Chapter 01

Intersystem Communications

Email:begna19mrblgo@gmail.com
***visit**:*   https://begnafrique.wordpress.com/

Outlines

- ❑ Architectures for integrating systems
- ❑ Web Services and Middleware
- ❑ Network Programming
- ❑ Message and queuing services
- ❑ Low level data communications

Intersystem Communication..

- In ordinary uniprocessor system there is one processor and one memory unit.
- The operation of this computer is sequential.
- The performance of single processor system is limited by the underlying fabrication technology.

Solution ?

- **Distributed computing** has become very popular as it provides a means to overcome the limitations imposed by sequential computers.
- The means for communication among processors, memory modules and other devices of a parallel computer is the Interconnection Network.

Intersystem Communication..

- Intersystem communication is used in distributed computing.
- A distributed system is basically a computer network.
- A distributed system relies entirely on the underlying computer network for the communication of data and control information between the nodes of which they are composed

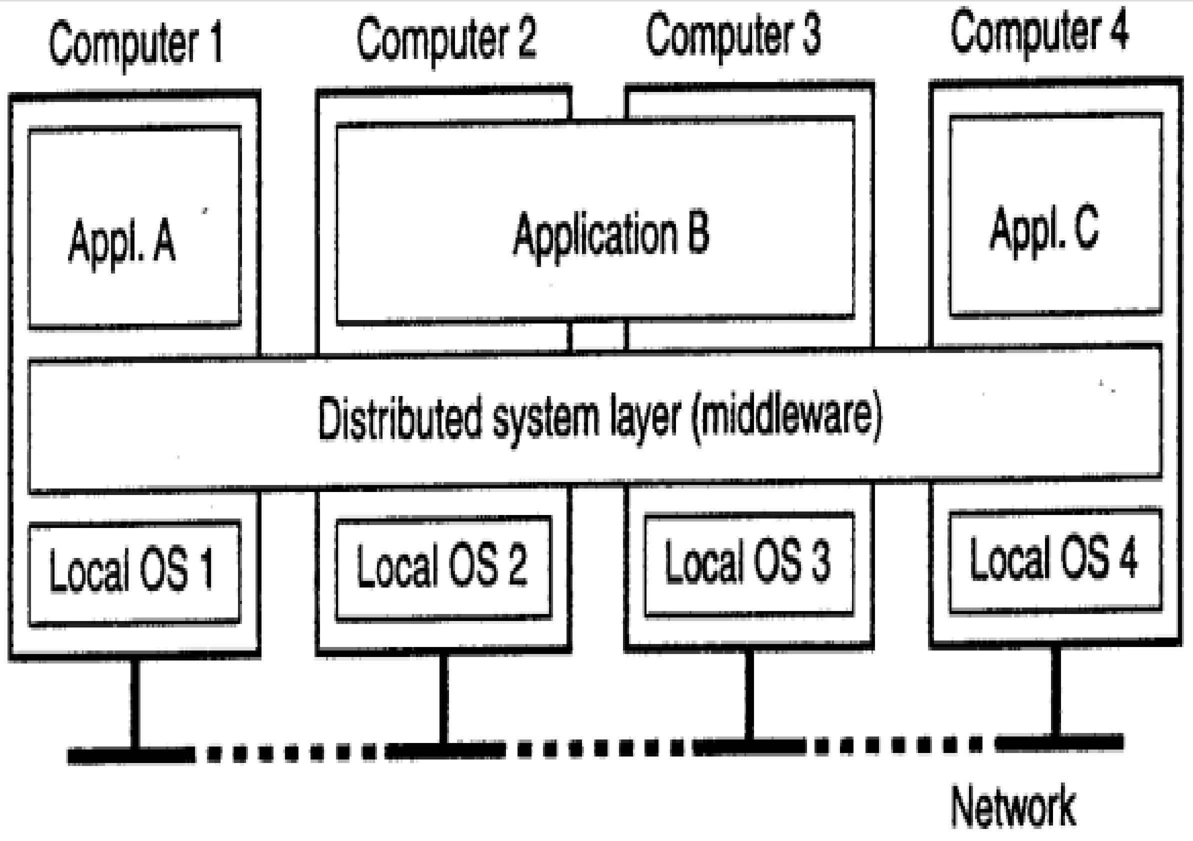
Intersystem Communication..

- A distributed system is a collection of independent computers that appears to its users as a single coherent system.
- This definition has several important aspects.
  - The first one is that a distributed system consists of components (i.e., computers) that are autonomous.
  - A second aspect is that users (be they people or programs) think they are dealing with a single system.
- This means that one way or the other the autonomous components need to collaborate.

Intersystem Communication..

- In order to support heterogeneous computers and networks while offering a single-system view, distributed systems are often organized by means of a layer of software-that is, logically placed between a higher-level layer consisting of users and applications, and a layer underneath consisting of operating systems and basic communication facilities.
- Accordingly, such a distributed system is sometimes called middleware.

Intersystem Communication..



Architectures for Integrating Systems

- Distributed systems are often complex pieces of software of which the components are by definition dispersed across multiple machines.
- To master their complexity, it is crucial that these systems are properly organized.
- There are different ways on how to view the organization of a distributed system,
- The organization of distributed systems is mostly about the software components that constitute the system.

Architectures for Integrating Systems..

- The software architectures tell us how the various software components are to be organized and how they should interact.
- The actual realization of a distributed system requires that we instantiate and place software components on real machines.
- There are many different choices that can be made in doing so.

DCOM as DS technology Solution

- It is the distributed version of Microsoft's COM technology which allows the creation and use of binary objects/components from languages other than the one they were originally written in.
- It currently supports Java(J++),C++, Visual Basic, JScript, and VBScript.
- DCOM works over the network by using proxy's and stubs.

DCOM as DS technology Solution

- When the client instantiates a component whose registry entry suggests that it resides outside the process space
- DCOM creates a wrapper for the component and hands the client a pointer to the wrapper.
- This wrapper, called a proxy, simply marshals methods calls and routes them across the network.
- On the other end, DCOM creates another wrapper, called a stub, which marshals methods calls and routes them to an instance of the component.

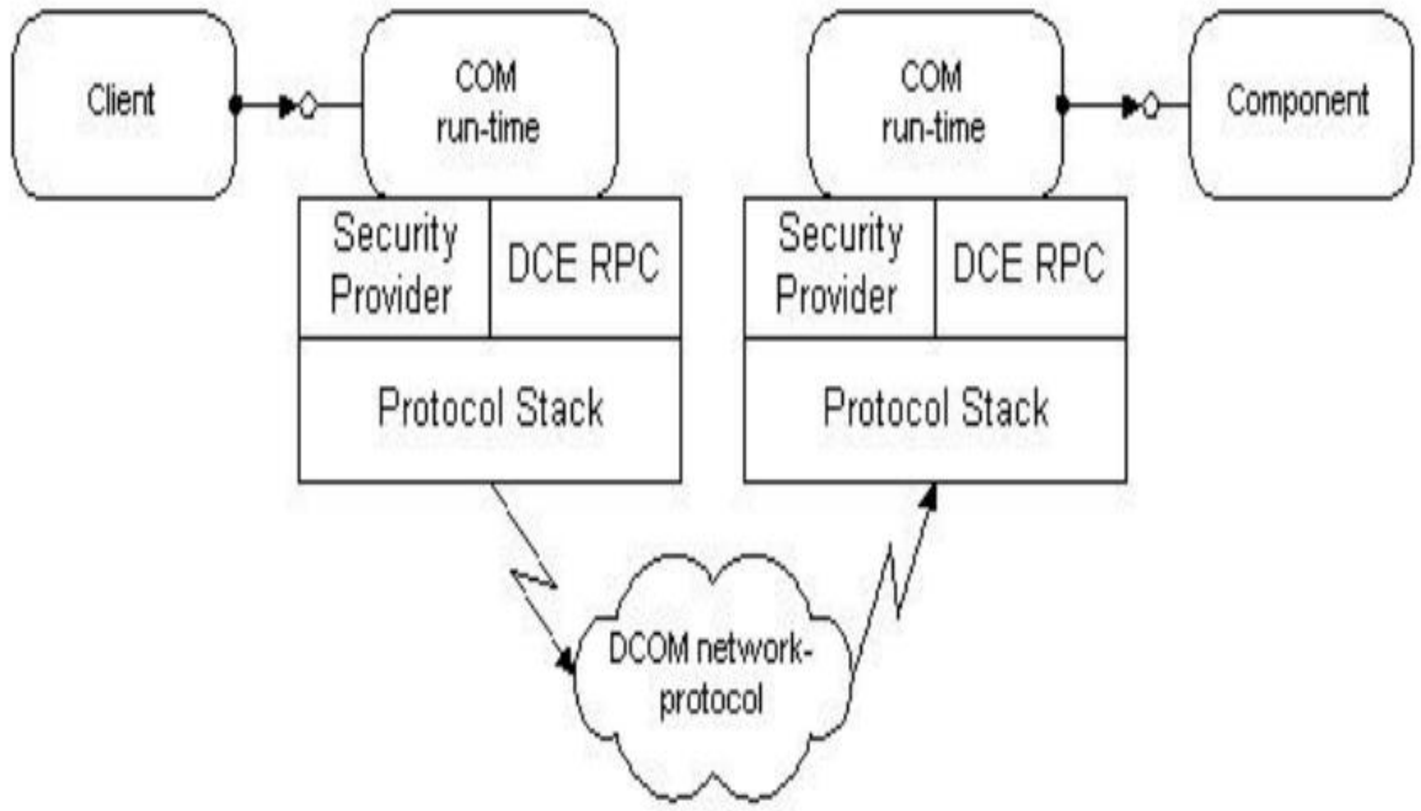
DCOM as DS technology Solution

- DCOM servers object can support multiple interfaces each representing a different behavior of the object.
- A DCOM client calls into the exposed methods of a DCOM server by acquiring a pointer to one of the server object's interfaces.
- The client object can the invoke the server object's exposed methods through the acquired interface pointer as if the server object resided in the client's address space.

DCOM as DS technology Solution

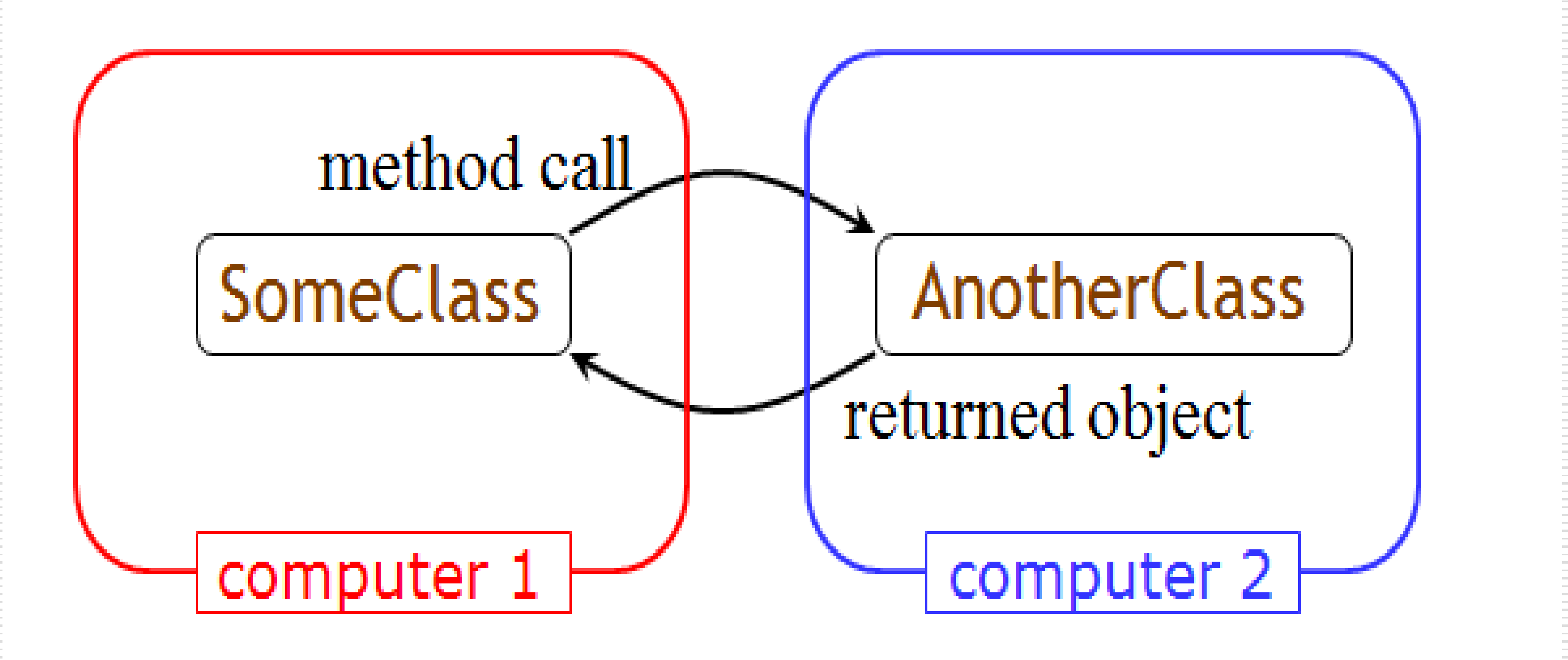
- All DCOM components and interfaces must inherit from IUnknown, the base DCOM interface.
- IUnknown consists of the methods AddRef(), Release() and QueryInterface().
- AddRef() and Release() are used to for reference counting and memory management.
- Essentially, when an object's reference count becomes zero, it must self-destruct.
- In COM, the request and responses are delivered via the *Lightweight Remote Procedure Calls (LRPC)*.

DCOM as DS technology Solution



RMI as Distributed technology solution

- Consider the following program organization:



RMI as Distributed technology solution

- Java RMI is a mechanism that allows one to invoke a method on an object that exists in another address space.
- The other address space could be on the same machine or a different one.
- The RMI mechanism is basically an object-oriented RPC mechanism.
- **Java/RMI** relies on a protocol called the **Java Remote Method Protocol (JRMP)**.

RMI as Distributed technology solution

- Java relies heavily on Java Object Serialization, which allows objects to be marshaled (or transmitted) as a stream.
- Since Java Object Serialization is specific to Java, both the Java/RMI server object and the client object have to be written in Java

RMI as Distributed technology solution

- Each Java/RMI Server object defines an interface, which can be used to access the server object outside of the current Java Virtual Machine (JVM) and on another machine's JVM.

- The interface exposes a set of methods, which are indicative of the services offered by the server object.
- For a client to locate a server object for the first time, RMI depends on a naming mechanism called an RMIRegistry that runs on the Server machine and holds information about available Server Objects.

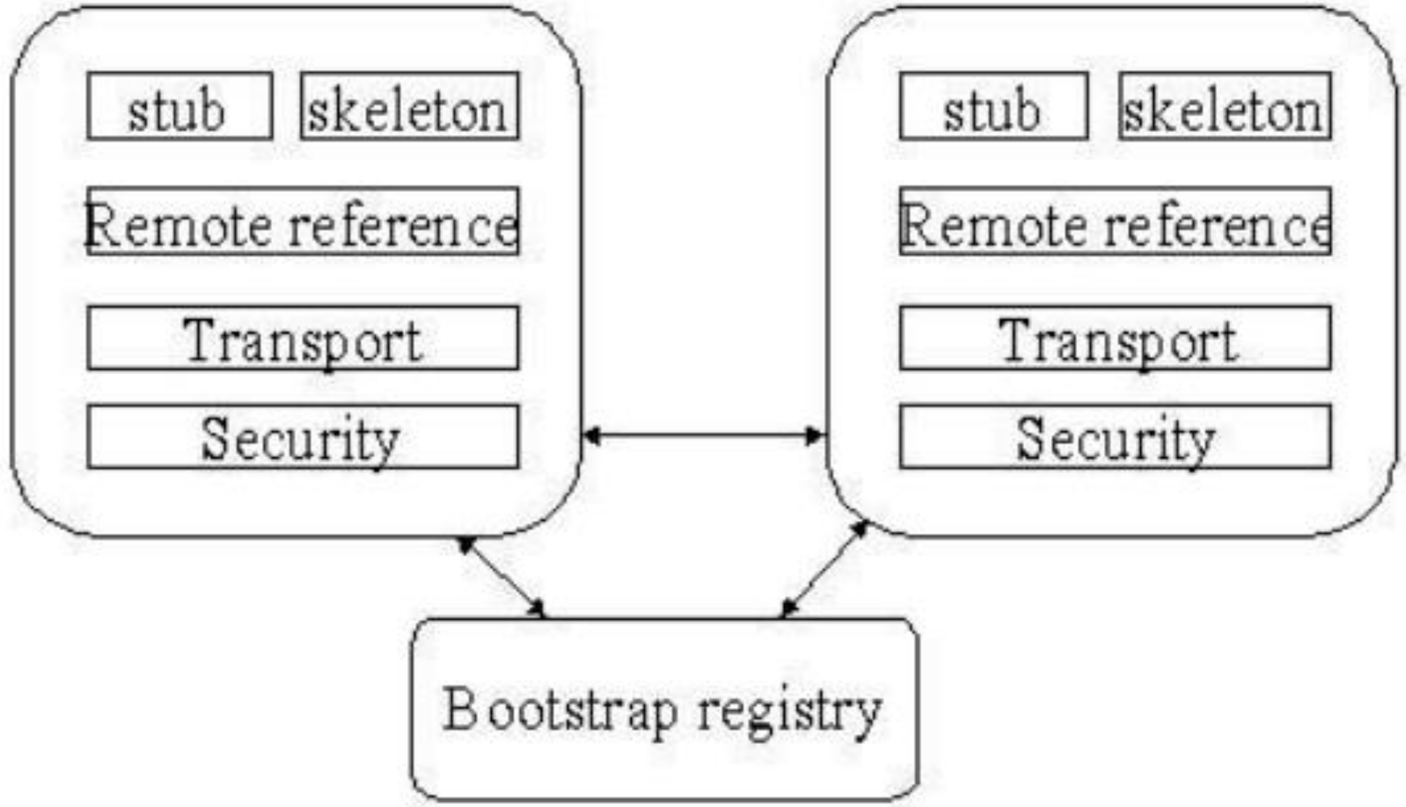
RMI as Distributed technology solution

- A Java/RMI client acquires an object reference to a Java/RMI server object by doing a lookup for a Server Object reference and invokes methods on the Server Object as if the Java/RMI server object resided in the client's address space.
- Java/RMI server objects are named using URLs and for a client to acquire a server object reference, it should specify the URL of the server object as you would with the URL to a HTML page.

RMI as Distributed technology solution

- It is composed of three parts:
  - The Stub/Skeleton Layer: The stub/skeleton layer provides the static client stubs and server skeletons.
  - The Remote Reference Layer: This layer handles the object references and management.
  - The Transport Layer: This layer is simply the Internet transport protocol used.
- Sun provides and implementation that uses TCP/IP; but someone else could replace it with UDP if he wishes.

RMI as Distributed technology solution



CORBA as Distributed technology solution

- CORBA is a software standard that is defined and maintained by the Object Management Group (OMG).
- CORBA is based on the Request-Response architecture.
- It consists of a standard framework for developing and maintaining **distributed software systems**.
- CORBA is to allow interoperability between objects on distributed systems.

CORBA as Distributed technology solution

- CORBA works by allowing clients and servers to communicate without worrying about the network protocol and other communication aspects.
- There is an object implementation on the server, which the client requests to execute.
- The client and the server object implementation do not have any restrictions on the address space,
- For example the client and the server can exist in the same address space, or can be located in separate address spaces on the same node, or can be located on separate nodes altogether.

CORBA as Distributed technology solution

CORBA objects are essentially object that supports the

- CORBA::
  - Object IDL interface and
  - The Remote references are called Object References.
- There is a specification of the CORBA IDL Language and how it is mapped with other languages.
- It essentially provides a means for the interface definitions.
- The Proxy or a local representative for the client side is called the IDL stub; the server-side proxy is the IDL skeleton.

CORBA as Distributed technology solution

The proxy represents an object created on the client side, which is used for more functionality like support

- for Dynamic invocation.
- For marshaling the request and the response, the information is delivered in a canonical format defined by the IIOP protocol used for CORBA interoperability on
- the Internet.

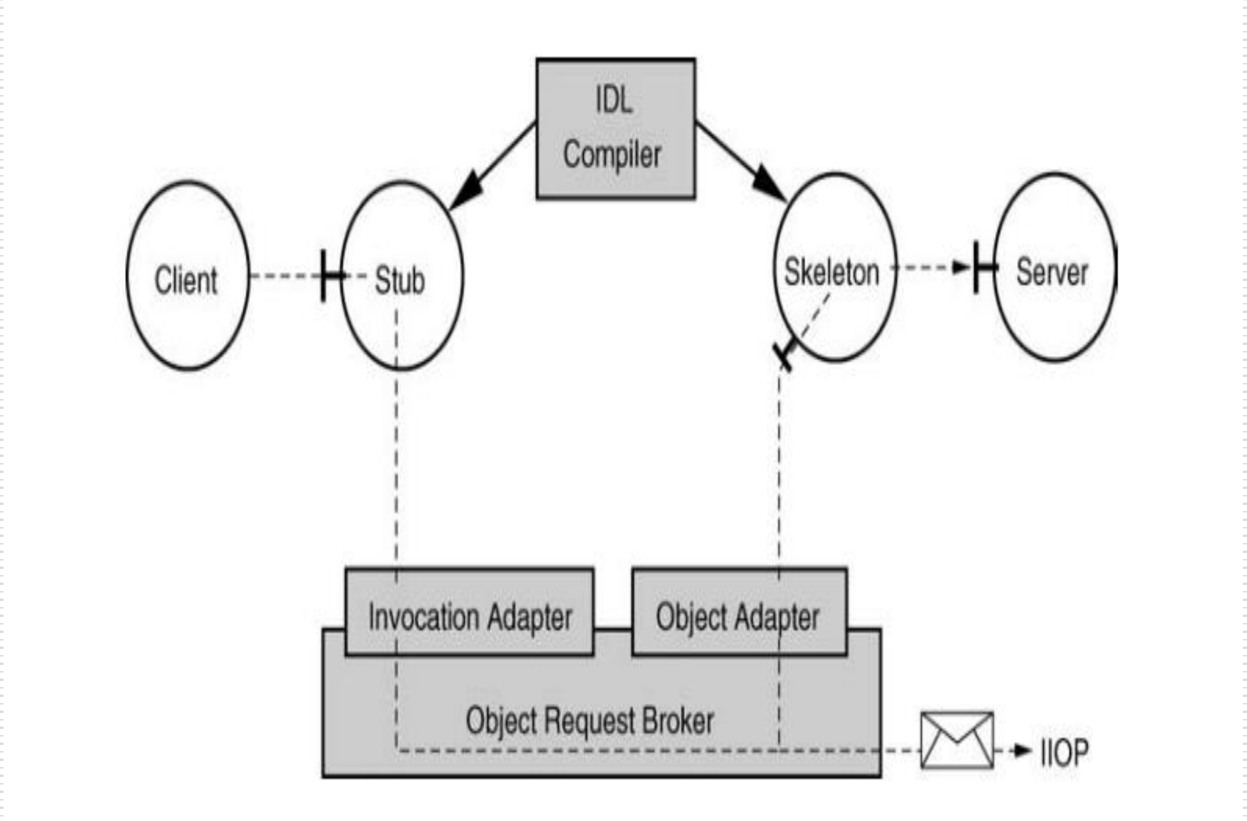
CORBA as Distributed technology solution

- IDL stub makes use of dynamic invocation interface for marshaling on the client side.
- Similarly on the server side, IDL Skeletons use the Dynamic Skeleton Interface for unmarshalling the information.
- The request (response) can also contain Object Reference as parameters; remote object can be passed by reference.

CORBA as Distributed technology solution

- The client does not need to know where the server is; the client can work just as if the object it is working with exists in the same process space.
- A CORBA implementation achieves this by creating client stubs and server skeletons; they marshal and unmarshal requests and responses.

CORBA as Distributed technology solution



Both server and client are processes executing on an OS. The server listens on a TCP socket, and the client opens a TCP connection to the remote server. The "language" spoken over this TCP connection is IIOP.

Object Request Broker

- The Object Request Broker (ORB) provides the "glue" between the client and server.
- The Object Request Broker (ORB) transmits operation invocations from a client to a server that can be located in the same address space, in different address spaces on the same computer, or on different computers.

Invocation and Object Adapters

- On the client side the invocation adapter enables an operation to be generated and invoked.
- In a similar way, the object adapter on the server side allows an invocation to be delivered to the object implementation.
- The task of the ORB is to accept operations at the invocation adapter and to forward and deliver them to the appropriate object adapter.

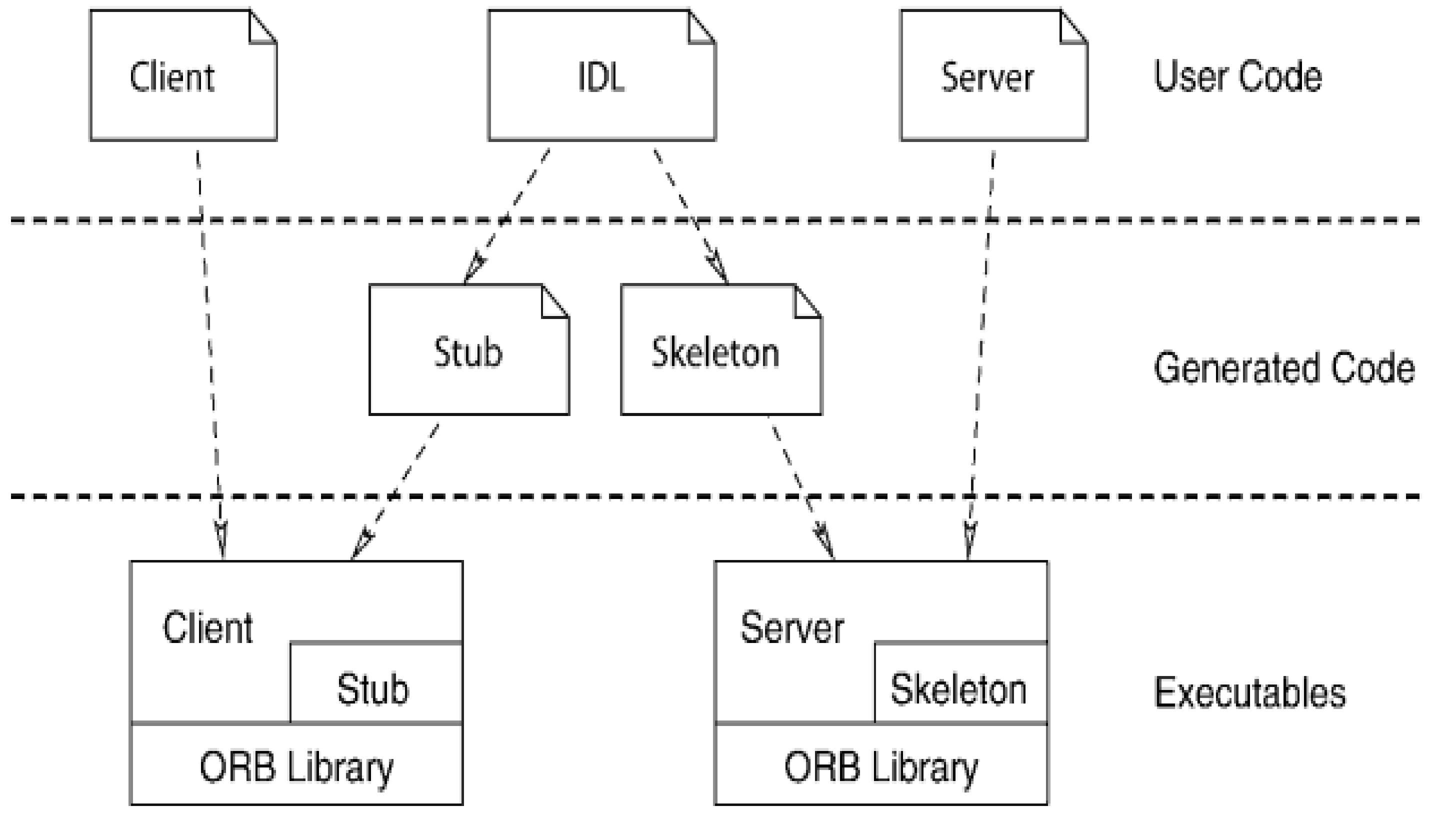
IDL

- In-order for the objects to make requests and receive response from other objects remotely of locally, Object request broker (ORB) is used, which is the object bus.
- Using this bus, the client is not aware of the mechanisms used to communicate with, activate, or store the server objects.
- There is predefined language called the Interface Definition Language, which achieves this.

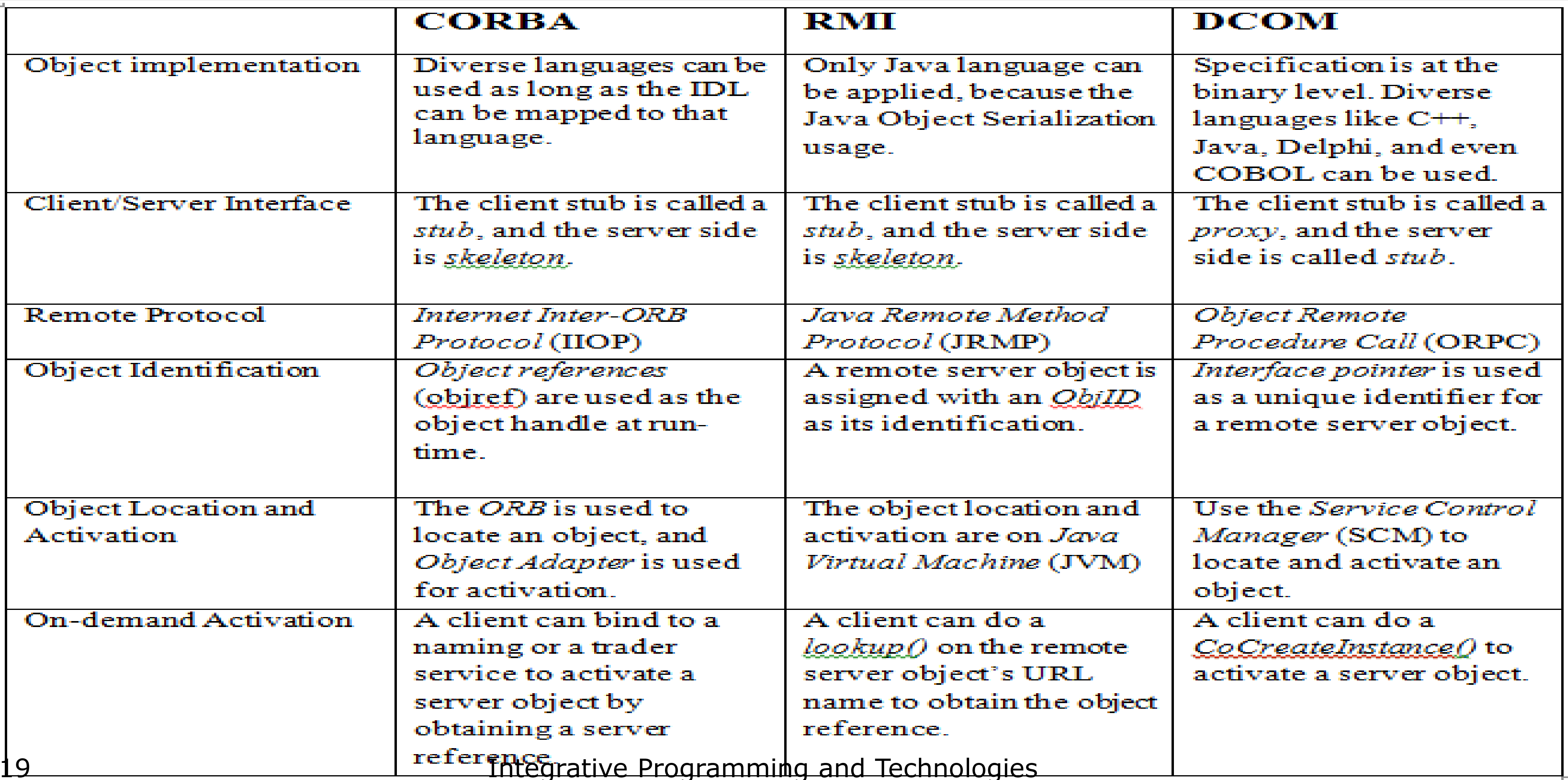
IDL

- The Interface Description Language (IDL) is a purely declarative language that specifies the interface that an object should implement and a client must use.
- In other word, it is used to specify object interfaces independently of a specific programming language.
- The IDL file is used to generate stub files for the client and server;
- It marshals parameters and sends them to the ORB.
- Thus, the client and server do not need to know location information; allowing the programmer to concentrate on the object implementation, not network handling.

CORBA as Distributed technology solution



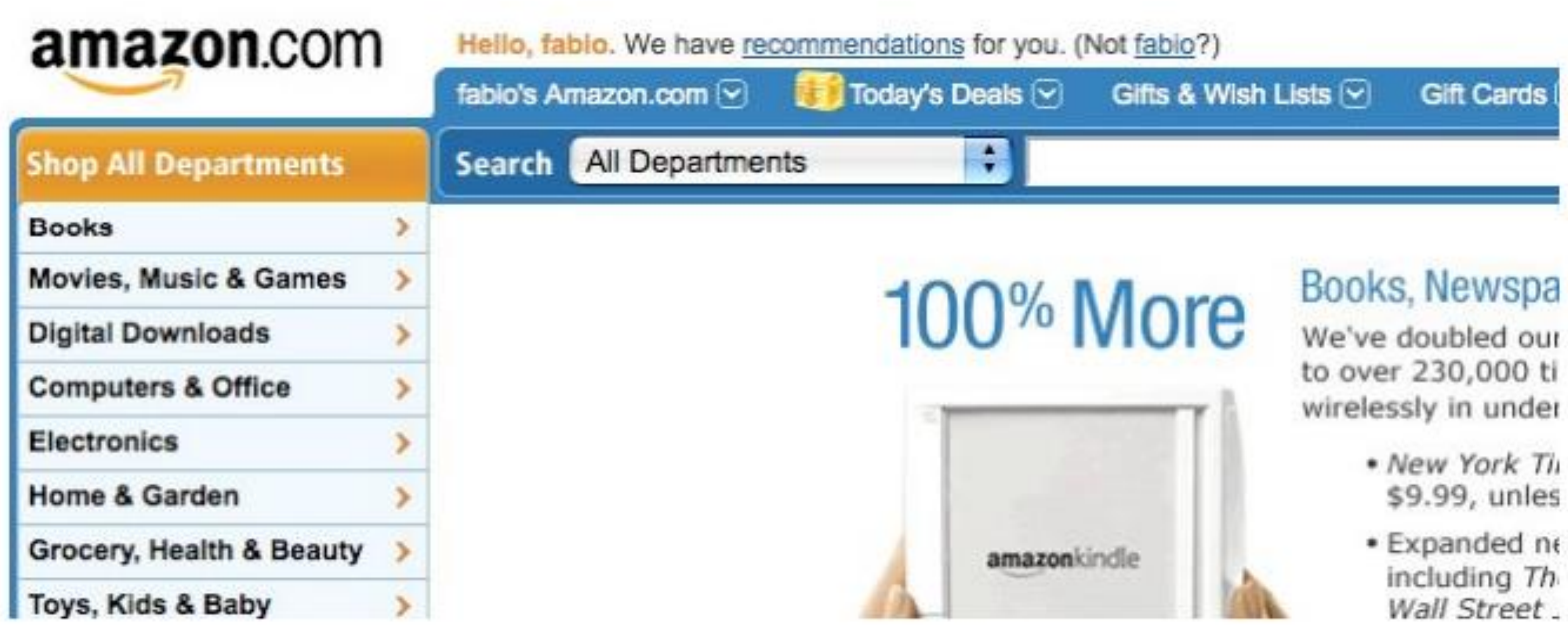
Summary



	CORBA	RMI	DCOM
Object implementation	Diverse languages can be used as long as the IDL can be mapped to that language.	Only Java language can be applied, because the Java Object Serialization usage.	Specification is at the binary level. Diverse languages like C++, Java, Delphi, and even COBOL can be used.
Client/Server Interface	The client stub is called a *stub*, and the server side is *skeleton*.	The client stub is called a *stub*, and the server side is *skeleton*.	The client stub is called a *proxy*, and the server side is called *stub*.
Remote Protocol	*Internet Inter-ORB Protocol* (IIOP)	*Java Remote Method Protocol* (JRMP)	*Object Remote Procedure Call* (ORPC)
Object Identification	*Object references* (objref) are used as the object handle at run-time.	A remote server object is assigned with an *ObjID* as its identification.	*Interface pointer* is used as a unique identifier for a remote server object.
Object Location and Activation	The *ORB* is used to locate an object, and *Object Adapter* is used for activation.	The object location and activation are on *Java Virtual Machine* (JVM)	Use the *Service Control Manager* (SCM) to locate and activate an object.
On-demand Activation	A client can bind to a naming or a trader service to activate a server object by obtaining a server reference.	A client can do a *lookup()* on the remote server object's URL name to obtain the object reference.	A client can do a *CoCreateInstance()* to activate a server object.

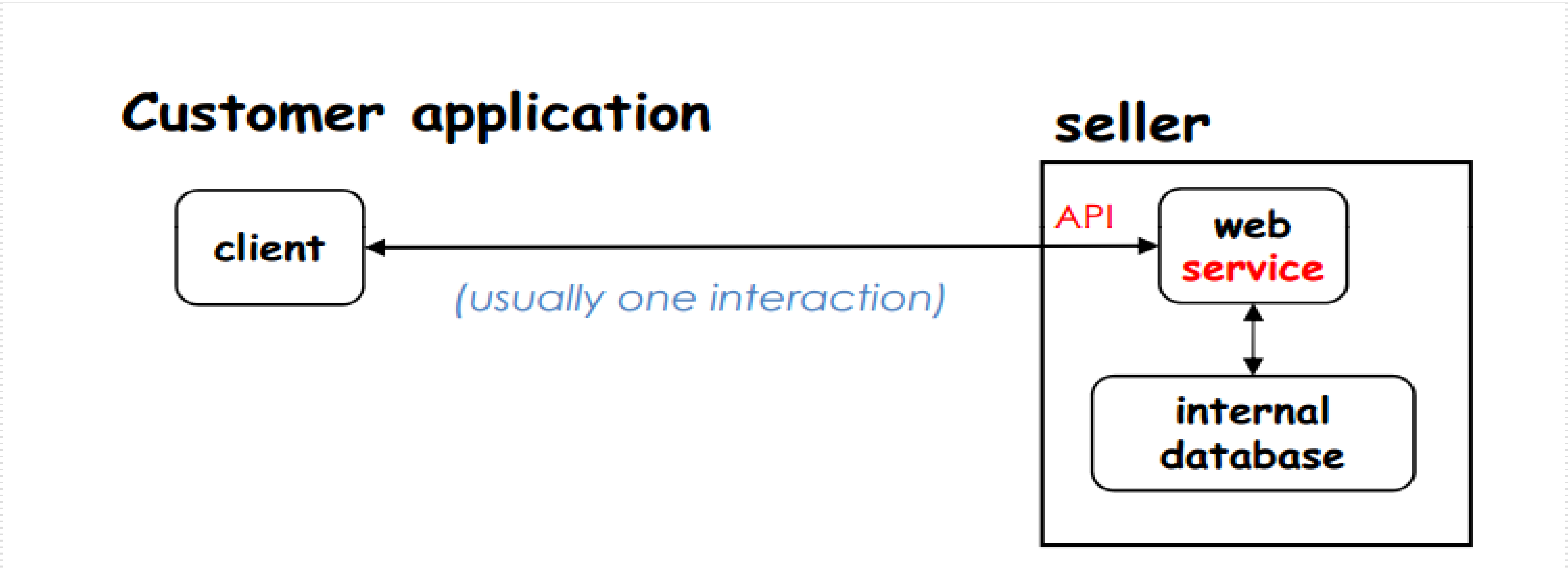
Web Services -Introduction

- Example - Amazon / Searching for books



Web Services -Introduction

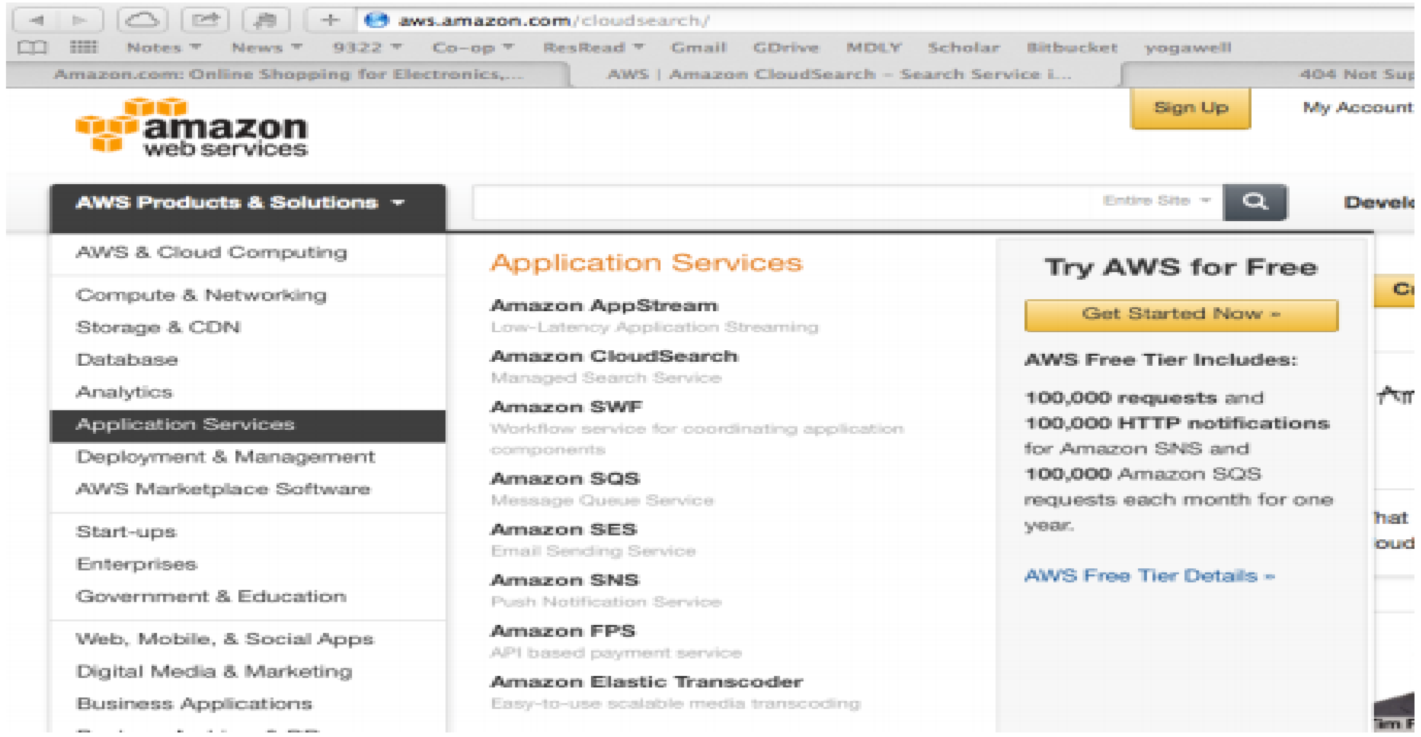
- Amazon Search as a Web Service



e.g., http://webservices.amazon.com/AWSECommerceService/AWSECommerceService.wsdl

Web Services -Introduction

- Amazon as a Web Service (App Integration)


aws.amazon.com/cloudsearch/

Notes News 9322 Co-op ResRead Gmail GDrive MDLY Scholar Bitbucket yogawell

Amazon.com: Online Shopping for Electronics,... AWS | Amazon CloudSearch – Search Service i... 404 Not Sup

Sign Up My Account

amazon web services

AWS Products & Solutions

Entire Site

AWS & Cloud Computing
Compute & Networking
Storage & CDN
Database
Analytics
Application Services
Deployment & Management
AWS Marketplace Software
Start-ups
Enterprises
Government & Education
Web, Mobile, & Social Apps
Digital Media & Marketing
Business Applications

Application Services

Amazon AppStream
Low-Latency Application Streaming

Amazon CloudSearch
Managed Search Service

Amazon SWF
Workflow service for coordinating application components

Amazon SQS
Message Queue Service

Amazon SES
Email Sending Service

Amazon SNS
Push Notification Service

Amazon FPS
API based payment service

Amazon Elastic Transcoder
Easy-to-use scalable media transcoding

Try AWS for Free

Get Started Now »

AWS Free Tier Includes:

100,000 requests and 100,000 HTTP notifications for Amazon SNS and 100,000 Amazon SQS requests each month for one year.

AWS Free Tier Details »

Web Services -Introduction

What is Web Service ?

- The Web is an immensely scalable information space filled with interconnected resources.
- A Web resource is any type of named information object— such as a word processing document, a digital picture, a Web page, an e-mail account, or an application—that’s accessible through the Web.

Web Services -Introduction

- All resources on the Web are connected via the Internet, and you access Web resources using standard Internet protocols.
- The Web is pervasive and provides universal connectivity.
- A service is an application that exposes its functionality through an application programming interface (API).
- In other words, a service is a resource that is designed to be consumed by software rather than by humans.

Web Services -Introduction

- Web service is an application that provides a **Web API**.
- An API supports application-to-application communication.
- A Web API is an API that lets the applications communicate using XML and the Web.
- Web services use the Web to perform application-to-application integration.

Web Services -Introduction

- Key concept: a Web service is a software component designed to support interoperable machine-to-machine interaction over a network.
  - Allow applications to share data and invoke capabilities from other applications
- Key facts:
  - No need to consider how the 'other' applications were built, what operating system or platform they run on
  - A standardized way of application-to-application communication based on XML open standards (ie., SOAP, WSDL and UDDI) over an Internet proto col backbone.

Unlike traditional client/server models, web services do not require

  - GUI, HTML or browser.

Web Services -Introduction

- Web service exhibits the following defining characteristics:
  - A Web service is a Web resource. You access a Web service using platform-independent and language-neutral Web protocols.
  - A Web service provides an interface—a Web API—that can be called from another program.
  - A Web service is typically registered and can be located through a Web service **registry**.
  - Web services support loosely coupled connections between systems.

Web Services -Introduction

Web services commercial frameworks

Microsoft

- (http://msdn.microsoft.com/en-us/library/ms950421.aspx)
- **IBM**

  (http://www.ibm.com/software/solutions/soa)
- **Oracle**

  (http://www.oracle.com/us/technologies/soa/index.html)
- **Hewlett Packard**

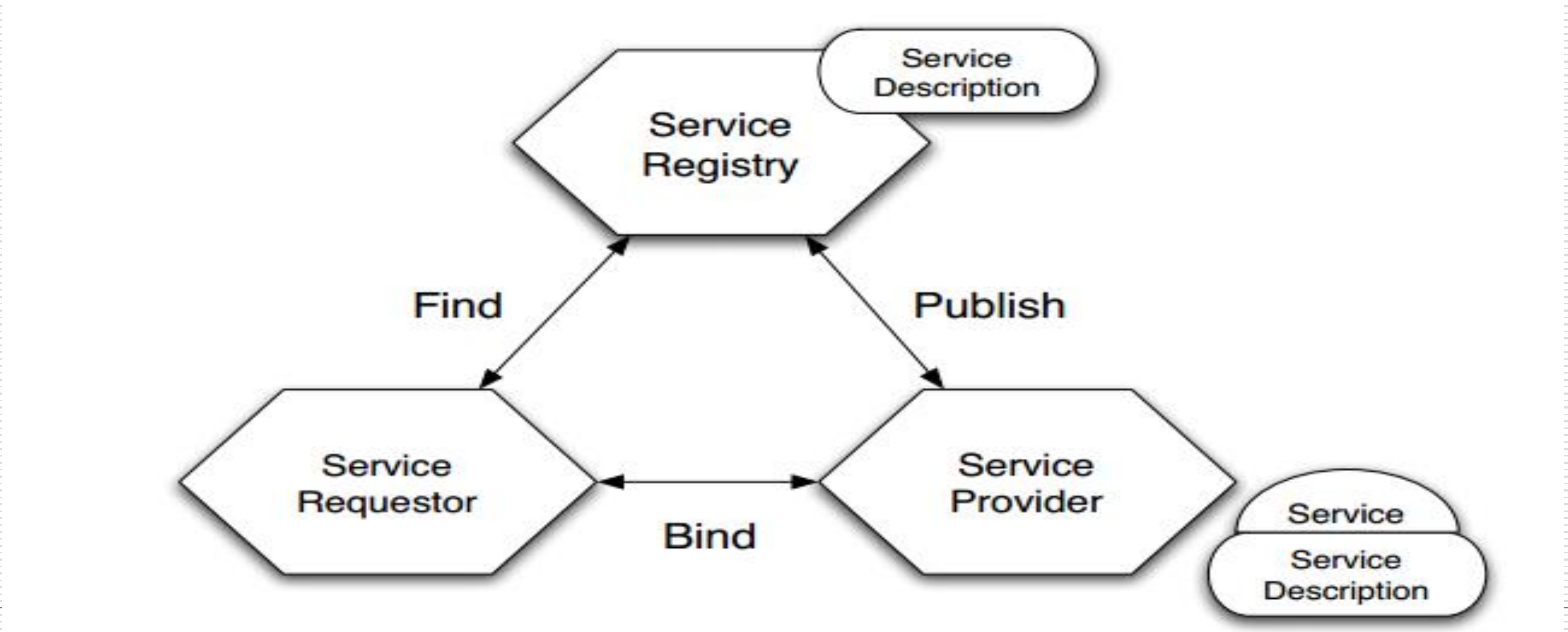
  (http://h71028.www7.hp.com/enterprise/w1/en/technologies/soa-overview.html)

and more ...

Web Services -Introduction

Web Services Conceptual Architecture (by IBM)

- Basic Web Services Architecture



Web Services -Introduction

Web Services Conceptual Architecture (by IBM)

- **Web Services Component**
  - Service
  - Service Description
- **Three Roles**:
  - **service provider:** develops an electronic service and registers its description at a publicly accessible service registry.
  - **service registry**: store/manage web services details
  - **service requestor**: query the registry to find an electronic service that meets his or her requirements.
  - A binding occurs between the service provider and the service requestor.

Why Web Services?

Integrate applications

- Web services communicate using XML and Web protocols, Which are pervasive, wor
  both internally and across the Internet, and support heterogeneous interoperability

Web service exhibits the following defining characteristics:

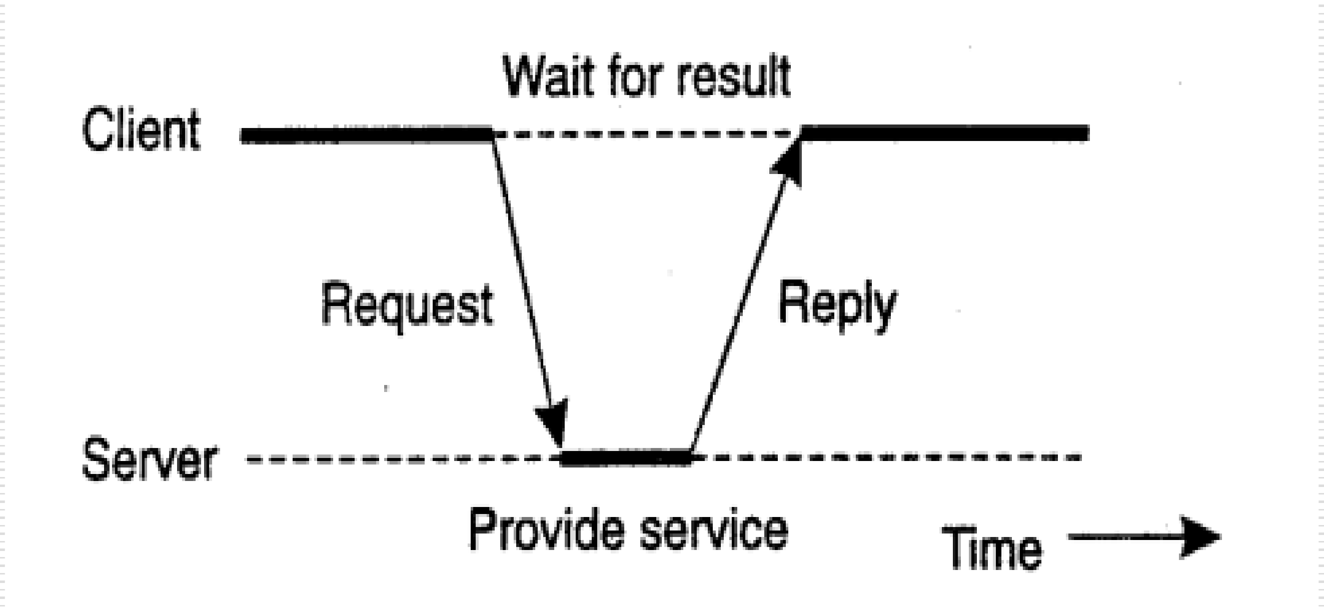
- ❑ A Web service is a Web resource. You access a Web service using platfor independent and language-neutral Web protocols.
- ❑ A Web service provides an interface Web API-that can be called from anothe program.
- ❑ A Web service is typically registered and can be located through a Web servic registry.

Web services support loosely coupled connections between systems.

Network Programming I
The Client-Server Communication Model

- At a basic level, network-based systems consist of a server , client , and a media for communication.
- A computer running a program that makes a request for services is called client machine.
- A computer running a program that offerrs requested services from one or more clients is called server machine.
- The media for communication can be wired or wireless network.
- This client-server interaction, also known as request-reply behavior.

Network Programming II
The Client-Server Communication Model



Generally, programs running on client machines make requests to a program (often called as server program) running on a server machine.

Network Programming III
The Client-Server Communication Model

- They involve networking services provided by the transport layer,which is part of the Internet software stack, often called TCP/IP (Transport Control Protocol/Internet Protocol) stack.
- The transport layer comprises two types of protocols, TCP (Transport Control Protocol) and UDP (User Datagram Protocol).
- The most widely used programming interfaces for these protocols are sockets.
- TCP is a connection-oriented protocol that provides a reliable flow of data between two computers.
- Example applications that use such services are HTTP, FTP, and Telnet.
- UDP is a protocol that sends independent packets of data, called datagrams , from one computer to another with no guarantees about arrival and sequencing.

Network Programming IV
The Client-Server Communication Model

- Example applications that use such services include Clock server and Ping.
- The TCP and UDP protocols use ports to map incoming data to a particular process running on a computer.
- Port is represented by a positive (16-bit) integer value.
- Some ports have been reserved to support common/well known services:

-FTP	21/tcp
-TELNET	23/tcp
-SMTP	25/tcp
-Login	513/tcp
-http	80/tcp,udp
-https	443/tcp,udp

- User-level process/services generally use port number value 1024.

Network Programming V
The Client-Server Communication Model

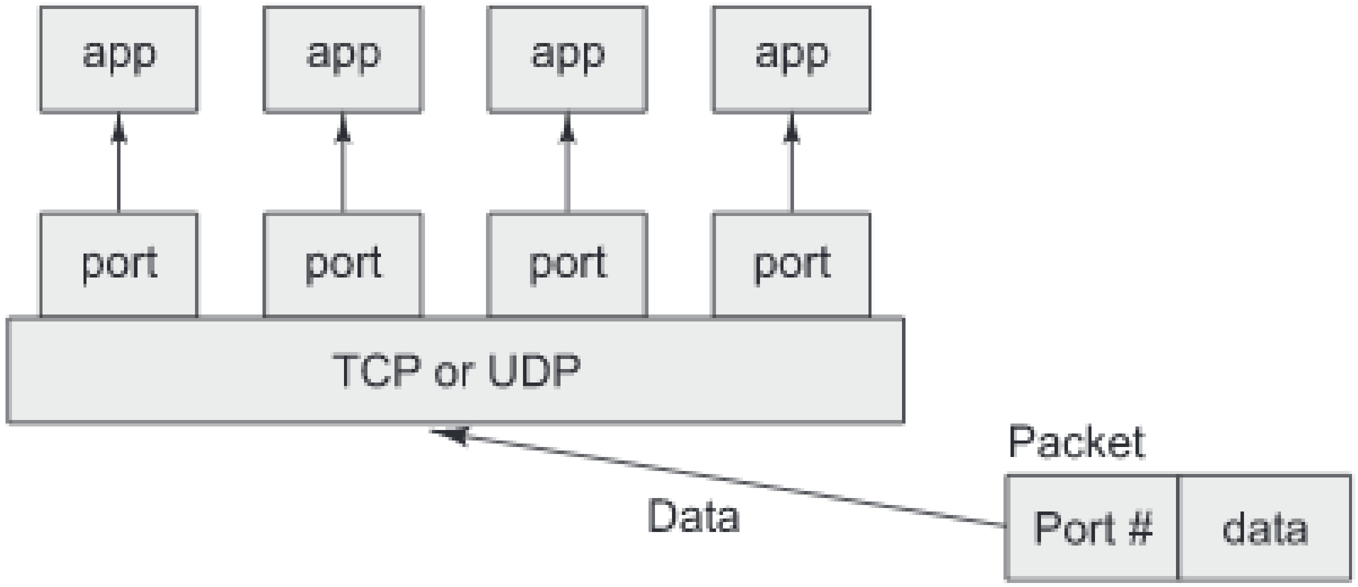


Figure: TCP/UDP mapping of incoming packets to appropriate port/process

Network Programming VI
The Client-Server Communication Model

Hosts Identification and Service Ports

- Every computer on the Internet is identified by a unique, 4-byte IP address.
- This is typically written in dotted quad format like 128.250.25.158 where each byte is an unsigned value between 0 and 255.
- This representation is clearly not user-friendly because it does not tell us anything about the content and then it is difficult to remember.
- Hence, IP addresses are mapped to names like www.buyya.com or www.google.com, which are easier to remember.
- Internet supports name servers that translate these names to IP addresses.

Network Programming VII
The Client-Server Communication Model

- In general, each computer only has one Internet address.
- However, computers often need to communicate and provide more than one type of service or to talk to multiple hosts/computers at a time.
- For example, there may be multiple ftp sessions, web connections, and chat programs all running at the same time.
- To distinguish these services, a concept of ports, a logical access point, represented by a 16-bit integer number is used.
- That means, each service ordered by a computer is uniquely identified by a port number.
- Each Internet packet contains both the destination host address and the port number on that host to which the message/request has to be delivered.

Network Programming VII
The Client-Server Communication Model

- The host computer dispatches the packets it receives to programs by looking at the port numbers specified within the packets.
- That is, IP address can be thought of as a house address when a letter is sent via post snail mail and port number as the name of a specific individual to whom the letter has to be delivered.

Network Programming VIII
The Client-Server Communication Model

Sockets and Socket-based Communication

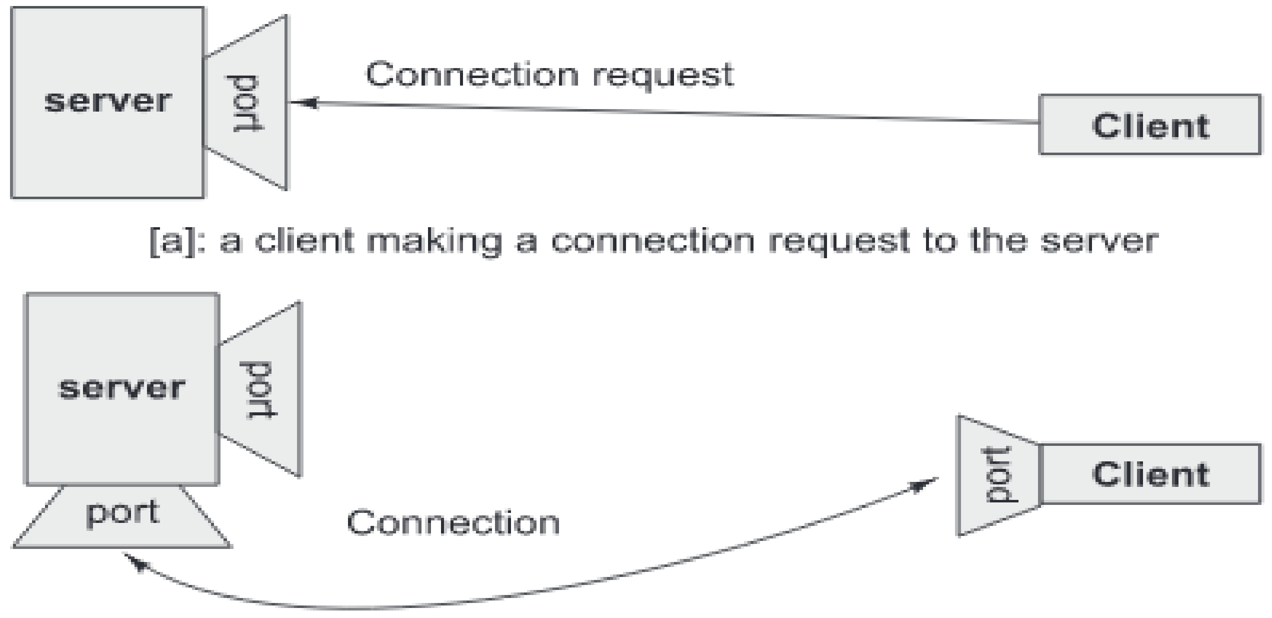
- Sockets provide an interface for programming networks at the transport layer.
- Network communication using Sockets is very much similar to performing file I/O.
- In fact, socket handle is treated like file handle.
- The streams used in file I/O operation are also applicable to socket-based I/O.
- Socket-based communication is independent of a programming language used for implementing it.
- That means, a socket program written in Java language can communicate to a program written in non Java (say C or C++) socket program.

Network Programming IX
The Client-Server Communication Model

- A server (program) runs on a specific computer and has a socket that is bound to a specific port.
- The server listens to the socket for a client to make a connection request.
- If everything goes well, the server accepts the connection.
- Upon acceptance, the server gets a new socket bound to a different port.
- It needs a new socket (consequently a different port number) so that it can continue to listen to the original socket for connection requests while serving the connected client.

Network Programming X
The Client-Server Communication Model

Socket Programming



[a]: a client making a connection request to the server

[b]: session established with temporary ports used for two way communication.

Figure: Establishment of path for two-way communication between a client and server

Message and queuing services I

- A message is information sent by a sender process to a receiver process.
- A message queue is a mechanism that allows a sender process and a receiver process to exchange messages;
- The sender posts a message in the queue, and the receiver retrieves the message from the queue.
- The senders and receivers of messages may communicate in a synchronous way or in an asynchronous way.
- With a synchronous communication protocol, a receiver waits for a message from a sender, i.e., it blocks until the message arrives.
- Whereas with an asynchronous communication protocol, the receiver continues executing and is notified of the reception of a message when this one arrives.

Message and queuing services II

A message queuing system provides several facilities:

- creating messages, creating queues, initializing
- sender and receiver processes,
- providing a means to send and receive messages.

Several message queuing systems are proposed.

Message and queuing services III

Some are proprietary and others are open source.

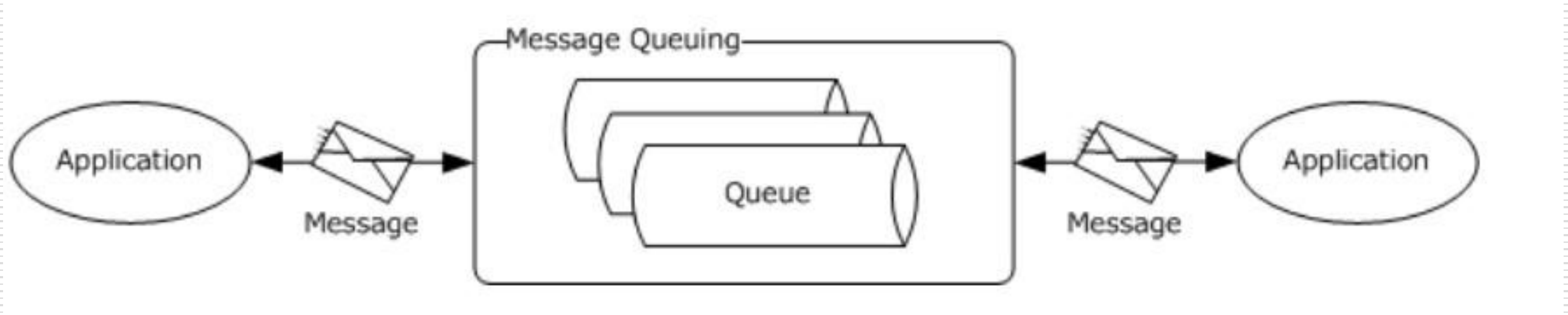
- Oracle proposes Advanced Queuing for Oracle databases,
- Skype has Skytools PgQ for PostgreSQL a database
- IBM provides WebSphere MQ
- Microsoft has MSMQ
- Sun Microsystems defines Java Message Service (JMS) as a specification of a Java standard for message queuing systems

- Open source message queuing systems include ActiveMQ, Jboss Messaging, and JORAM

Message and queuing services IV

- It is a communications service that temporally decouples message sends operations from message receive operations.
- The functionality enables applications to communicate even if those applications are not executed concurrently.
- Applications send messages to a queue and/or receive messages from a queue.
- The queue provides persistence of the messages, enabling them to survive across application restarts.
- As such, this abstraction enables an application to send message even if the receiving application is not executing or is unreachable due to a network outage.

Message and queuing services V



Message Queuing enables the following message exchange patterns between applications:

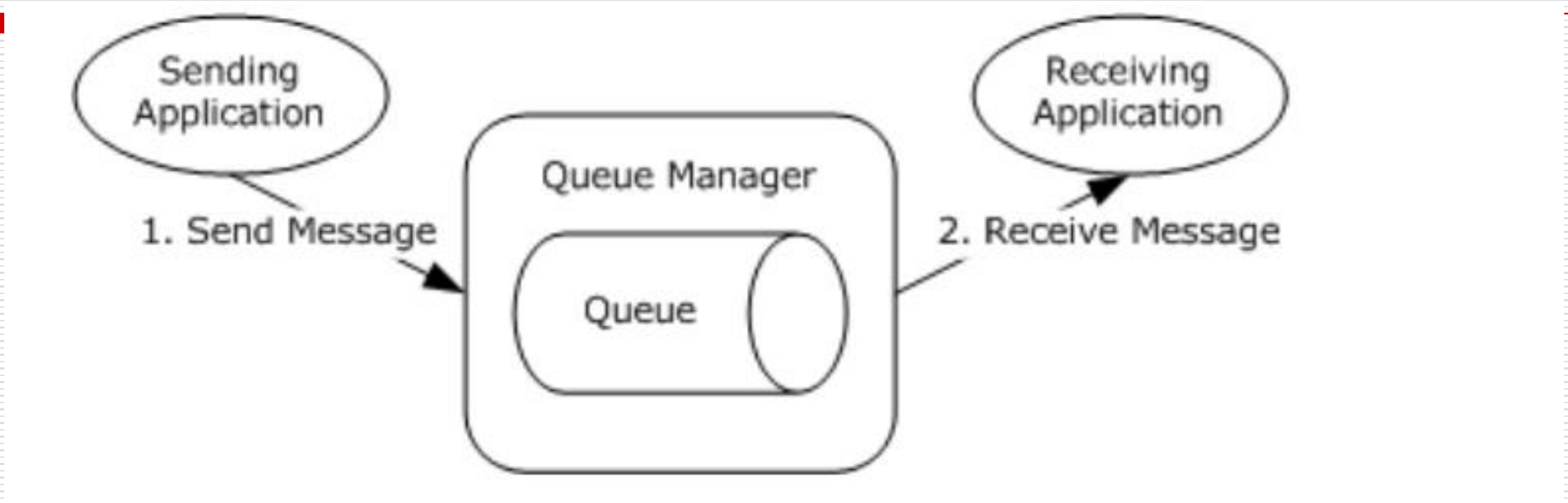
One-Way Messaging:

A source application sends messages to a destination application and does not wait for the outcome of the message processing.

Message and queuing services VI

- The receiving application receives the request message and sends the response message to a queue specified by the sender in the request message.
- The sending application receives the response message and correlates it to the original request message.
- **Broadcast**:
- A source application sends messages that can be received by zero or more applications.
- This pattern is useful in implementing publish-and subscribe types of applications.
- The simplest Message Queuing deployment involves two applications and a single queue that is accessible to both the applications.
- The queue is hosted and managed by a single queue manager.
- One application sends messages to the queue, and the other application receives the messages from the same queue.

Message and queuing services VII



The sending application sends a message to the queue.
When the send operation is successful, the application proceeds with other work, or terminates(1).
The receiving application subsequently receives the message asynchronously (2).
The message is removed from the queue.

Message and queuing services VIII

MSMQ - Queue Manager

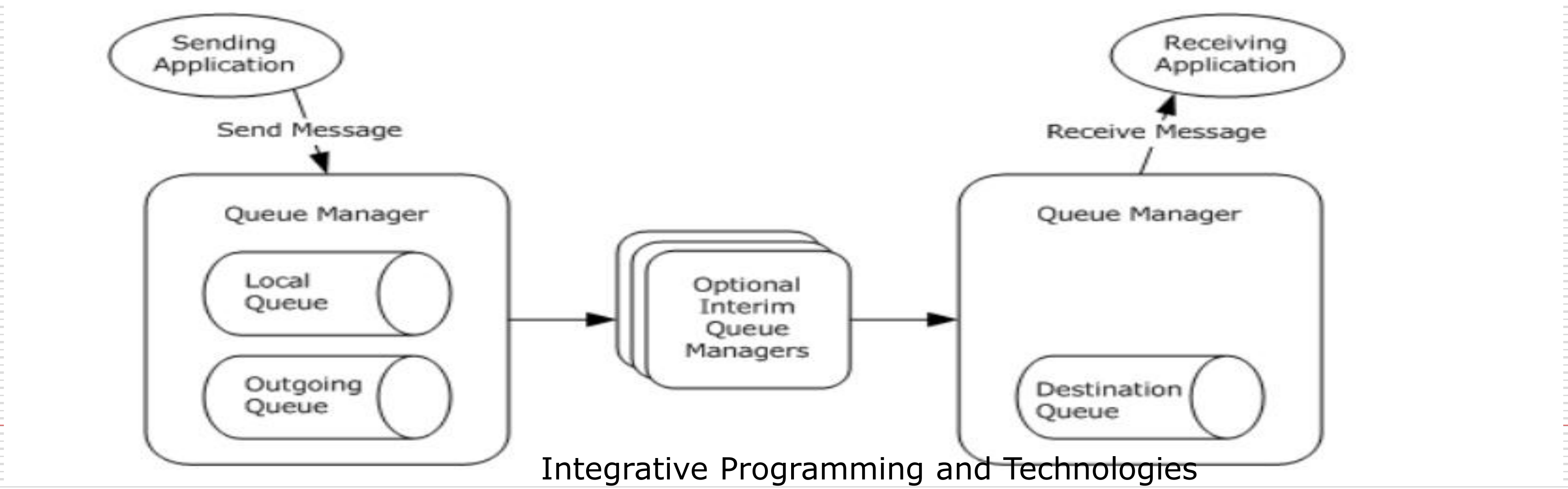
- Queues are hosted and managed by a queue manager that plays the queue server role.
- The queue manager hosts and manages a set of local queues, acts as an intermediary placeholder for storing and forwarding messages to their final destinations, and interacts with the applications for sending and receiving messages.
- The queue manager performs the following tasks:
- On the send side, the queue manager manages its queues, accepts messages from the sending application, and optionally transfers messages to other queue managers.
- If the messages are destined for a queue that the send-side queue manager hosts, the messages are placed in that queue on the machine.

Message and queuing services IX

- Alternatively, if the messages belong to a queue that is not hosted by the queue manager on the send side, the messages are placed in an outgoing queue and subsequently transferred to the destination queue manager.
- On the receive side, the queue manager manages its queues, accepts messages transferred from other queue managers, and delivers messages to the receiving application.
- Optionally, there can be other queue managers between the send and the receive queue managers.
- This approach facilitates efficient message routing between the source and the destination queues.
- These interim queue managers store incoming messages and route them to the next hop so that they can eventually reach the final destination queue.
- In other words: A sending application sends a message to a nearby queue manager.
- If the destination queue is hosted by the queue manager (a local queue), the queue manager stores the message in the local queue.

Message and queuing services X

- Alternatively, if the destination queue is hosted by another queue manager on a different machine, the queue manager places the message in an outgoing queue.
- In either case, the sending application can proceed to do other work.
- The queue manager asynchronously transfers the message from the outgoing queue to the queue manager of the destination queue, optionally through interim queue managers for routing the message.
- Subsequently, a receiving application reads the message from the destination queue.



Reading Assignment

- List commonly used low level data communications protocols
- State conditions for when each protocol should be used
- Outline the protocol for one low level communications protocol