|  |  |  |  |
| --- | --- | --- | --- |
| **Assignment No. 01** | | **Date** |  |
| **Title** | **Remote Method Invocation** | | |

**PROBLEM DEFINITION**

Use the RMI / RPC concept to define the operations on one machine and use those on remotely placed machine’s application.

**HW/SW REQUIREMENTS**

|  |  |  |
| --- | --- | --- |
| **Software** | **Language Used** | **Software Required** |
|  |  |
|  |  |

**THEORY**

**Introduction**

**The RMI in JAVA**

*Distributed systems require that computations running in different address spaces, potentially on different hosts, be able to communicate*. For a basic communication mechanism, the Java programming language supports sockets, which are flexible and sufficient for general communication. However, sockets require the client and server to engage in applications-level protocols to encode and decode messages for exchange, and the design of such protocols is cumbersome and can be error-prone.

**A Simple Client/Server Application using RMI**

In a simple Client Server the *server receives a request from a client, processes it, and returns a result*. In the following Example, the request specifies the two numbers. The server adds these together and returns the sum.

**Steps for Implementing Calculator Using RMI**

**Step I**

1. An interface is a method which contains abstract methods; these methods must be implemented by another class. All remote interfaces must extend the Remote interface, which is part java.rmi. Remote defines no members. Its purpose is simply to indicate that an interface uses remote methods. All remote methods can throw a RemoteException.

2. CalculatorServerImpl.java, implements the remote interface. All remote objects must extend UnicastRemoteObject, which provides functionality that is needed to make objects available from remote machines.

3. CalculatorServer.java, contains the main program for the server machine. Its primary function is to update the RMI registry on that machine. This is done by using the rebind( ) method of the Naming class (found in java.rmi). That method associates a name with an object reference. The first argument to the rebind( ) method is a string that names the server as “AddServer”. Its second argument is a reference to an instance of CalculatorServerImpl.

4. CalculatorClient.java, implements the client side of this distributed application.

**Step II**

To generate stubs and skeletons, you use a tool called the RMI compiler, which is invoked from the command line, as shown here:

rmic CalculatorImpl

**Step III**

Install Files on the Client and Server Machines

Copy CalculatorClient.class, CalculatorImpl\_Stub.class, and CalculatorServerIntf.class to a directory on the client machine. Copy CalculatorServerIntf.class, CalculatorServerImpl.class, CalculatorServerImpl\_Skel.class, CalculatorServerImpl\_Stub.class, and CalculatorServer.class to a directory on the server machine.

**Step IV**

Start the RMI Registry on the Server Machine The Java 2 SDK provides a program called rmiregistry, which executes on the server machine. It maps names to object references. First, check that the CLASSPATH environment variable includes the directory in which your files are located. Then, start the RMI Registry from the command line, as shown here:

#### start rmiregistry

When this command returns, you should see that a new window has been created. You need to leave this window open until you are done experimenting with the RMI example.

**Step V**

Start the Server

The server code is started from the command line, as shown here:

java calculatorServer

Recall that the CalculatorServer code instantiates CalculatorServerImpl and registers that object with the name “CalculatorServer”.

**Step VI**

Start the Client with two arguments

java calculatorClient 5 7

**QUESTIONS & ANSWERS**

**1. What is RPC, LRPC, and RMI?**

**2. How stub is generated in RPC?**

**4. Explain call semantics in RPC and RMI invocations.**

**5. How applications are developed in RMI?**

**6. List the advantages and disadvantages of RMI.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Assignment No. 02** | | **Date** |  |
| **Title** | **CORBA** | | |

**PROBLEM DEFINITION**

Develop Distributed System Application using CORBA to demonstrate object brakering.

**HW/SW REQUIREMENTS**

|  |  |  |
| --- | --- | --- |
| **Software** | **Language Used** | **Software Required** |
|  |  |
|  |  |

**THEORY**

The Common Object Request Broker Architecture (CORBA) is a standard defined by the Object Management Group (OMG) that enables software components written in multiple computer languages and running on multiple computers to work together.

CORBA is a standard for distributing objects across networks so that operations on those objects can be called remotely. CORBA is not associated with a particular programming language, and any language with a CORBA binding can be used to call and implement CORBA objects. Objects are described in a syntax called Interface Definition Language (IDL).

CORBA includes four components:

**Object Request Broker (ORB)**

The Object Request Broker (ORB) handles the communication, marshaling, and unmarshaling of parameters so that the parameter handling is transparent for a CORBA server and client applications.

**CORBA server**

The CORBA server creates CORBA objects and initializes them with an ORB. The server places references to the CORBA objects inside a naming service so that clients can access them.

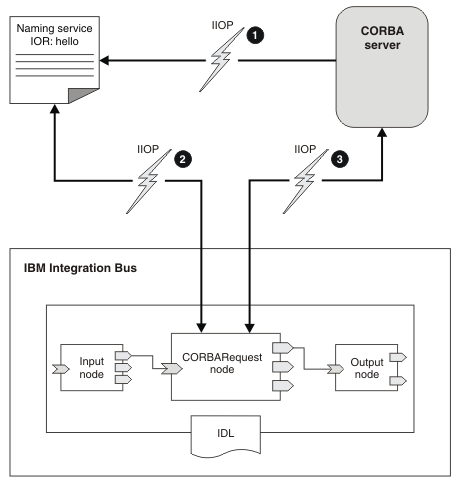
**Naming service**

The naming service holds references to CORBA objects.

**CORBARequest node**

The CORBARequest node acts as a CORBA client.

The following diagram shows the layers of communication between IBM® Integration Bus and CORBA.



The diagram illustrates the following steps.

1. CORBA server applications create CORBA objects and put object references in a naming service so that clients can call them.
2. At deployment time, the node contacts a naming service to get an object reference.
3. When a message arrives, the node uses the object reference to call an operation on an object in the CORBA server.

**Steps to run the application:**

idlj -fall Calculator.idl

javac \*.java CalculatorApp/\*.java

orbd -ORBInitialPort 1050

java CalcServer -ORBInitialPort 1050 -ORBInitialHost localhost

java CalcClient -ORBInitialPort 1050 -ORBInitialHost localhost

**QUESTIONS & ANSWERS**

1. **How CORBA is useful for interaction with other objects? Write in details.**
2. **What are advantage of CORBA?**
3. **What parts does CORBA use?**
4. **What is the role of CORBA in Distributed processing?**

|  |  |  |  |
| --- | --- | --- | --- |
| **Assignment No. 03** | | **Date** |  |
| **Title** | **MPI Processor** | | |

**PROBLEM DEFINITION**

Develop a distributed system to find sum of N elements in an array by distributing N/n elements to n number of processors MPI or OpenM2.

**HW/SW REQUIREMENTS**

|  |  |  |
| --- | --- | --- |
| **Software** | **Language Used** | **Software Required** |
|  |  |
|  |  |

**THEORY**

A distributed system consists of a collection of autonomous computers, connected through a network and distribution middleware, which enables computers to coordinate their activities and to share the resources of the system so that users perceive the system as a single, integrated computing facility.

Let us say about Google Web Server, from users perspective while they submit the searched query, they assume google web server as a single system. However, behind the curtain, Google has built a lot of servers which is distributed (geographically and computationally) to give us the result within a few seconds.

Message Passing Interface (MPI) is a standardized and portable message-passing system developed for distributed and parallel computing. MPI provides parallel hardware vendors with a clearly defined base set of routines that can be efficiently implemented. As a result, hardware vendors can build upon this collection of standard low-level routines to create higher-level routines for the distributed-memory communication environment supplied with their parallel machines.

MPI gives users the flexibility of calling a set of routines from C, C++, Fortran, C#, Java, or Python. The advantages of MPI over older message passing libraries are portability (because MPI has been implemented for almost every distributed memory architecture) and speed (because each implementation is in principle optimized for the hardware on which it runs)

The advantages of MPI over other message-passing framework is portability and speed. It has been implemented for almost every distributed memory architecture and each implementation is in principle optimized for the hardware on which it runs.

Even though there are options available for multiple languages, Python is the most preferred one due to its simplicity, and ease of writing the code. So, now, we will now look at how to install MPI on ubuntu 14.10.

MPI IMPLEMENTATION

Steps:-

1. Set MPJ\_HOME environment variables:

export MPJ\_HOME= --path to mpj directory --

2. Write your MPJ Express program (ScatterGather.java) and save it.

3. Compile : javac -cp $MPJ\_HOME/lib/mpj.jar ScatterGather.java

4. Execute : $MPJ\_HOME/bin/mpjrun.sh -np 4 ScatterGather

**QUESTIONS & ANSWERS**

* + - 1. **Is MPI shared or Distributed?**
      2. **Why MPI is faster than OpenMp?**
      3. **Is OpenMp Shared or Distributed?**
      4. **Can openMp and MPI be used together?**
      5. **What is the difference between OpenMPI and MPI?**

|  |  |  |  |
| --- | --- | --- | --- |
| **Assignment No. 04** | | **Date** |  |
| **Title** | **Berkely Algorithm** | | |

**PROBLEM DEFINITION**

Implement Berkely Algorithm for Clock Synchronization.

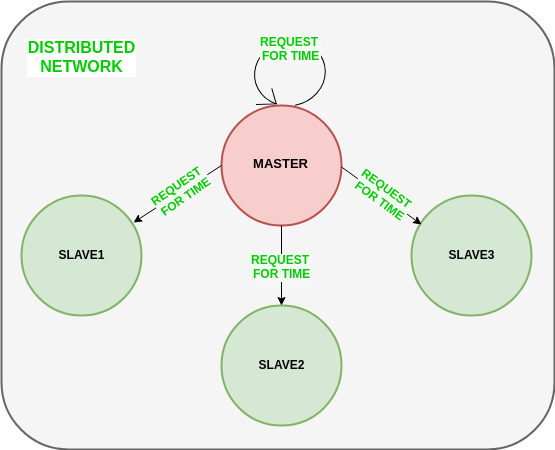
**HW/SW REQUIREMENTS**

|  |  |  |
| --- | --- | --- |
| **Software** | **Language Used** | **Software Required** |
|  |  |
|  |  |

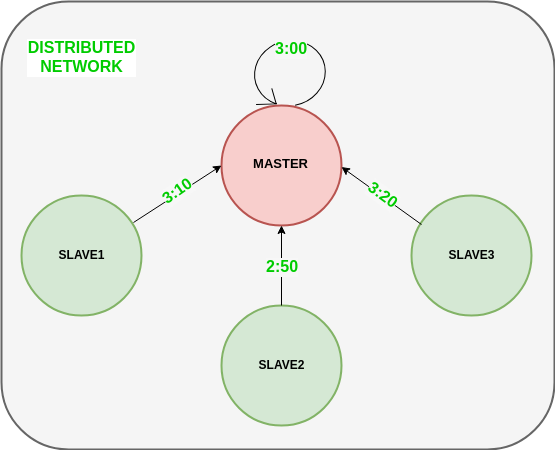
**THEORY**

**I. Introduction**

Berkeley’s Algorithm is a clock synchronization technique used in distributed systems. The algorithm assumes that each machine node in the network either doesn’t have an accurate time source or doesn’t possess a UTC server.  
**Algorithm**   
1) An individual node is chosen as the master node from a pool node in the network. This node is the main node in the network which acts as a master and the rest of the nodes act as slaves. The master node is chosen using an election process/leader election algorithm.  
2) Master node periodically pings slaves nodes and fetches clock time at them using [Cristian’s algorithm](https://www.geeksforgeeks.org/cristians-algorithm/).  
The diagram below illustrates how the master sends requests to slave nodes. 

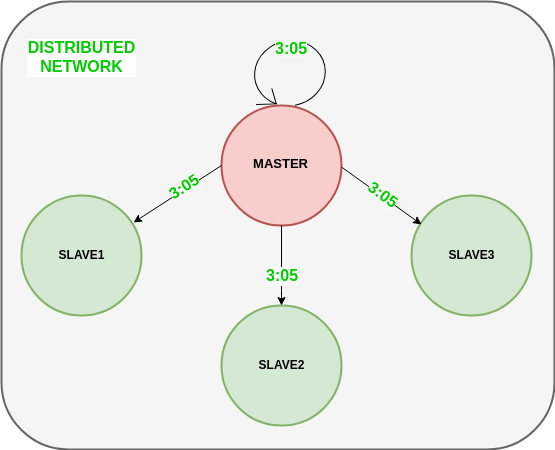


The diagram below illustrates how slave nodes send back time given by their system clock. 



3) Master node calculates the average time difference between all the clock times received and the clock time given by the master’s system clock itself. This average time difference is added to the current time at the master’s system clock and broadcasted over the network.  
Pseudocode for above step:

The diagram below illustrates the last step of Berkeley’s algorithm. 



**Scope of Improvement**

* Improvision inaccuracy of Cristian’s algorithm.
* Ignoring significant outliers in the calculation of average time difference
* In case the master node fails/corrupts, a secondary leader must be ready/pre-chosen to take the place of the master node to reduce downtime caused due to the master’s unavailability.
* Instead of sending the synchronized time, master broadcasts relative inverse time difference, which leads to a decrease in latency induced by traversal time in the network while the time of calculation at slave node.

**features of Berkeley’s Algorithm:**

**Centralized time coordinator:** Berkeley’s Algorithm uses a centralized time coordinator, which is responsible for maintaining the global time and distributing it to all the client machines.

**Clock adjustment:**The algorithm adjusts the clock of each client machine based on the difference between its local time and the time received from the time coordinator.

**Average calculation:**The algorithm calculates the average time difference between the client machines and the time coordinator to reduce the effect of any clock drift.

**Fault tolerance:** Berkeley’s Algorithm is fault-tolerant, as it can handle failures in the network or the time coordinator by using backup time coordinators.

**Accuracy: T**he algorithm provides accurate time synchronization across all the client machines, reducing the chances of errors due to time discrepancies.

**Scalability:**The algorithm is scalable, as it can handle a large number of client machines, and the time coordinator can be easily replicated to provide high availability.

**Security:**Berkeley’s Algorithm provides security mechanisms such as authentication and encryption to protect the time information from unauthorized access or tampering.

**QUESTIONS & ANSWERS**

1. **What is physical clock?**
2. **Why clocks need to be synchronized?**
3. **Explain Clock synchronization Algorithm.**
4. **What is event Ordering?**
5. **Explain Berkely Algorithm.**
6. **Write a note on: Coordinated Universal Time, Drifting of clocks, Clock Skew(Offset).**

|  |  |  |  |
| --- | --- | --- | --- |
| **Assignment No. 05** | | **Date** |  |
| **Title** | **Token Ring Algorithm** | | |

**PROBLEM DEFINITION**

Implement Token Ring based mutual exclusion algorithm.

**HW/SW REQUIREMENTS**

|  |  |  |
| --- | --- | --- |
| **Software** | **Language Used** | **Software Required** |
|  |  |
|  |  |

**THEORY**

**I. Introduction**

Token Ring algorithm achieves mutual exclusion in a distributed system by creating a bus

network of processes. A logical ring is constructed with these processes and each process is

assigned a position in the ring. Each process knows who is next in line after itself. When the

ring is initialized, process 0 is given a token. The token circulates around the ring. When a

process acquires the token from its neighbor, it checks to see if it is attempting to enter a critical

region. If so, the process enters the region, does all the work it needs to, and leaves the region.

After it has exited, it passes the token to the next process in the ring. It is not allowed to enter

the critical region again using the same token. If a process is handed the token by its neighbor

and is not interested in entering a critical region, it just passes the token along to the next

process.

• Advantages:

o The correctness of this algorithm is evident. Only one process has the token at any

instant, so only one process can be in a CS

o Since the token circulates among processes in a well-defined order, starvation

cannot occur.

• Disadvantages

o Once a process decides it wants to enter a CS, at worst it will have to wait for every

other process to enter and leave one critical region.

o If the token is ever lost, it must be regenerated. In fact, detecting that it is lost is

difficult, since the amount of time between successive appearances of the token on

the network is not a constant. The fact that the token has not been spotted for an

hour does not mean that it has been lost; some process may still be using it.

o The algorithm also runs into trouble if a process crashes, but recovery is easier than

in the other cases. If we require a process receiving the token to acknowledge

receipt, a dead process will be detected when its neighbor tries to give it the token

and fails. At that point the dead process can be removed from the group, and the

token holder can pass the token to the next member down the line

Token Ring algorithm achieves mutual exclusion in a distributed system by creating a bus

network of processes. A logical ring is constructed with these processes and each process is

assigned a position in the ring. Each process knows who is next in line after itself. When the

ring is initialized, process 0 is given a token. The token circulates around the ring. When a

process acquires the token from its neighbor, it checks to see if it is attempting to enter a critical

region. If so, the process enters the region, does all the work it needs to, and leaves the region.

After it has exited, it passes the token to the next process in the ring. It is not allowed to enter

the critical region again using the same token. If a process is handed the token by its neighbor

and is not interested in entering a critical region, it just passes the token along to the next

process.

Token Ring algorithm achieves mutual exclusion in a distributed system by creating a bus

network of processes. A logical ring is constructed with these processes and each process is

assigned a position in the ring. Each process knows who is next in line after itself. When the

ring is initialized, process 0 is given a token. The token circulates around the ring. When a

process acquires the token from its neighbor, it checks to see if it is attempting to enter a critical

region. If so, the process enters the region, does all the work it needs to, and leaves the region.

After it has exited, it passes the token to the next process in the ring. It is not allowed to enter

the critical region again using the same token. If a process is handed the token by its neighbor

and is not interested in entering a critical region, it just passes the token along to the next

process.

The Token Ring algorithm is a distributed algorithm used for achieving mutual exclusion in a system where multiple processes compete for a shared resource. In this algorithm, processes are organized in a logical ring structure, and a special token is passed among the processes to determine which process has the right to access the shared resource. The main method creates an array of processes and starts them. The first process acquires the token initially, ensuring that the token starts circulating in the ring. Finally, the main thread waits for all the processes to finish using the **join** method.

Note that in a real distributed system, the processes would communicate and pass the token among themselves, possibly using message passing techniques. However, this simplified example demonstrates the core idea of the Token Ring algorithm for mutual exclusion.

Token Ring algorithm achieves mutual exclusion in a distributed system by creating a bus

network of processes. A logical ring is constructed with these processes and each process is

assigned a position in the ring. Each process knows who is next in line after itself. When the

ring is initialized, process 0 is given a token. The token circulates around the ring. When a

process acquires the token from its neighbor, it checks to see if it is attempting to enter a critical

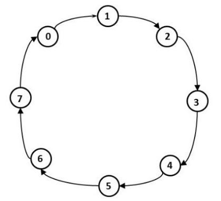
region. If so, the process enters the region, does all the work it needs to, and leaves the region.

After it has exited, it passes the token to the next process in the ring. It is not allowed to enter

the critical region again using the same token. If a process is handed the token by its neighbor

and is not interested in entering a critical region, it just passes the token along to the next

process.



* In this algorithm it is assumed that all the processes in the system are organized in a logical ring. The figure blow describes the structure.
* The ring positions may be allocated in numerical order of network addresses and is unidirectional in the sense that all messages are passed only in clockwise or anti-clockwise direction.
* When a process sends a request message to current coordinator and does not receive a reply within a fixed timeout, it assumes the coordinator has crashed. It then initializes the ring and process Pi is given a token.
* The token circulates around the ring. It is passed from process k to k+1 in point to point messages. When a process acquires the token from its neighbor it checks to see if it is attempting to enter a critical region. If so the process enters the region does all the execution and leaves the region. After it has exited it passes the token along the ring. It is not permitted to enter a second critical region using the same token.
* If a process is handed the token by its neighbor and is not interested in entering a critical region it just passes along. When no processes want to enter any critical regions the token just circulates at high speed around the ring.
* Only one process has the token at any instant so only one process can actually be in a critical region. Since the token circulates among the process in a well-defined order, starvation cannot occur.
* Once a process decides it wants to enter a critical region, at worst it will have to wait for every other process to enter and leave one critical region.
* The disadvantage is that if the token is lost it must be regenerated. But the detection of lost token is difficult. If the token is not received for a long time it might not be lost but is in use.

Bottom of Form

Token Ring algorithm achieves mutual exclusion in a distributed system by creating a bus

network of processes. A logical ring is constructed with these processes and each process is

assigned a position in the ring. Each process knows who is next in line after itself. When the

ring is initialized, process 0 is given a token. The token circulates around the ring. When a

process acquires the token from its neighbor, it checks to see if it is attempting to enter a critical

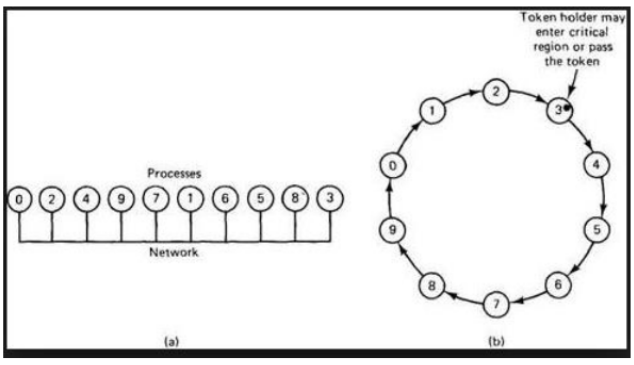
region. If so, the process enters the region, does all the work it needs to, and leaves the region.

After it has exited, it passes the token to the next process in the ring. It is not allowed to enter

the critical region again using the same token. If a process is handed the token by its neighbor

and is not interested in entering a critical region, it just passes the token along to the next

process.

****

**QUESTIONS & ANSWERS**

1. **What is Token ring algorithm to achieve mutual exclusion?**
2. **Which topology is most commonly used with token ring network?**
3. **Why did token ring failed?**
4. **What type of bridge is used for token ring network?**
5. **Which algorithm is best for mutual exclusion?**
6. **What is the size of token in ring topology?**

|  |  |  |  |
| --- | --- | --- | --- |
| **Assignment No. 06** | | **Date** |  |
| **Title** | **Bully Ring Algorithm** | | |

**PROBLEM DEFINITION**

Implement Bully and Ring algorithm for leader election.

**HW/SW REQUIREMENTS**

|  |  |  |
| --- | --- | --- |
| **Software** | **Language Used** | **Software Required** |
|  |  |
|  |  |

**THEORY**

**I. Introduction**

**Election Algorithms:** Election algorithms choose a process from a group of processors to act as a coordinator. If the coordinator process crashes due to some reasons, then a new coordinator is elected on other processor. Election algorithm basically determines where a new copy of the coordinator should be restarted. Election algorithm assumes that every active process in the system has a unique priority number. The process with highest priority will be chosen as a new coordinator. Hence, when a coordinator fails, this algorithm elects that active process which has highest priority number.Then this number is send to every active process in the distributed system. We have two election algorithms for two different configurations of a distributed system.

**1. The Bully Algorithm –** This algorithm applies to system where every process can send a message to every other process in the system. **Algorithm –** Suppose process P sends a message to the coordinator.

1. If the coordinator does not respond to it within a time interval T, then it is assumed that coordinator has failed.
2. Now process P sends an election messages to every process with high priority number.
3. It waits for responses, if no one responds for time interval T then process P elects itself as a coordinator.
4. Then it sends a message to all lower priority number processes that it is elected as their new coordinator.
5. However, if an answer is received within time T from any other process Q,
   * (I) Process P again waits for time interval T’ to receive another message from Q that it has been elected as coordinator.
   * (II) If Q doesn’t responds within time interval T’ then it is assumed to have failed and algorithm is restarted.

**2. The Ring Algorithm –** This algorithm applies to systems organized as a ring(logically or physically). In this algorithm we assume that the link between the process are unidirectional and every process can message to the process on its right only. Data structure that this algorithm uses is **active list**, a list that has a priority number of all active processes in the system.

**Algorithm –**

1. If process P1 detects a coordinator failure, it creates new active list which is empty initially. It sends election message to its neighbour on right and adds number 1 to its active list.
2. If process P2 receives message elect from processes on left, it responds in 3 ways:
   * (I) If message received does not contain 1 in active list then P1 adds 2 to its active list and forwards the message.
   * (II) If this is the first election message it has received or sent, P1 creates new active list with numbers 1 and 2. It then sends election message 1 followed by 2.
   * (III) If Process P1 receives its own election message 1 then active list for P1 now contains numbers of all the active processes in the system. Now Process P1 detects highest priority number from list and elects it as the new coordinator.

**QUESTIONS & ANSWERS**

1. **Explain need of election in a network.**
2. **Difference between Bully and Ring Algorithm.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Assignment No. 07** | | **Date** |  |
| **Title** | **Web Service Creation** | | |

**PROBLEM DEFINITION**

Create a simple webservice and write any distributed application to consume the web service.

**HW/SW REQUIREMENTS**

|  |  |  |
| --- | --- | --- |
| **Software** | **Language Used** | **Software Required** |
|  |  |
|  |  |

**THEORY**

**Introduction**

Distributed systems have been a part of computer science for decades. They are systems where one or more computers or devices communicate with other computers or devices. Such a system ideally has a dynamic life where other systems may join or quit at any time without the whole system failing. The communication between the components happens only by passing messages. The technology of web services on the other hand, is a relatively new development. It is based upon the principles of distributed systems. A web service is a set of functions that are published to a network for use by other programs. Many people regard web services as a technology only for publishing software services on the Internet via browsers, while others regard them as the "new big thing" in distributed computing that is working as general purpose architectures. This thesis will analyse both technologies to see if web services can be used as a foundation for distributed systems

The Internet is the worldwide connectivity of hundreds of thousands of computers of various types that belong to multiple networks. On the World Wide Web, a web service is a standardized method for propagating messages between client and server applications. A web service is a software module that is intended to carry out a specific set of functions. Web services in cloud computing can be found and invoked over the network.  
The web service would be able to deliver functionality to the client that invoked the web service.

A web service is a set of open protocols and standards that allow data to be exchanged between different applications or systems. Web services can be used by software programs written in a variety of programming languages and running on a variety of platforms to exchange data via computer networks such as the Internet in a similar way to inter-process communication on a single computer.

Any software, application, or cloud technology that uses standardized web protocols (HTTP or HTTPS) to connect, interoperate, and exchange data messages – commonly XML (Extensible Markup Language) – across the internet is considered a web service.  
Web services have the advantage of allowing programs developed in different languages to connect with one another by exchanging data over a web service between clients and servers. A client invokes a web service by submitting an XML request, which the service responds with an XML response.

Functions of Web Services

* It’s possible to access it via the internet or intranet networks.
* XML messaging protocol that is standardized.
* Operating system or programming language independent.
* Using the XML standard, it is self-describing.
* A simple location approach can be used to locate it.

**File -1(server side)**

1)download net-beans(version 13)

2)go to file -&gt;new project-&gt;java web-&gt;web application-&gt;fill details(proj name) -&gt;finish.

3)right click on created project -&gt; NEW-&gt;Web Service-&gt;add web service name-&gt;add package

as(org.me.string)-&gt;click checkbox -&gt;finish.

4)we can add operations under webservice folder(created)-&gt;add name of the

operation(stringops) -&gt;add parameters as per requirements.

5)add our logic to newly created stringops file.

6)Right click on NewWebApp(created file) -&gt;Deploy

7)after deploy we can check output tab and under that server we can see where the server is

listening.

8)after every change do deploy for reflecting the changes.

(Optional)- 9)after deploy go to string application which is available in folder Web Services

.-&gt;right click on string aaplication and click on test services to test the web service is deployed or

not.

**File-2(Client side):**

1) go to file -&gt;new project-&gt;java -&gt;web application-&gt;fill details(give project name-

new\_webApp clientApplin) -&gt;finish.

2) Right click on new\_Web\_client App -&gt;NEW-&gt;Web Service Client-&gt;browse the project and

select String application under NewWebApp(earlier created ).-&gt;Ok-&gt;finish.

3) Web Service refrences created -&gt;go on destructuring the folder until the stringOps and then

drag and drop the stringOps file into NewWebApp\_Client file.

4) Update the code and call the operations.

5) Right click on client file(into NewWebApp\_Client) and run .

|  |
| --- |
|  |

**QUESTIONS & ANSWERS**

1. **Explain Web Service Definition Language (WSDL).**
2. **What are the limitations of using GET Hyperlinks and POST Hyperlinks to communicate with a web service.**
3. **How to make a class accessible as a webservice?**
4. **How to make a method of web service class accessible through internet?**