Important Instructions:

1) Open this MS-Word document and start writing answers below each respective question given on page 2.
2) Answers the question in the same sequence in which they appear.
3) Provide to the point and concrete answers. Some of the questions are open ended and therefore must be answered using your own opinion and thoughts but backed with logical reasons.
4) First read the questions and understand what is required of you before writing the answer.
5) Attempt the paper yourself and do not copy from your friends or the Internet. Students with exactly similar answers or copy paste from the Internet will not get any marks for their assignment.
6) You can contact me for help if you have any doubt in the above instructions or the assignment questions.
7) All questions must be attempted.
8) Do not forget to write your name, university ID, class and section information.
9) Rename you answer file with your university ID# before uploading to SIC.
10) When you are finished with writing your answers and are ready to submit your answer, convert it to PDF and upload it to SIC unzipped, before the deadline mentioned on SIC.

Spring Semester 2020 Final Exam
Course: - Distributed Computing

Deadline: - Mentioned on SIC **Marks: - 50**

Program: - MS (CS) **Dated: 24 June 2020**

Student Name: Iman **Student ID#: 13523**

Class and Section: MS(Cs)

Section: Remote Invocation

Q1. Describe briefly the purpose of the three communication primitives in request-reply protocols. (6)

Q2. Explain the technical difference between RPC and RMI? (4)

Section: Indirect Communication

Q:3 In contrast to Direct Communication, which two important properties are present in Indirect Communication? (6)

Q:4 Provide three reasons as why group communication (single multicast operation) is more efficient than individual unicast operation? (9)

Section: OS Support

Q5. Differentiate a between a network OS and distributed OS. (6)

Q6. Describe briefly how the OS supports middleware in a distributed system by providing and managing (6)

a) Process and threads
b) System Virtualization

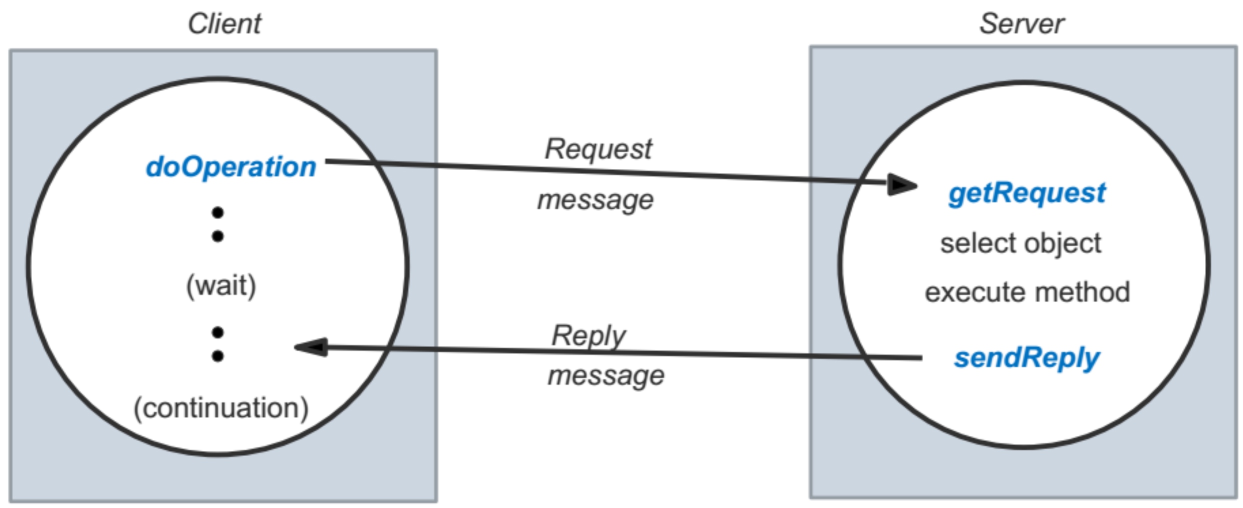
Section: Distributed Objects and Components

Q7. Write in your own words the issues with Object (distributed) oriented middlewares. (13)

Q1. Describe briefly the purpose of the three communication primitives in request-reply protocols?

Answer;

The request-reply protocol represents the model above message transmission and supports the two-way message exchange encountered in client-server computing.
The request-reply protocol we describe here is based on three types of communication primitives, doOperation, getRequest, and sendReply.



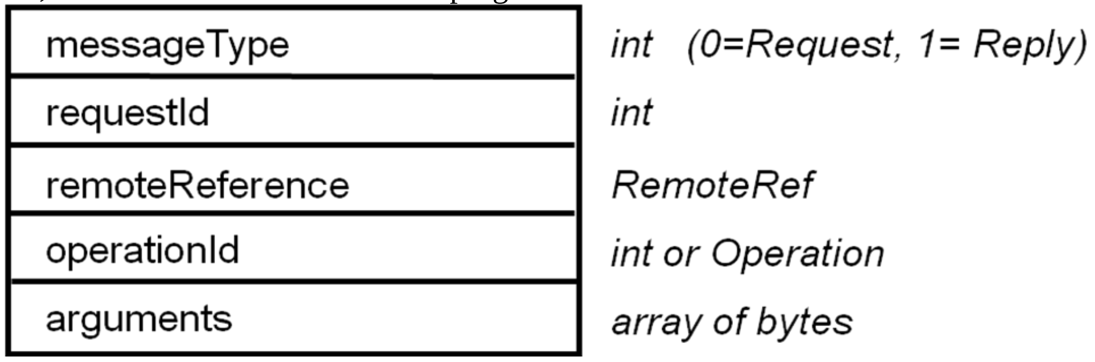
The request-reply protocol matches the request with the response. It can be designed to provide a certain delivery guarantee. If UDP datagrams are used, delivery guarantee must be provided by the request-reply protocol, which can use server response messages as confirmation of client request messages.

public byte[] doOperation (RemoteRef s, int operationId, byte[] arguments)
sends a request message to the remote server and returns the reply.
The arguments specify the remote server, the operation to be invoked and the arguments of that operation.

public byte[] getRequest ();
acquires a client request via the server port.

public void sendReply (byte[] reply, InetAddress clientHost, int clientPort);
sends the reply message reply to the client at its Internet address and port.

The client uses the **doOperation** method to call the remote operation. Its arguments specify the remote server and the operation to be invoked, as well as other information (arguments) required for the operation. The client that calls doOperation collects the arguments into a byte array and unmasks the results of the returned byte array. The first arguments of doOperation is an instance of the RemoteRef class, which represents a reference to the remote server. The doOperation method sends a request message to the server whose Internet address and port are specified in the remote reference given in the arguments. After sending the request message, doOperation calls to receive the response message, and then extracts the result from it and returns it to the caller. The caller of doOperation will be blocked until the server performs the requested operation and sends the response message to the client process. The server process uses getRequest to get the service request. After the server calls the specified operation, it will use sendReply to send the response message to the client. When the client receives the response message, the original doOperation is released, and the execution of the client program continues.



messageType	*int (0=Request, 1= Reply)*
requestId	*int*
remoteReference	*RemoteRef*
operationId	*int or Operation*
arguments	*array of bytes*

Message identifiers; Any message processing scheme requires that each message has a unique message identifier, through which it can be referenced. Any architecture that involves message processing requires a unique message identifier to provide the following attributes:
Reliable delivery
Request-reply communication
Parts of a message identifier; RequestID, which comes from the increasing
The whole process is determined by the sending process.
The identifier of the sending process, such as its port and Internet address.
Failure model of the request-reply protocol; If three primitives doOperation, getRequest and sendRhness are implemented on the UDP datagram, they will suffer the same communication failure. There is no guarantee that messages will be delivered in the order of the sender. In addition, the protocol may suffer process failures. In order to solve the situation where the server crashes or abandons the request or response message, doOperation uses a timeout while waiting to obtain the response message from the server.
Timeouts; After expiration, doOperation can perform different choices. The simplest option is to immediately return doOperation failure. To compensate for the possibility of missing messages, doOperation will repeatedly send request messages until a response is obtained or it is reasonably determined that the delay is caused by the server's lack of response (rather than response). Lost message. Finally, when doOperation returns, it will indicate to the client that no exception has been received.
Discarding duplicate request messages; In the case of resending the request message, the server can receive it multiple times. For example, the server may receive the first request message, but it takes longer than the client's timeout to complete the command and return a response.
Lost reply messages; If the server has sent a response when it receives a repeated request, it will have to perform the operation again to obtain the result unless the result of the original execution is stored. Some servers can run their operations multiple times and get the same results each time. The operations of the server are idempotent, and no special measures are required to avoid performing their operations multiple times.
Styles of exchange protocols;

- The Request (R) protocol.
- The Request-Reply (RR) protocol;
- The Request-Reply-Acknowledge reply (RRA) protocol

In the R protocol, the client sends a single request message to the server. The R protocol can be used when there is no value to return from the remote operation and the client does not need any request to confirm that the operation has been performed.

Q2. Explain the technical difference between RPC and RMI?

Remote procedure call (RPC)

The idea of RPC is to call remotely implemented code, just like we are just calling functions. Remote Procedure Call (RPC) is a programming language function designed to perform distributed computing based on the semantics of local procedure calls.

Remote Method invocation (RMI)

RMI is a possible way to access business objects distributed from another Java virtual machine. RMI uses the serialization of objects to marshal and request parameters.

If you want to send an object through a thread, your class (object) must implement a serializable interface. RMI can adopt a natural, direct and fully optimized method to provide distributed enterprise computing technology, which enables us to add Java functionality to the entire system. Remote method invocation (RMI) is similar to RPC, but it is language-specific and is a function of Java.

Remote procedure call (RPC)	**Remote Method invocation (RMI)**
RPC stands for remote procedure call that supports procedure programming.	RMI stands for remote method invocation, similar to PRC, but it supports Java's object-oriented programming.
The process passes methods/functions.	Remote objects can be accessed by reference.
The proxy server participates in the process call.	Implement object-to-object implementation between different Java objects to implement a distributed communication model.
Call remote procedures, such as calling methods.	RMI passes the object as a parameter to the remote method.
Deletion is not completely transparent to the client.	RMI calls remote methods from the object.
RPC can be used to call functions through proxy functions.	RMI can be used to call object methods.
RPC supports process programming.	RMI supports object-oriented programming.
RPC is an old version of RMI.	Although this is a subsequent version of RPC.
Simple code in RPC requires some code.	Although a few codes are not necessary for simple applications in RMI.
There is a huge version control problem in RPC.	Although there is a version using RDMI.
Programming with RPC is very easy.	Although programming in RMI is not easy.

The only difference between RMI and RPC is the case where the RPC function is called through a proxy function, and for RMI, we call the method through a proxy object.

RPC and RMI are similar, but the fundamental difference between RPC and RMI is that RPC supports process programming; on the other hand, RMI supports object-oriented programming.

Q:3 In contrast to Direct Communication, which two important properties are present in Indirect Communication?

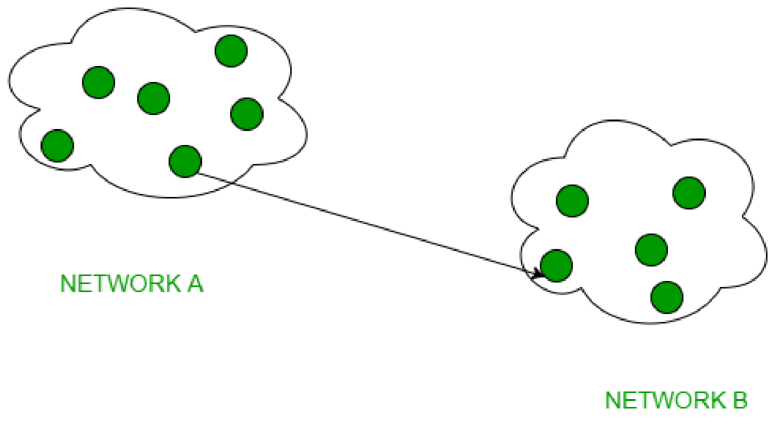
- **Key properties of indirect communication**
- **Space and time uncoupling**

Indirect communication is defined as communication between entities of a distributed system through an intermediary, without direct coupling between the sender and receiver.

Key properties
Space uncoupling; The sender does not know or need to know the identity of the receiver, and vice versa. There is no need to know the identity of the recipient and vice versa. Participants can be replaced, updated, copied or migrated.
Time uncoupling; The sender and receiver can have independent lifetimes.
Independent service life Or Independent Lifetimes.
Indirect communication is often used in distributed systems where changes are expected.
Examples; A mobile environment where users can quickly connect to or disconnect from the network. Management of event flow in the financial system.
Send and receive messages from mailboxes (sometimes called ports). The mailbox has a unique identifier(id). If two processes share a mailbox, they can communicate. A process can communicate with multiple processes. Sending and receiving are shown below.
Send (A, message): Send a message to mailbox A.
Receive (A, message): Receive a message from mailbox A.

Q:4 Provide three reasons as why group communication (single multicast operation) is more efficient than individual unicast operation?
The term conversion here means that some data (packet stream) is transmitted to the client receiver through the communication channel, which helps them communicate.

1. Unicast; This type of information transmission is useful when a single sender and a single receiver are involved. So, in short, you can call it one-to-one transmission. For example, if a device with an IP address of 10.1.2.0 in one network wants to send traffic (data packets) to a device with an IP address of 20.12.4.2 in another network, unicast becomes a picture. It is the most common form of data transmission over the network.

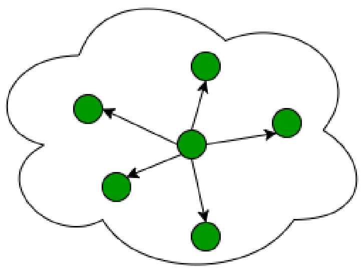


UNICAST EXAMPLE

2. Broadcast

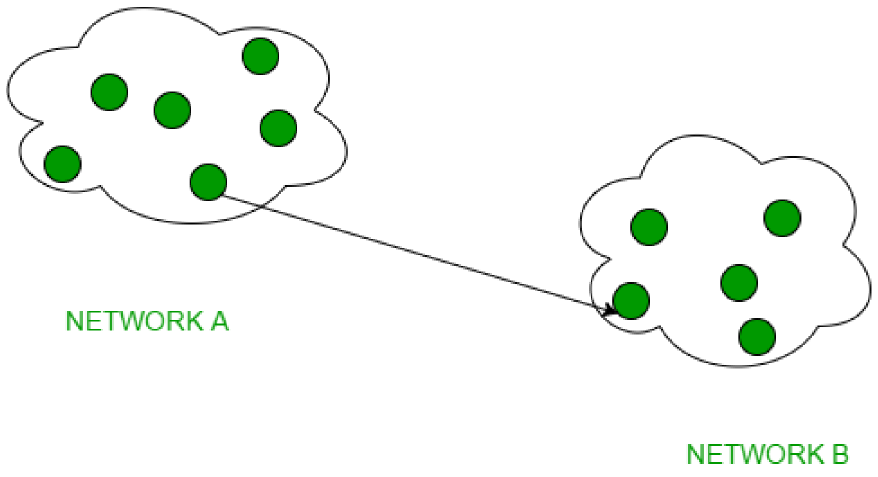
Broadcast transmission technology (one-to-one) can be divided into two types:

Limited Broadcasting; Suppose you have to send a stream of packets to all the devices on the network where you reside, this broadcast is very convenient. To this end, it will add 255.255.255.255 (every 32 bits of the IP address is defined as 1) to the destination address of the data header (packet), called the limited broadcast address, which is reserved for the transmission of a single client on the network (Sender) information for all recipients.



NETWORK CLUSTER

Direct Broadcasting; This is useful when devices on one network want to stream packets to all devices on another network. This is achieved by converting all bits in the host ID portion of the target address to 1, called the direct broadcast address in the data header, for information transmission.



UNICAST EXAMPLE

Television networks mainly use this mode for video and audio distribution. In computer networks, an important protocol of this type is the Address Resolution Protocol (ARP), which is used to resolve IP addresses into physical addresses required for basic communications.

3. Multicast; In multicast, one or more senders and one or more receivers participate in data transmission traffic. In this method, traffic is skewed between unicast (one-to-one) and broadcast (one-to-one) limits. Multicasting allows servers to directly copy data streams and then emulate and route them to the host requesting them. IP multicast requires the support of some other protocols, such as IGMP (Internet Group Management Protocol) for multicast routing to work. Also in class IP addressing, class D is reserved for multicast groups.

Q5. Differentiate a between a network OS and distributed OS.

The main difference between network operating systems and distributed operating systems is that network operating systems provide network-related functions, while distributed operating systems connect multiple independent computers through the network to perform tasks similar to one computer. The operating system acts as an interface between the user and the hardware. It monitors program execution and undertakes various tasks. It performs file management, device management, memory management, protects data and resources, monitors system performance, processor management, and more. Therefore, the operating system is an important part of the computer system. There are different types of operating systems. Network operating system and distributed operating system are two. The network operating system consists of software and related protocols that allow a group of computer networks to be used together. A distributed operating system is an ordinary centralized operating system, but it can run on multiple independent processors. You can access remote resources by connecting to the desired remote computer or transferring data from the remote computer to the user's own computer. Users access remote resources in the same way as local resources. The main difference between these two operating systems (network operating system and distributed operating system)
In addition, in order to distribute the operating system, each The node or system has the same operating system as the network operating system.

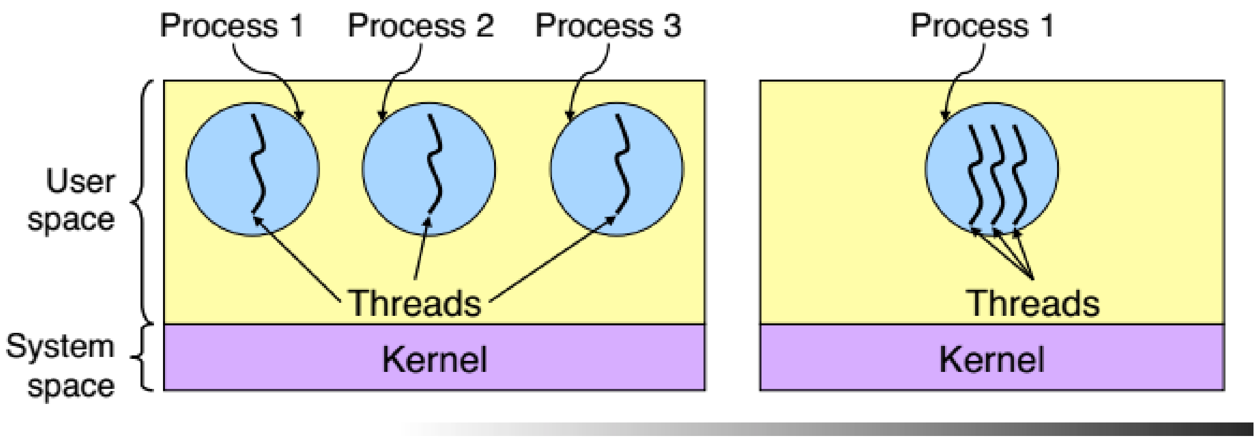
The main goal of the network operating system is to provide local services to remote clients.	The main purpose of a distributed operating system is to manage hardware resources.
In network operating systems, communication is based on files.	In distributed operating systems, communication is based on messages and shared memory.
Network operating systems are more scalable than distributed operating systems.	The scalability of distributed operating systems is lower than network operating systems.
In a network operating system, all nodes can have different operating systems.	In a distributed operating system, all nodes have the same operating system.

Networks and distributed operating systems have a common hardware foundation, but the difference is in software.

Q6. Describe briefly how the OS supports middleware in a distributed system by providing and managing

a) **Process and threads**
b) **System Virtualization**

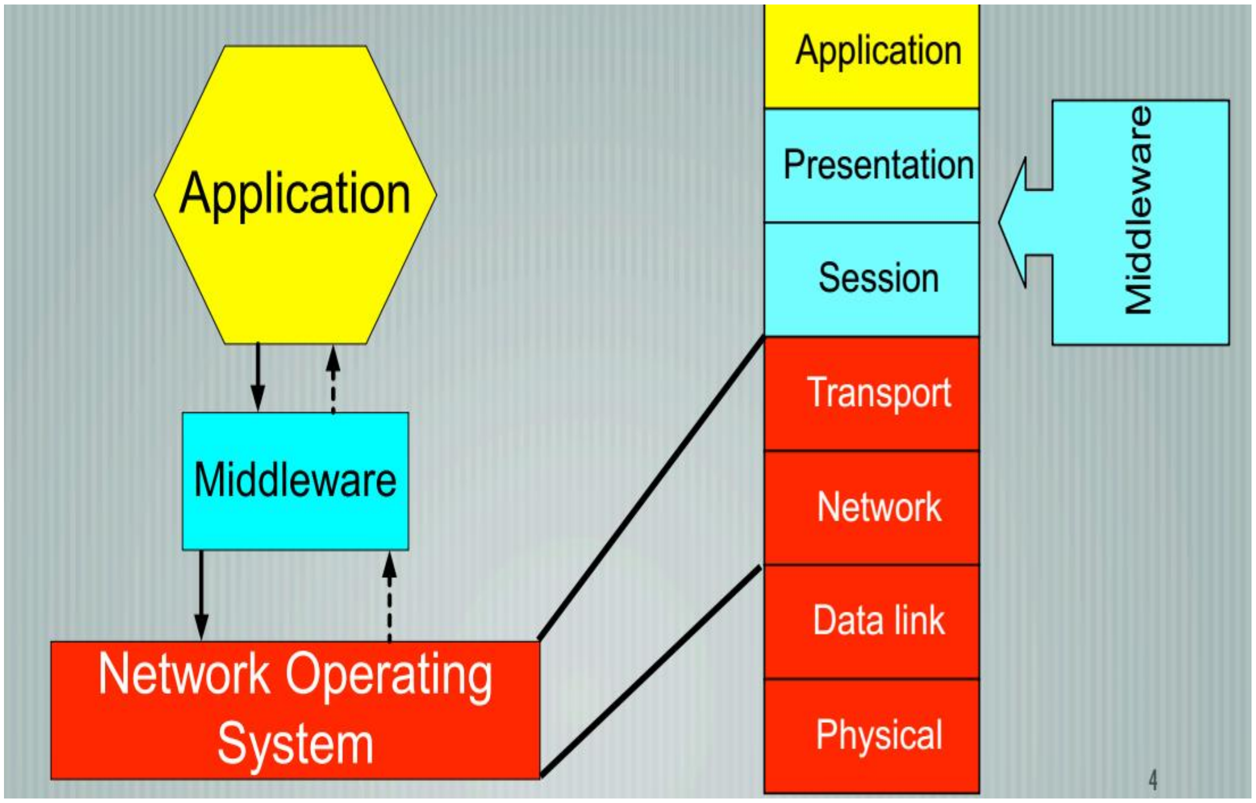
Process and Threads; The main task of the operating system is to run and manage processes. A program is a specification that contains definitions of data types, how to access data, and a set of instructions that perform useful "work" when executed. Sequential processes are activities produced by sequential processors executing programs and their data. The procedure is passive, while the process is active.
Process and Process Management; In a multi-program and time-sharing environment, process management is required. A program may require several processes. Many processes can be instances of the same program. Many processes from different programs can run simultaneously.
Threads; User and kernel-level threads
Process =Address Space
Thread =Program Counter/Stream of Instructions
Two examples



Three processes, each with one thread
One process with three threads
System Virtualization
Virtualization is the creation of a virtual rather than a real version of something (such as an operating system (OS), server, storage device, or network resource).
Virtualization uses software that simulates hardware functions to create virtual systems. This approach allows IT departments to run multiple operating systems, multiple virtual systems, and various applications on the same server. The benefits of virtualization include higher efficiency and economies of scale. Operating system virtualization is the use of software that allows one hardware to run multiple operating system images simultaneously. The technology was first used on mainframes for decades, allowing administrators to avoid wasting expensive processing power.

Q7. Write in your own words the issues with Object (distributed) oriented middlewares.
Middleware
Middleware simplifies the construction of distributed systems by implementing the presentation layer and the session layer. It is very difficult to build a distributed system directly on the

transport layer. Communication usually must send complex parameters. Different encodings of data types in memory. Parameters and return values may refer to another distributed system component. Developers and administrators should implement activation components. Character safety is an important concern. The manual implementation of type safety is subject to error. The implementation of synchronization is very tedious and error-prone. Middleware simplifies the construction of distributed systems by implementing the presentation layer and the session layer.



Transaction-Oriented Middleware

Transaction-oriented middleware is usually associated with distributed database applications. Object-oriented middleware has the capability of transaction-oriented middleware.

Message-Oriented Middleware

Message-Oriented Middleware is used when reliable, asynchronous communication is the dominant form of distributed system interaction.

When reliable asynchronous communication is the main form of distributed system interaction, message-oriented middleware will be used.

Remote Procedure Calls (RPCs)

Remote procedure call (RPC) is a procedure call that crosses the boundaries of the host. It was invented by Sun Microsystems as part of its Open Network Computing Platform (ONC). The origin of object-oriented middleware is RPC.

RPC-Interface Definition Language

The server component that runs RPC is called an RPC program. RPC is divided into RPC programs. The RPC program has an interface definition, which defines a process that can be called remotely. The interface definition also defines what can be passed to the remote program.

DCE (Distributed Computing Environment) includes a definition language interface for defining these exports independently of the programming language.

RPC – Presentation Layer

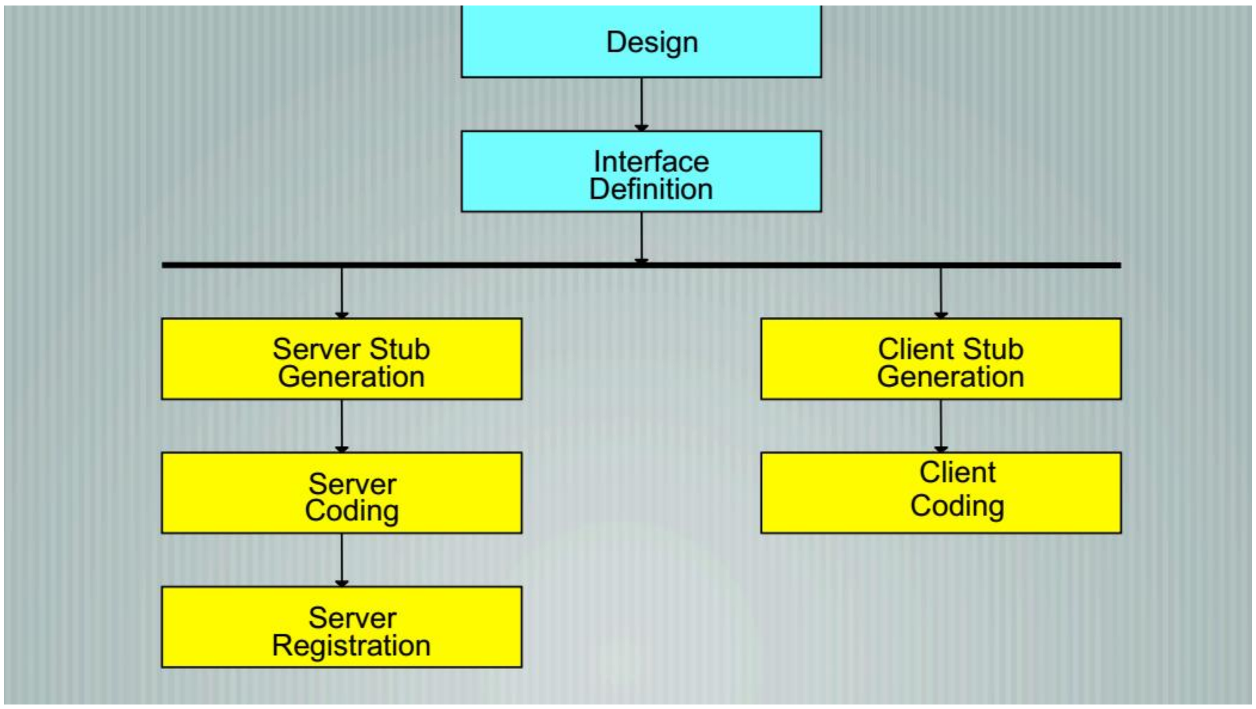
The integration of the RPC presentation layer maps the application data structure to a homogeneous and transferable form. Object-oriented middleware also supports client and server stubs. The presentation layer of object-oriented middleware must map object references to transmission formats (marshalling and unmarshalling).

Session Layer Implementation

The implementation of the conversation layer of object-oriented middleware is more complicated than RPC. The OO middleware session layer must map object references to the host. Server object addressing is based on object references. Object references are generated by middleware. The implementation of the session layer of object-oriented middleware maps object references to the host in a completely transparent manner.

Developing with Object-Oriented Middleware

Design and Implementation Process



Design server objects according to existing requirements, and future client objects may have. Use CASE tools that support object-oriented notation (UML).

- Class diagrams
- Sequence diagrams
- State diagrams