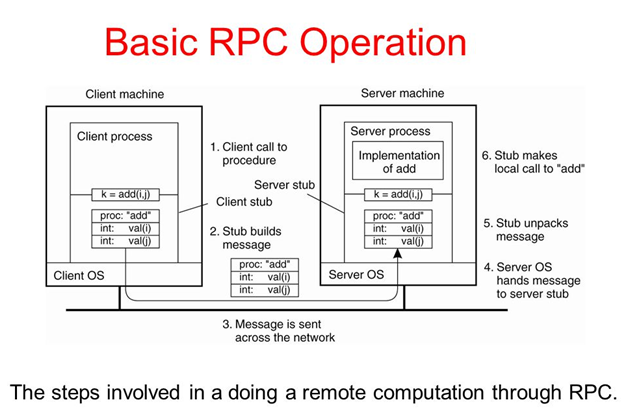
Architectures that refer typically to web applications which further forward their requests to other enterprise services. This type of application is the one most responsible for the success of [application servers](https://en.wikipedia.org/wiki/Application_server).

[**Peer-to-peer**](https://en.wikipedia.org/wiki/Peer-to-peer)

Architectures where there is no special machines that provide a service or manage the network resources. Instead all responsibilities are uniformly divided among all machines, known as peers. Peers can serve both as clients and as servers.

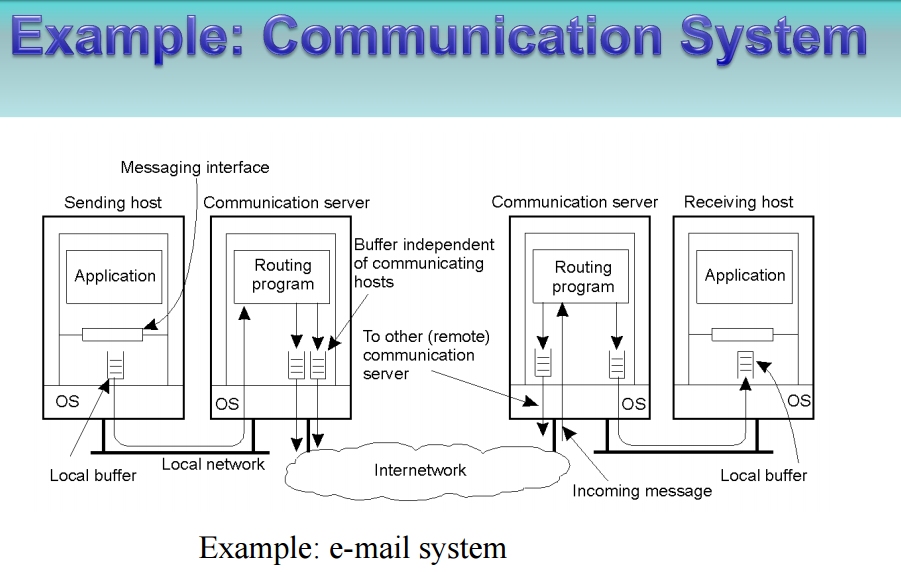


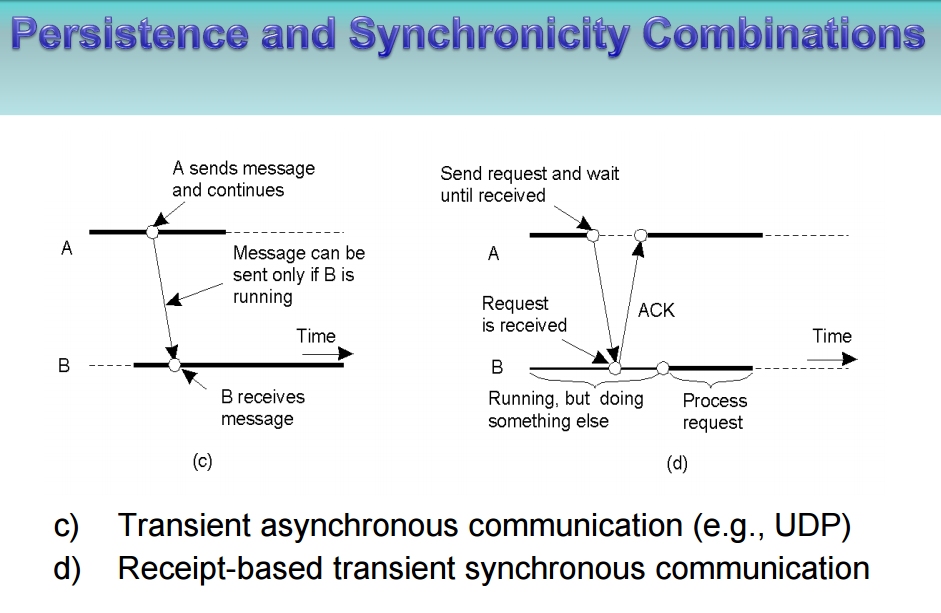
12 a) Describe about the basic RPC operation. 5



b) Explain about message oriented transient communication. 5

Message-oriented communication – Persistence and synchronicity – Message-oriented transient communication • Berkeley socket • MPI – Message-oriented persistent communication • Stream-oriented communication – Data stream – Quality of services – Stream synchronization





Transient communication – Messages are stored by communication middleware only for as long as the sending and receiving application are executing – Example: TCP/UDP

13 a) Explain the role of software agents in distributed system. 5

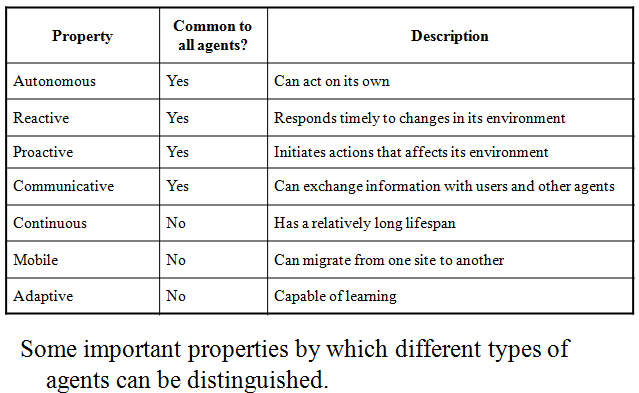
**SOFTWARE AGENTS**

* What is a software agent?
  + “An autonomous unit capable of performing a task in collaboration with other, possibly remote, agents”.
* The field of Software Agents is still immature, and much disagreement exists as to how to define what we mean by them.
* However, a number of types can be identified.

**TYPES OF SOFTWARE AGENTS**

1. Collaborative Agent – also known as “multi-agent systems”, which can work together to achieve a   
   common goal (e.g., planning a meeting).
2. Mobile Agent – code that can relocate and continue executing on a remote machine.
3. Interface Agent – software with “learning abilities” (that damned MS paperclip, and the ill-fated “bob”).
4. Information Agent – agents that are designed to collect and process geographically dispersed data and information.

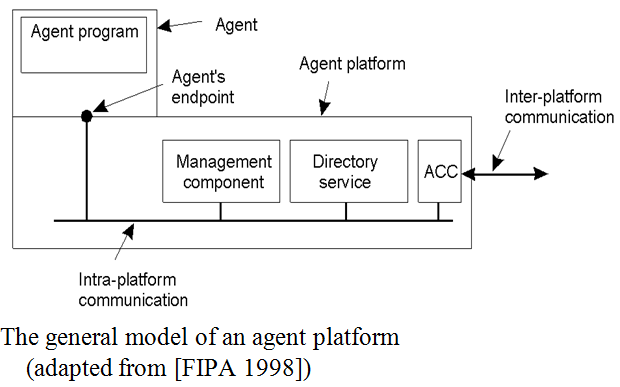
**Software Agents in Distributed Systems**

****

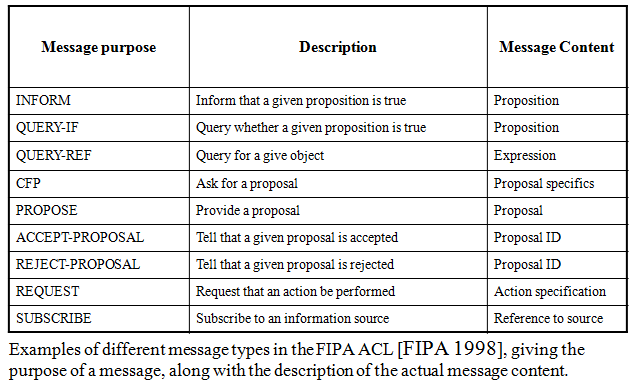
**Agent Technology – Standards**

* The general model of an agent platform has   
  been standardized by FIPA (“Foundation   
  for Intelligent Physical Agents”) located at [http://www.fipa.org](http://www.fipa.org/)
* Specifications include:
  + Agent Management Component.
  + Agent Directory Service.
  + Agent Communication Channel.
  + Agent Communication Language.

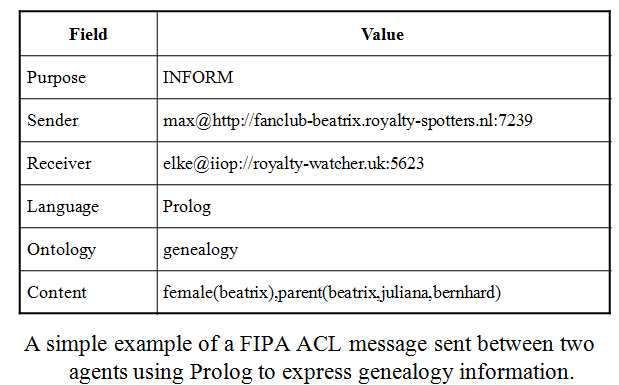
**AGENT TECHNOLOGY**



**AGENT COMMUNICATION LANGUAGES**



**AGENT COMMUNICATION LANGUAGES**



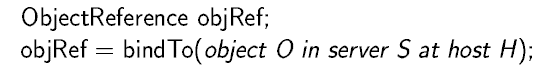
b) Discuss about the implementation of namespace. 5

**NAMING**

In CORBA, it is essential to distinguish specification-level and implementation-level object References

**Specification level:** An object reference is considered to be the same as a proxy for the referenced object having an object reference means you can directly invoke methods; there is no separate client to- object binding phase

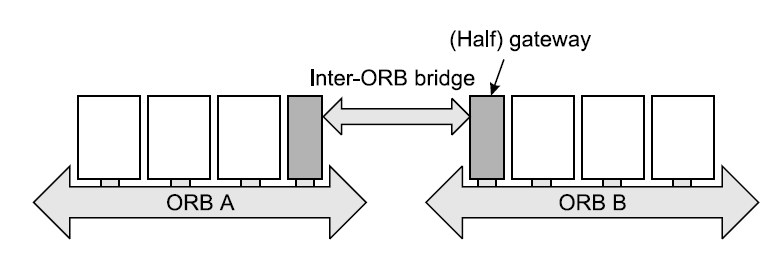
**Implementation level:** When a client gets an object reference, the implementation ensures that, one way or the other, a proxy for the referenced object is placed in the client address space.



**Conclusion:** Object references in CORBA used to be highly **implementation dependent**: different implementations of CORBA could normally not exchange their references.

**Interoperable Object References**

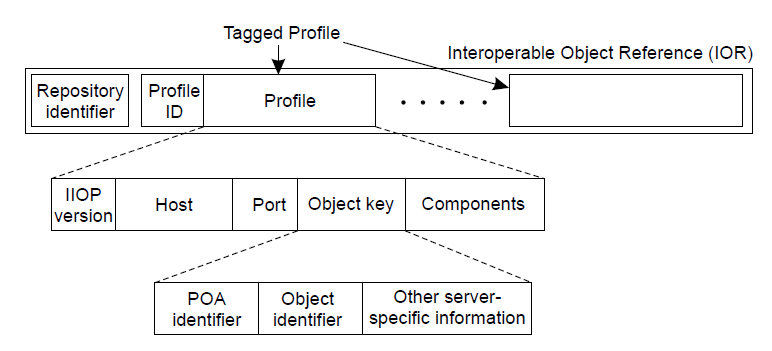
**Observation:** Recognizing that object references are implementation dependent, we need a separate referencing mechanism to cross ORB boundaries



**Solution:** Object references passed from one ORB to another are transformed by the bridge through which they pass (different transformation schemes can be implemented)

**Observation:** Passing an object reference from ORB A to ORB B circumventing the A-to-B bridge may be useless if ORB B doesn’t understand ref A.

**Observation:** To allow all kinds of *different* systems to communicate, we standardize the reference that is passed between bridges:



**NAMING SERVICE**

**Essence:** CORBA’s naming service allows servers to associate a name to an object reference, and have clients subsequently bind to that object by resolving its name

**Observation:** In most CORBA implementations, object references denote servers at specific hosts; naming makes it easier to relocate objects

**Observation:** In the naming graph all nodes are objects; there are no restrictions to binding names to objects 􀀀 CORBA allows arbitrary naming graphs

**Question:** How do you imagine cyclic name resolution stops?

**Observation:** There is no single root; an initial context node is returned through a special call to the ORB. Also: the naming service can operate *across* different ORBs🡺 **interoperable naming service**

14 a) Explain CORBA services in detail. 5

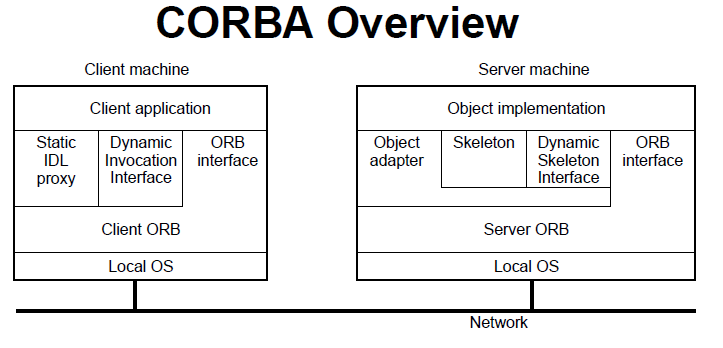
**CORBA (COMMON OBJECT REQUEST BROKER ARCHITECTURE)**

1. Developed by the Object Management Group (OMG) in response to industrial demands for object based middleware and currently in version #2.4 with #3 (almost) done

2. CORBA is a specification: different implementations of CORBA exist

3. Very much the work of a committee: there are over 800 members of the OMG and many of them have a say in what CORBA should look like

**Essence:** CORBA provides a simple distributed-object model, with specifications for many supporting services.



**Object Request Broker (ORB):** CORBA’s object broker that connects clients, objects, and services

**Proxy/Skeleton:** Precompiled code that takes care of un-marshaling invocations and results

**Dynamic Invocation/Skeleton Interface (DII/DSI):** To allow clients to “construct” invocation requests at runtime instead of calling methods at a proxy, and having the server-side“reconstruct” those request into regular method invocations

**Object adapter:** Server-side code that handles incoming invocation requests.

**Interface repository:** Database containing interface definitions and which can be queried at runtime.

**Implementation repository:** Database containing the implementation (code, and possibly also state) of objects. Effectively: a server that can launch object servers.

**CORBA OBJECT MODEL**

**Essence:** CORBA has a “traditional” remote-object model in which an object residing at an object server is remote accessible through proxies

**Observation:** All CORBA specifications are given by means of interface descriptions, expressed in an IDL. CORBA follows an interface-based approach to objects:

1. Not the objects, but interfaces are the really important entities

2. An object may implement one or more interfaces \_

3. Interface descriptions can be stored in an interface repository, and looked up at runtime

4. Mappings from IDL to specific programming are part of the CORBA specification (languages include C, C++, Smalltalk, COBOL, ADA, and Java.

**CORBA SERVICES**

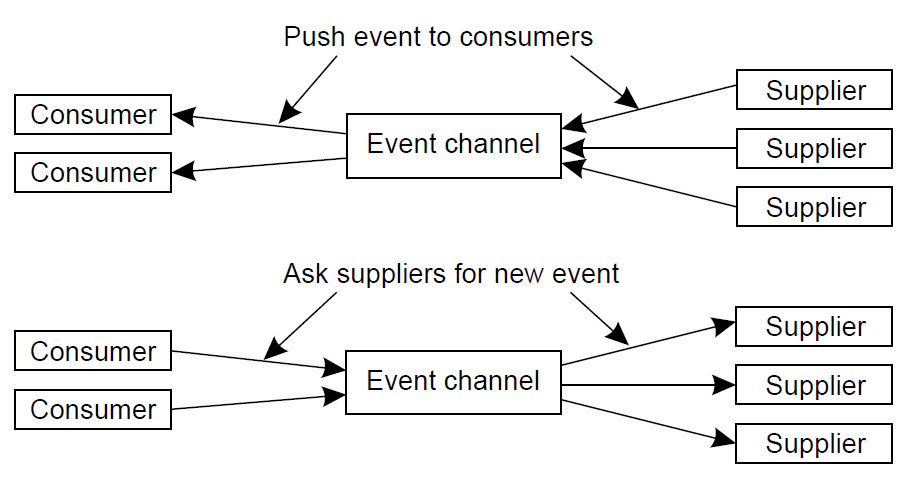
|  |  |
| --- | --- |
| **Service** | **Description** |
| Collection | Facilities for grouping objects into lists, queue, sets, etc. |
| Query | Facilities for querying collections of objects in a declarative manner |
| Concurrency | Facilities to allow concurrent access to shared objects. |
| Transaction | Flat and nested transactions on method calls over multiple objects |
| Event | Facilities for asynchronous communication through events |
| Notification | Advanced facilities for event-based asynchronous communication |
| Externalization | Facilities for marshaling and un-marshaling of objects |
| Life cycle | Facilities for creation, deletion, copying, and moving of objects |
| Licensing | Facilities for attaching a license to an object |
| Naming | Facilities for system wide naming of objects |
| Property | Facilities for associating (attribute, value) pairs with objects |
| Trading | Facilities to publish and find the services an object has to offer |
| Persistence | Facilities for persistently storing objects |
| Relationship | Facilities for expressing relationships between objects |
| Security | Mechanisms for secure channels, authorization, and auditing |
| Time | Provides the current time within specified error margins |

**COMMUNICATION MODELS**

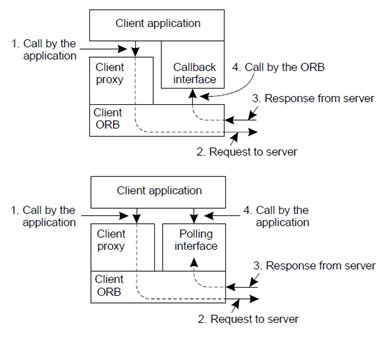
**Object invocations:** CORBA distinguishes three different forms of direct invocations:

|  |  |  |
| --- | --- | --- |
| **Request type** | **Failure sem.** | **Description** |
| Synchronous | At-most-once | Caller blocks |
| One-way | Unreliable | Non-blocking call |
| Deferred synchronous | At-most-once | Non-blocking, but can pickup results later |

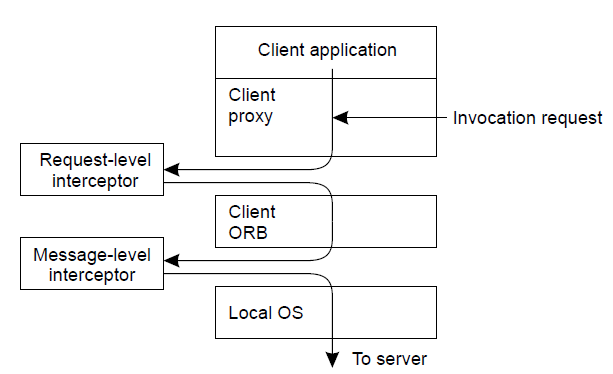
**Event communication:** There are also additional facilities by means of **event channels**:

****

**Messaging facilities:** reliable asynchronous and persistent method invocations:

**  
PROCESSES**

Most aspects of processes for in CORBA have been discussed in previous classes. What remains is the concept of **interceptors**:

****

**Request-level:** Allows you to modify invocation semantics (e.g., multicasting)

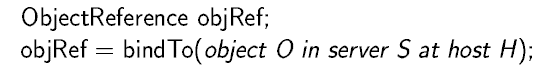
**Message-level:** Allows you to control message-passing between client and server (e.g., handle reliability and fragmentation)

**NAMING**

In CORBA, it is essential to distinguish specification-level and implementation-level object References

**Specification level:** An object reference is considered to be the same as a proxy for the referenced object having an object reference means you can directly invoke methods; there is no separate client to- object binding phase

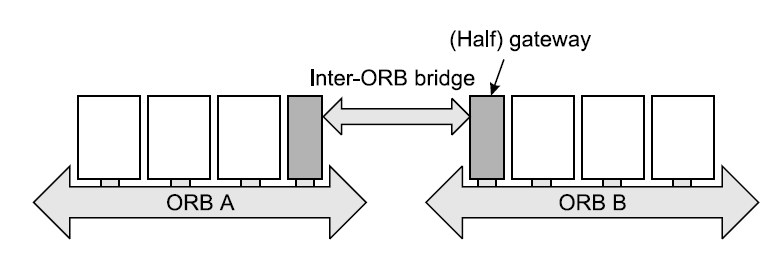
**Implementation level:** When a client gets an object reference, the implementation ensures that, one way or the other, a proxy for the referenced object is placed in the client address space.



**Conclusion:** Object references in CORBA used to be highly **implementation dependent**: different implementations of CORBA could normally not exchange their references.

**Interoperable Object References**

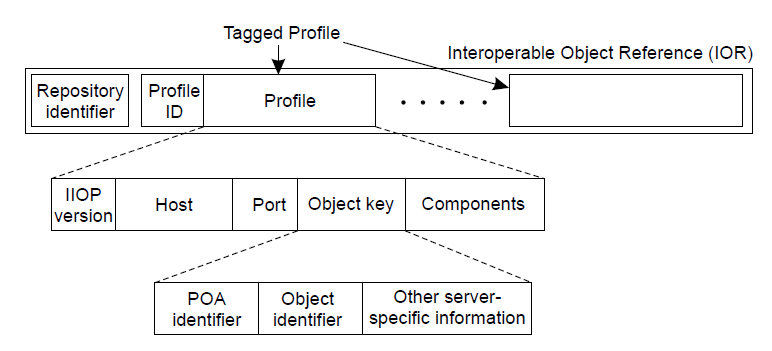
**Observation:** Recognizing that object references are implementation dependent, we need a separate referencing mechanism to cross ORB boundaries



**Solution:** Object references passed from one ORB to another are transformed by the bridge through which they pass (different transformation schemes can be implemented)

**Observation:** Passing an object reference from ORB A to ORB B circumventing the A-to-B bridge may be useless if ORB B doesn’t understand ref A.

**Observation:** To allow all kinds of *different* systems to communicate, we standardize the reference that is passed between bridges:



**NAMING SERVICE**

**Essence:** CORBA’s naming service allows servers to associate a name to an object reference, and have clients subsequently bind to that object by resolving its name

**Observation:** In most CORBA implementations, object references denote servers at specific hosts; naming makes it easier to relocate objects

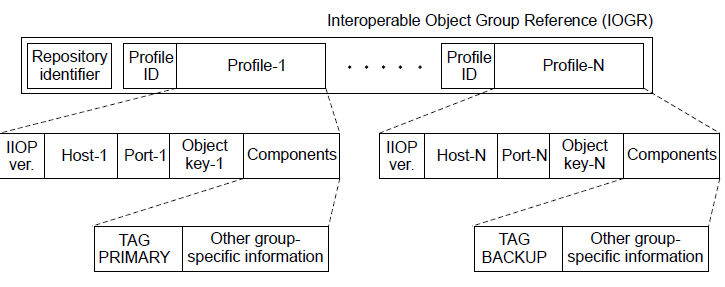
**Observation:** In the naming graph all nodes are objects; there are no restrictions to binding names to objects 􀀀 CORBA allows arbitrary naming graphs

**Question:** How do you imagine cyclic name resolution stops?

**Observation:** There is no single root; an initial context node is returned through a special call to the ORB. Also: the naming service can operate *across* different ORBs🡺 **interoperable naming service**

**FAULT TOLERANCE**

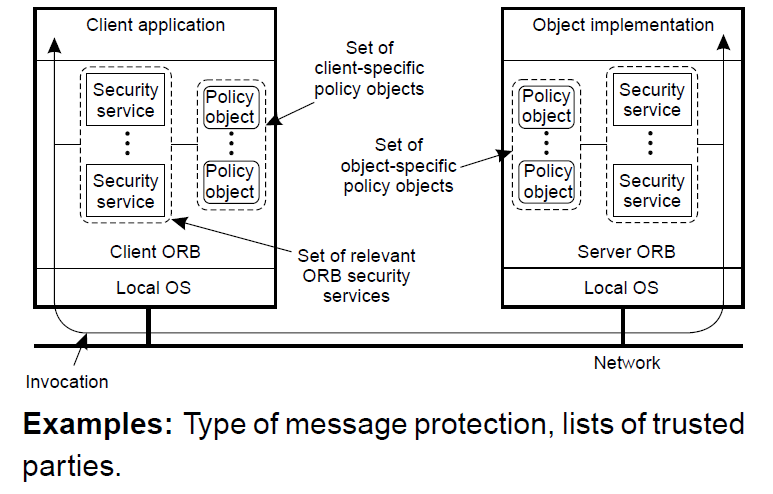
**Essence:** Mask failures through replication, by putting objects into **object groups**. Object groups are transparent to clients: they appear as “normal” objects. This approach requires a separate type of object reference: **Interoperable Object Group Reference**:



**Note:** IOGRs have the same structure as IORs; the main difference is that they are *used* differently. In IORs an additional profile is used as an alternative; in IOGR, it denotes another replica.

**SECURITY**

**Essence:** Allow the client and object to be mostly unaware of all the security policies, except perhaps at binding time; the ORB does the rest. Specific policies are passed to the ORB as (local) objects and are invoked when necessary:



**DISTRIBUTED COM (DCOM: DISTRIBUTED COMPONENT OBJECT MODEL)**

1. Microsoft’s solution to establishing inter-process communication, possibly across machine boundaries.

2. Supports a primitive notion of distributed objects

3. Evolved from early Windows versions to current NT-based systems (including Windows 2000)

4. Comparable to CORBA’s object request broker

**b) What is Globe? Explain the steps of binding a process to an object in globe. 5**

**GLOBE**

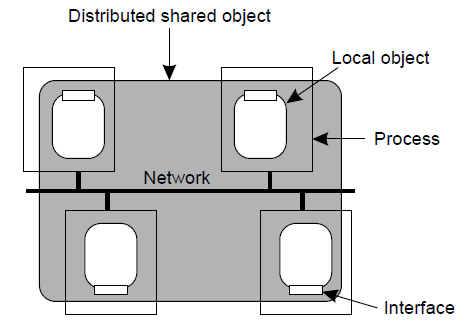
1. Experimental wide-area system currently being developed at Vrije Universiteit

2. Unique for its focus on scalability by means of truly distributed objects.

3. Prototype version up and running across multiplemachines distributed in NL and across Europe and the US.

**Object Model**

**Essence:** A Globe object is a **physically distributed shared object**: the object’s state may be physically distributed across several machines

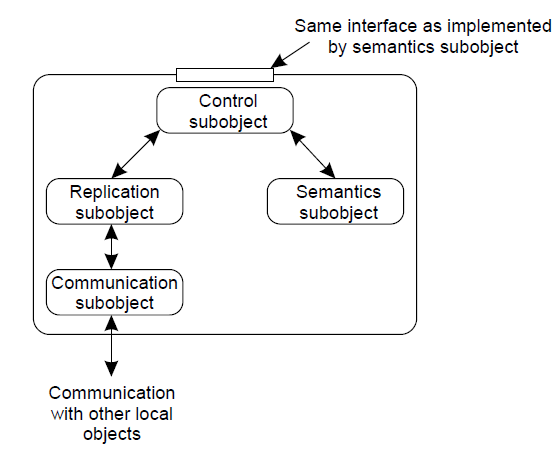


**Local object:** A non-distributed object residing a single address space, often representing a distributed shared object

**Contact point:** A point where clients can contact the distributed object; each contact point is described through a **contact address**

**Object Model**

**Observation:** Globe attempts to separate functionality from distribution by distinguishing different local Sub objects:



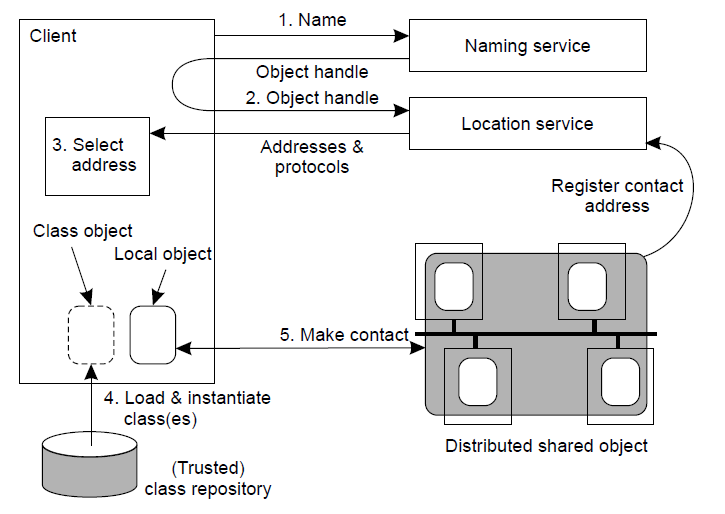
**Semantics sub-object:** Contains the methods that implement the functionality of the distributed shared object

**Communication sub-object:** Provides a (relatively simple), network-independent interface for communication between local objects

**Replication sub-object:** Contains the implementation of an **object-specific** consistency protocol that controls exactly when a method on the semantics sub-object may be invoked

**Control sub-object:** Connects the user-defined interfaces of the semantics sub-object to the generic, predefined interfaces of the replication sub-object

**Client-to-Object Binding**



**Observation:** Globe’s contact addresses correspond to CORBA’s object references

**GLOBE SERVICES**

|  |  |  |
| --- | --- | --- |
| **Service** | **Possible implementation** | **Av?** |
| Collection | Separate object that holds references to other objects | No |
| Concurrency | Each object implements its own concurrency control strategy | No |
| Transaction | Separate object representing a transaction manager | No |
| Event/Notif. | Separate object per group of events (as in DCOM) | No |
| Externalization | Each object implements its own marshaling routines | Yes |
| Life cycle | Separate class objects combined with per-object implementations | Yes |
| Licensing | Implemented by each object separately | No |
| Naming | Separate service, implemented by a collection of naming objects | Yes |
| Property | Separate service, implemented by a collection of directory objects | No |
| Persistence | Implemented on a per-object basis | Yes |
| Security | Implemented per object, combined with (local) security services | Yes |
| Replication | Implemented on a per-object basis | Yes |
| Fault tolerance | Implemented per object combined with fault-tolerant servers | Yes |

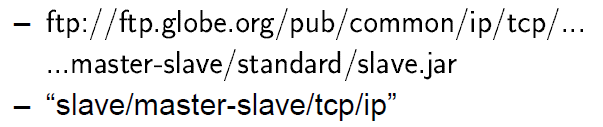
**Object References**

**Essence:** Globe uses location-independent object handles which are to be resolved to **contact addresses** (which describes **where** and **how** an object can be contacted):

1. Associated with a contact point of the distributed object

2. Specifies (for example) a transport-level network address to which the object will listen

3. Contains an **implementation handle**, specifying exactly what the client should implement if it wants to communicate through the contact point:



**Observation:** Objects in Globe have their own objectspecific implementations; there is no “standard” proxy that is implemented for all clients

**NAMING OBJECTS**

**Observation:** Globe separates naming from locating objects (as described in Chapter 04). The current naming service is based on DNS, using TXT records for storing object handles

**Observation:** The location service is implemented as a generic, hierarchical tree, similar to the approach explained in Chapter 04.

**CACHING AND REPLICATION**

**Observation:** Here’s where Globe differs from many other systems:

1. The organization of a local object is such that replication is inherently part of each distributed

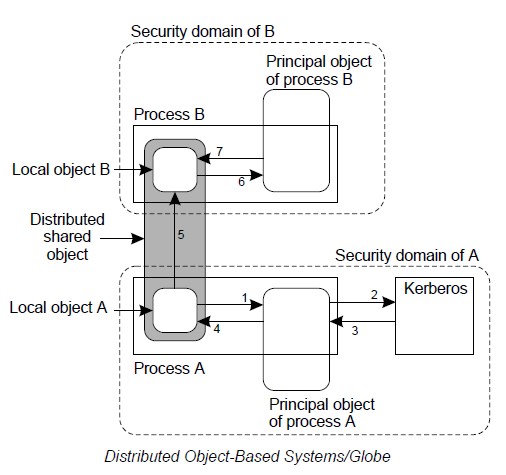
shared object

2. All replication sub-objects have the same interface:

|  |  |
| --- | --- |
| **Method** | **Description** |
| start | Called to synchronize replicas of the semantics subobjects, obtain locks if necessary, etc. |
| send | Provide marshaled arguments of a specific method, and pass invocation to local objects in other address spaces |
| invoked | Called after the control sub-object has invoked a specific method at the semantics subobject |

\_ This approach allows to implement any **object-specific** caching/replication strategy

**Security: Essence:** Additional security sub-object checks for authorized communication, invocation, and parameter values. Globe can be integrated with existing security services:



15 Discuss the quality of service management in distributed multimedia system in detail. 10

**QUALITY OF SERVICE MANAGEMENT**

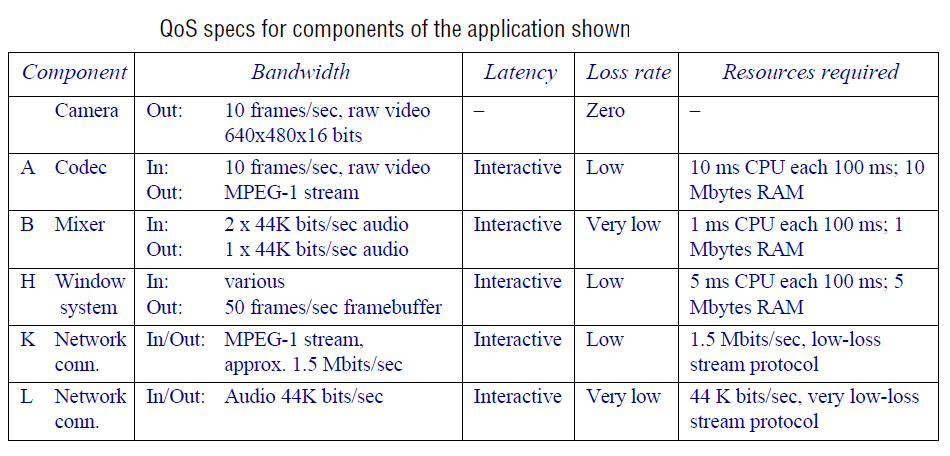
The management and allocation of resources to provide such guarantees is referred to as **quality of service management.**

**QOS MANAGER** is a system component of **QUALITY OF SERVICE MANAGEMENT** responsible for the allocation and scheduling of those resources.

When multimedia applications run in networks of personal computers they compete for resources at the workstations running the applications (processor cycles, bus cycles, buffer capacity) and in the networks (physical transmission links, switches, gateways). Workstations and networks may have to support several multimedia and conventional applications. There is competition between the multimedia and conventional applications, between different multimedia applications and even between the media streams within individual applications.

The concurrent use of physical resources for a variety of tasks has long been possible with multi-tasking operating systems and shared networks. In multi-tasking operating systems the central processor is allocated to individual tasks (or processes) in a round-robin or other scheduling scheme that shares the processing resources on a best-efforts basis amongst all of the tasks currently competing for the central processor.

Collisions can occur and when they do sending nodes wait for random back off periods in order to prevent repeated collisions.



**QOS MANAGER’S TWO MAIN SUB-TASKS:**

1. Quality of service negotiation. 2. Admission control.

**QUALITY OF SERVICE NEGOTIATION**

The application indicates its resource requirements to the QoS manager. The QoS manager evaluates the feasibility of meeting the requirements against a database of the available resources and current resource commitments and gives a positive or negative response. If it is negative, the application may be reconfigured to use reduced resources and the process is repeated.

To negotiate QoS between an application and its underlying system, an application must specify its QoS requirements to the QoS Manager. This is done by the transmission of a set of parameters. Three parameters are of primary interest when it comes to processing and transporting multimedia streams: bandwidth, latency, and loss rate.

**BANDWIDTH:** It of a multimedia stream or component is the rate at which data flows through it. Bandwidth is the characterization of burstiness.

**LATENCY:** It is the time required for an individual data element to move through a stream from the source to the destination. Of course this may vary depending on the volume of other data in the system and other characteristics of the system load. This variation is termed jitter – formally, jitter is the first derivative of the latency.

**LOSS RATE:** Since the late delivery of multimedia data is of no value, data elements will be dropped when it is impossible to deliver them before their scheduled delivery time. To avoid the data loss we either use leaky bucket or token bucket algorithm.

**TRAFFIC SHAPING ALGORITHMS**



**NOTE:**

The model of linear-bounded arrival processes (LBAP) used in defines the maximum number of messages in a stream during any time interval t as Rt + B where R is the rate and B is the maximum size of burst. The burst parameter defines the amount of buffer space required to avoid loss.

**ADMISSION CONTROL**

If the result of the resource evaluation is positive, the requested resources are reserved and the application is given a Resource Contract, stating the resources that have been reserved. The contract includes a time limit. The application is then free to run. If it changes its resource requirements it must notify the QoS Manager. If the requirements decrease, the resources released are returned to the database as available resources. If they increase, a new round of

negotiation and admission control is initiated.

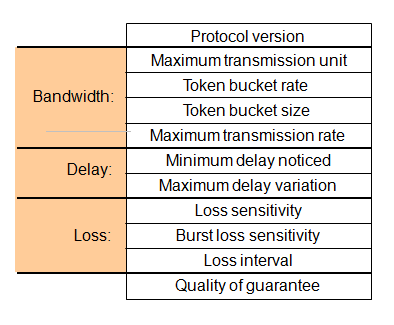
**TRAFFIC SHAPING**

Traffic shaping is the term used to describe the use of output buffering to smooth the flow of data elements. The bandwidth parameter of a multimedia stream typically provides an idealistic approximation of the actual traffic pattern that will occur when the stream is transmitted. The closer the actual traffic pattern matches the description, the better a system will be able to handle the traffic, in particular when it uses scheduling methods that are designed for periodic requests.

The LBAP model of bandwidth variations calls for regulation of the burstiness of multimedia streams. Any stream can be regulated by inserting a buffer at the source and by defining a method by which data elements leave the buffer.

**NOTE:**

**For traffic shaping we user flow specification format (RFC 1363 Flow Spec.).**



**FLOW SPECIFICATIONS**

A collection of QoS parameters is typically known as a flow specification, or flow spec for short. Several examples of flow specs exist and are all similar. In Internet RFC 1363, a flow spec is defined as eleven 16-bit numeric values (Figure Above) that reflect the QoS parameters discussed above in the following way:

• The maximum transmission unit and maximum transmission rate determine the maximum bandwidth required by the stream.

• The token bucket size and rate determine the burstiness of the stream.

• The delay characteristics are specified by the minimum delay that an application can notice (since we wish to avoid over-optimization for short delays) and maximum jitter it can accept.

• The loss characteristics are defined by the total acceptable number of losses over a certain interval and the maximum number of consecutive losses.

**NEGOTIATION PROCEDURES**

For distributed multimedia applications, the components of a stream are likely to be located in several nodes. There will be a QoS manager at each node. A straightforward approach to QoS negotiation is to follow the flow of data along each stream from the source to the target. A source component initiates the negotiation by sending out a flow spec to its local QoS manager.

**ADMISSION CONTROL** regulates access to resources to avoid resource overload and to protect resources from requests that they cannot fulfil. It involves turning down service requests should the resource requirements of a new multimedia stream would violate existing QoS guarantees.

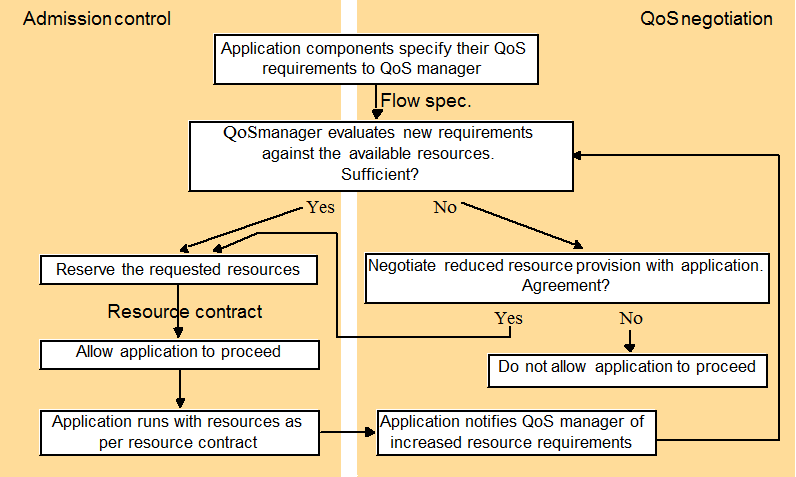
An admission control scheme is based on some knowledge of both the overall system capacity and the load generated by each application. The bandwidth requirement specification for an application may reflect the maximum amount of bandwidth that an application will ever require, the minimum bandwidth it will need to function, or some average value in between. Correspondingly, an admission control scheme may base its resource allocation on any of these values.

For resources that have a single allocator, admission control is straightforward. Resources that have distributed access points, such as many local area networks, require either a centralized admission control entity or some distributed admission control algorithm that avoids conflicting concurrent admissions. Bus arbitration within workstations falls into this category – however, even multimedia systems that perform bandwidth allocation extensively do not control bus admission as bus bandwidth is not considered to be in the window of scarcity.

**STATISTICAL MULTIPLEXING** Because of the potential under-utilization that can occur, it is common to overbook resources. The resulting guarantees, often called statistical or soft

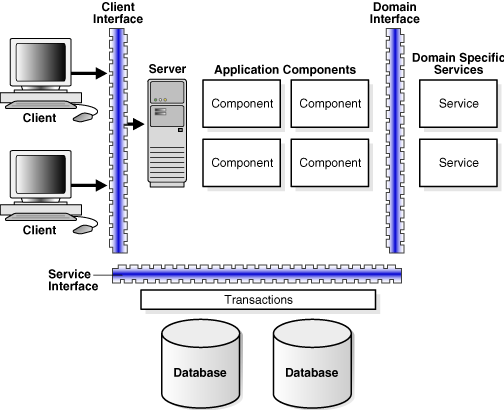
guarantees to distinguish them from the deterministic or hard guarantees introduced before, are only valid with some (usually very high) probability. Statistical guarantees tend to provide better resource utilization as they do not consider the worst case.

Statistical multiplexing is based on the hypothesis that for a large number of streams the aggregate bandwidth required remains nearly constant regardless of the bandwidth of individual streams.



16 a) Explain the role of middleware in distributed system. 5

**Middleware** is the software that connects software components or enterprise applications. **Middleware** is the software layer that lies between the operating**system** and the applications on each side of a**distributed** computer network (Figure 1-1). Typically, it supports complex, **distributed** business software applications.



b) What is stream oriented communication? Explain. 5

Text messages would be a message **oriented** protocol as each text message is distinct from the other messages. A phone call is **stream oriented** as there is a continuous flow of audio throughout the call. Common protocols used on the internet are UDP (message **oriented**) and TCP (**stream oriented**).

Message Oriented protocols send data in distinct chunks or groups. The receiver of data can determine where one message ends and another begins. Stream protocols send a continuous flow of data.

Here is an example with mobile phones. Text messages would be a message oriented protocol as each text message is distinct from the other messages. A phone call is stream oriented as there is a continuous flow of audio throughout the call.

Common protocols used on the internet are UDP (message oriented) and TCP (stream oriented). Wikipedia these terms for more information.

17 Write short notes on the following :

a) Threads in distributed system 5

**PROCESS**

A program under execution is called as a process.

**THREAD**

1. It is a light weight program.

2. **Traditional operating systems**: concerned with the “local” management and scheduling of processes.

3. **Modern distributed systems**: a number of other issues are of equal importance.

4. **There are three main areas of study**

a. Threads and virtualization within clients/servers.

b. Process and code migration.

c. Software agents.

5. Modern OSs provide “virtual processors” within which programs execute.

6. A programs execution environment is documented in the process table and assigned   
a PID.

7. To achieve acceptable performance in distributed systems, relying on the OS’s idea   
of a process is often not enough – finer granularity is required.

* The solution: Threading.

**PROBLEMS WITH PROCESSES**

1. Creating and managing processes is generally   
regarded as an expensive task (fork system call).

2. Making sure all the processes peacefully co-exist on the system is not easy (as concurrency transparency comes at a price).

3. **Threads** can be thought of as an “execution of a part of a program (in user-space)”.

4. Rather than make the OS responsible for concurrency transparency, it is left to the individual application to manage the creation and scheduling of each thread.

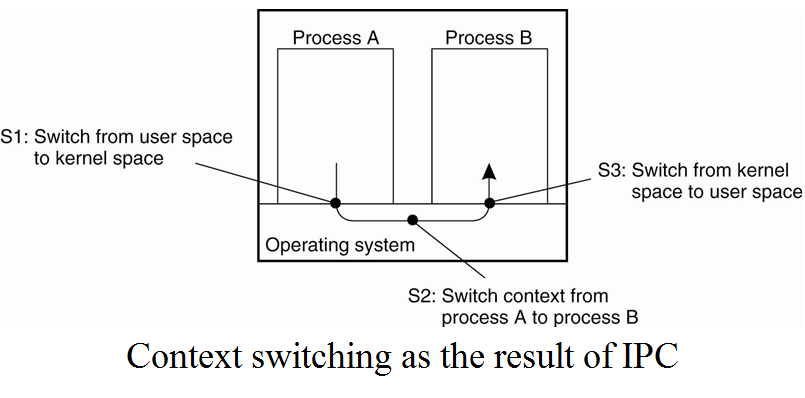
**Two Important Implications**

1.Threaded applications often run faster than non-threaded applications (as context-switches between kernel and user-space are avoided).

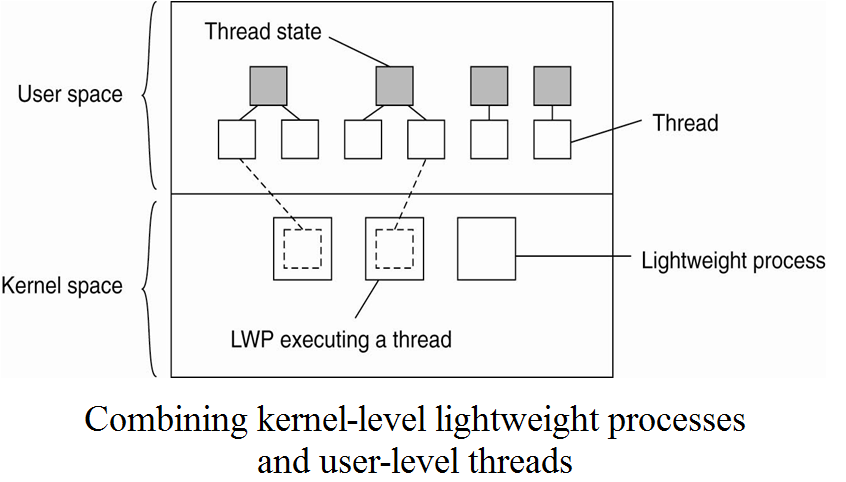
**2.** Threaded applications are harder to develop (although simple, clean designs can help here).

Additionally, the assumption is that the development environment provides a   
Threads Library for developers to use (most modern environments do).

**THREAD USAGE IN NON-DISTRIBUTED SYSTEMS**



**THREAD IMPLEMENTATION**

****

**THREADS IN NON-DISTRIBUTED SYSTEMS**

**Advantages:**

1.Blocking can be avoided

2. Excellent support for multi-processor systems (each running their own thread).

3. Expensive context-switches can be avoided.

4. For certain classes of application, the design and implementation is made considerably easier.

**THREADS IN DISTRIBUTED SYSTEMS**

**1.** Important characteristic: a blocking call in a thread does not result in the entire process   
being blocked.

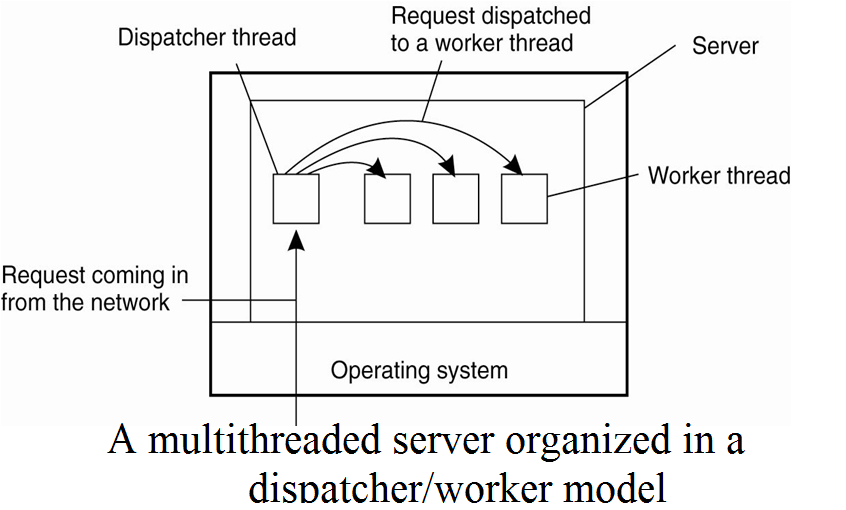
**2.** This leads to the key characteristic of threads within distributed systems:

“We can now express communications in the form of maintaining multiple logical connections at the same time (as opposed to a single, sequential, blocking process).”

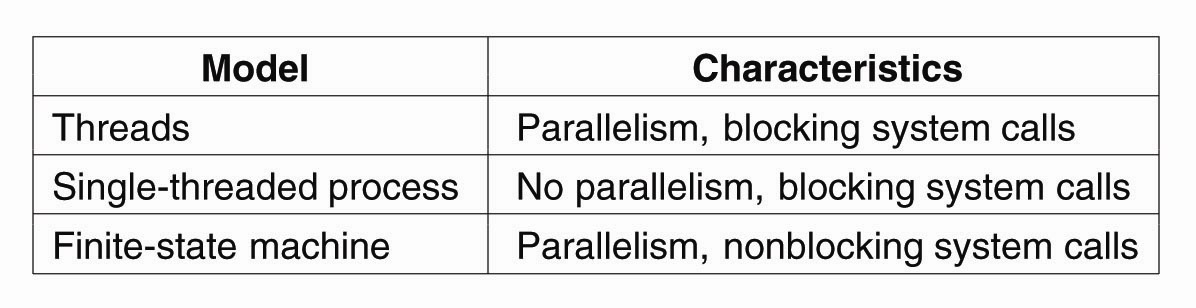
**Example: MT Clients and Servers**

* Mutli-Threaded Client: to achieve acceptable levels of perceived performance, it is often necessary to hide communications latencies.
* Consequently, a requirement exists to start communications while doing something else.
* Example: modern Web browsers.
* This leads to the notion of “truly parallel streams of data” arriving at a multi-threaded client application.
* Although threading is useful on clients, it is much more useful in distributed systems servers.
* The main idea is to exploit parallelism to attain high performance.
* A typical design is to organize the server as a single “dispatcher” with multiple threaded “workers”, as diagrammed overleaf.

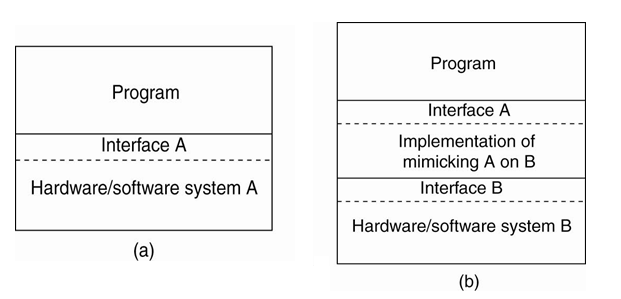
**MULTITHREADED SERVERS**



**MULTITHREADED SERVERS**

****

**THE ROLE OF VIRTUALIZATION IN DISTRIBUTED SYSTEMS**

****

**a. General organization between a program, interface, and system**

**b. General organization of virtualizing system A on top of system B**

b) Security in DCOM 5

**DCOM** (**Distributed Component Object Model**) is a set of Microsoft concepts and program interfaces in which client program object s can request services from server program objects on other computers in a network. ... **DCOM** comes as part of the Windows operating **systems**.

DCOM (Distributed Component Object Model) is a set of Microsoft concepts and program interfaces in which [client](http://searchenterprisedesktop.techtarget.com/definition/client) program [object](http://searchsoa.techtarget.com/definition/object) s can request services from [server](http://whatis.techtarget.com/definition/server) program objects on other computers in a network. DCOM is based on the [Component Object Model](http://searchwinit.techtarget.com/definition/Component-Object-Model) (COM), which provides a set of interfaces allowing clients and servers to communicate within the same computer (that is running Windows 95 or a later version).

For example, you can create a page for a Web site that contains a script or program that can be processed (before being sent to a requesting user) not on the Web site server but on another, more specialized server in the network. Using DCOM interfaces, the Web server site program (now acting as a client [object](http://searchsoa.techtarget.com/definition/object) ) can forward a Remote Procedure Call ( [RPC](http://searchsoa.techtarget.com/definition/Remote-Procedure-Call) ) to the specialized server object, which provides the necessary processing and returns the result to the Web server site. It passes the result on to the Web page viewer.

DCOM can also work on a network within an enterprise or on other networks besides the public Internet. It uses [TCP/IP](http://searchnetworking.techtarget.com/definition/TCP-IP) and [Hypertext Transfer Protocol](http://searchwindevelopment.techtarget.com/definition/HTTP) . DCOM comes as part of the Windows operating systems. DCOM is or soon will be available on all major UNIX platforms and on IBM's large server products. DCOM replaces OLE Remote Automation.

DCOM is generally equivalent to the Common Object Request Broker Architecture ( [CORBA](http://searchsqlserver.techtarget.com/definition/CORBA) ) in terms of providing a set of distributed services. DCOM is Microsoft's approach to a network-wide environment for program and data objects. CORBA is sponsored by the rest of the information technology industry under the auspices of the Object Management Group ( [OMG](http://searchsoa.techtarget.com/definition/Object-Management-Group) ).