Explain different types of Distributed System.

A Distributed System is a Network of Machines that can exchange information with each other through Message-passing. It can be very useful as it helps in resource sharing.

**1.Client/Server Systems:** Client requests to the server for resources or a task to do, the server allocates the resource or performs the task and sends the result in the form of response to the request of the client.

**2.Peer to Peer Systems**: As nodes are an important part of a system. In this, each node performs their own task on their local memory and share data through the supporting medium, this node can work as a server or as a client for a system.

**3.Middleware:** It works as a base for different interoperability applications running on different operating systems.  Data can be transferred to other between others by using this service.

**4.Three-tier:** In this data of the client is stored in the middle tire rather than sorting into the client system or on their server through which development can be done easily. This is mostly used in web or online applications.

**5.N-tier**: When interoperability sends the request to another application to perform a task or to provide a service.

===========================================================

**What is distributed transparency ? Explain types of transparency.**

Distribution transparency is the property of distributed databases by the virtue of which the internal details of the distribution are hidden from the users.

The three dimensions of distribution transparency are −

1. Location transparency
2. Fragmentation transparency
3. Replication transparency

*Location Transparency*

Location transparency ensures that the user can query on any table(s) or fragment(s) of a table as if they were stored locally in the user’s site. The fact that the table or its fragments are stored at remote site in the distributed database system, should be completely oblivious to the end user. The address of the remote site(s) and the access mechanisms are completely hidden.

Fragmentation Transparency

Fragmentation transparency enables users to query upon any table as if it were unfragmented. Thus, it hides the fact that the table the user is querying on is actually a fragment or union of some fragments. It also conceals the fact that the fragments are located at diverse sites.

Replication Transparency

Replication transparency ensures that replication of databases are hidden from the users. It enables users to query upon a table as if only a single copy of the table exists.Whenever a user updates a data item, the update is reflected in all the copies of the table.

=============================================================

**What is three-tier architecture?**

Three-tier architecture is a well-established software application architecture that organizes applications into three logical and physical computing tiers: the presentation tier, or user interface; the application tier, where data is processed; and the data tier, where the data associated with the application is stored and managed.

The 3-tier architecture consists of the three layers as follows −

**1.Presentation layer −** This layer is also called the client layer. The front-end layer consists of a user interface. The main purpose is to communicate with the application layer.

**2.Application layer** − This layer is also called the business logic layer. It acts as a middle layer between the client and the database server which are used to exchange partially processed data.

**3.Database layer −** In this layer the data or information is stored. This layer performs operations like insert, update and delete to connect with the database.

Diagram

Description automatically generated

=============================================================

**Content-Addressable-Network (CAN)**

CAN: Internet Scale Hash table

Interface

insert(key,value)

value = retrieve(key)

Idea: associate to each node and item a unique

coordinate in an d-dimensional Cartesian space.

Properties

–scalable

–operationally simple

–good performance

**Difference between Centralized Database and Distributed Database**

1. ***Centralized Database***

Diagram

Description automatically generatedA centralized database is basically a type of database that is stored, located as well as maintained at a single location only. This type of database is modified and managed from that location itself. This location is thus mainly any database system or a centralized computer system. The centralized location is accessed via an internet connection (LAN, WAN, etc). This centralized database is mainly used by institutions or organizations.

**Advantages** –

->Since all data is stored at a single location only thus it is easier to access and coordinate data.

->The centralized database has very minimal data redundancy since all data is stored in a single place.

**Disadvantages –**

->The data traffic in the case of centralized database is more.

->If any kind of system failure occurs at the centralized system then the entire data will be destroyed.

***2. Distributed Database :***

Diagram

Description automatically generatedA distributed database is basically a type of database which consists of

multiple databases that are connected with each other and are spread across different physical locations. The data that is stored on various physical locations can thus be managed independently of other physical locations. The communication between databases at different physical locations is thus done by a computer network.

Advantages –

->This database can be easily expanded as data is already spread across different physical locations.

->The distributed database can easily be accessed from different networks.

Disadvantages –

->This database is very costly and it is difficult to maintain because of its complexity.

->In this database, it is difficult to provide a uniform view to user since it is spread across different physical locations.

A picture containing text, watch, clock

Description automatically generated**Centralized systems and decentralized systems**

**CENTRALIZED SYSTEMS:**

We start with centralized systems because they are the most intuitive and easy to understand and define.

Centralized systems are systems that use client/server architecture where one or more client nodes are directly connected to a central server. This is the most used type of system in many organizations where a client sends a request to a company server and receives the response.

Example –

Wikipedia. Consider a massive server to which we send our requests and the server responds with the article that we requested. Suppose we enter the search term ‘junk food’ in the Wikipedia search bar. This search term is sent as a request to the Wikipedia servers (mostly located in Virginia, U.S.A) which then responds back with the articles based on relevance. In this situation, we are the client node, Wikipedia servers are the central server.

**Characteristics of Centralized System –**

**->Presence of a global clock**: As the entire system consists of a central node(a server/ a master) and many client nodes(a computer/ a slave), all client nodes sync up with the global clock(the clock of the central node).

**->One single central unit:** One single central unit which serves/coordinates all the other nodes in the system.

**->Dependent failure of components**: Central node failure causes the entire system to fail. This makes sense because when the server is down, no other entity is there to send/receive responses/requests.

**DECENTRALIZED SYSTEMS:**

These are other types of systems that have been gaining a lot of popularity, primarily because of the massive hype of Bitcoin. Now many organizations are trying to find the application of such systems.

In decentralized systems, every node makes its own decision. The final behavior of the system is the aggregate of the decisions of the individual nodes. Note that there is no single entity that receives and responds to the request.

Example –

Bitcoin. Let’s take Bitcoin for example because it is the most popular use case of decentralized systems. No single entity/organization owns the bitcoin network. The network is a sum of all the nodes who talk to each other for maintaining the amount of bitcoin every account holder has.

**Characteristics of Decentralized System –**

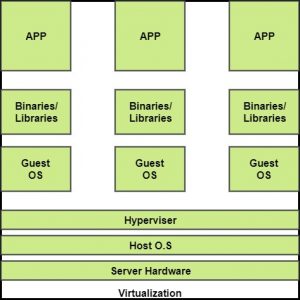
**->Lack of a global clock:** Every node is independent of each other and hence, has different clocks that they run and follow.

**->Multiple central units (Computers/Nodes/Servers):** More than one central unit which can listen for connections from other nodes

**->Dependent failure of components**: one central node failure causes a part of the system to fail; not the whole system

**Virtualization in Cloud Computing and Types**

Virtualization is a technique of how to separate a service from the underlying physical delivery of that service. It is the process of creating a virtual version of something like computer hardware. It was initially developed during the mainframe era. It involves using specialized software to create a virtual or software-created version of a computing resource rather than the actual version of the same resource. With the help of Virtualization, multiple operating systems and applications can run on same machine and its same hardware at the same time, increasing the utilization and flexibility of hardware.



The machine on which the virtual machine is going to be built is known as Host Machine and that virtual machine is referred as a Guest Machine.

**BENEFITS OF VIRTUALIZATION**

1. More flexible and efficient allocation of resources.

2. Enhance development productivity.

3. It lowers the cost of IT infrastructure.

4. Remote access and rapid scalability.

**Types of Virtualization:**

1.Application Virtualization.

2.Network Virtualization. 3.Desktop Virtualization.

4.Storage Virtualization. 5.Server Virtualization

**Explains X Window System**

The client/server model in X system works in reverse to typical client/server model, where the client runs on the local machine and asks for services from the server. In X system, the server runs on the local machine and provides its display and services to the client programs. The client programs may be local or remotely exist over different networks, but appear transparently.

X is used in networks of interconnected mainframes, minicomputers, workstations, and X Terminals. X window system consists of a number of interacting components, including:

X server: Manages the display and input hardware. It captures command-based and graphics-based inputs from input hardware and passes it to the client application that requested it. It also receives inputs from the client applications and displays the output under guidance from windows manager. The only component that interacts with hardware is X server. This makes it easier to recode it as per the requirements of different hardware architectures.

Windows manager: Is the client application that manages client windows. It controls the general operations of the window system like geometry, appearance, coordinates, and graphical properties of X display. Window manager can change the size and position of windows on the display and reshuffle windows in a window stack.

X client: Is an application program that communicates with X server using X protocol. Xterm, Xclock, and Xcalc are examples of X clients. X manages its windows in a hierarchal structure. The shaded area that fills the entire screen is the root window. X client application windows are displayed on top of the root window and are often called the children of the root.

**Cluster System**

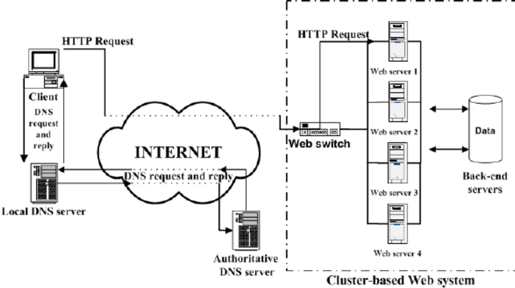
Cluster : It means that multiple servers are grouped together to achieve the same business and can be regarded as one computer.

A group of computers consisting of multiple servers, as a whole, provides users with a set of network resources, which are the nodes of the cluster.

Two features

Scalability : A service node in a cluster that dynamically adds machines to increase the processing power of the cluster.

High availability : If a node in a cluster fails, the services running on this node can be taken over by other service nodes, thus enhancing the high availability of the cluster.



**What is Code Migration**

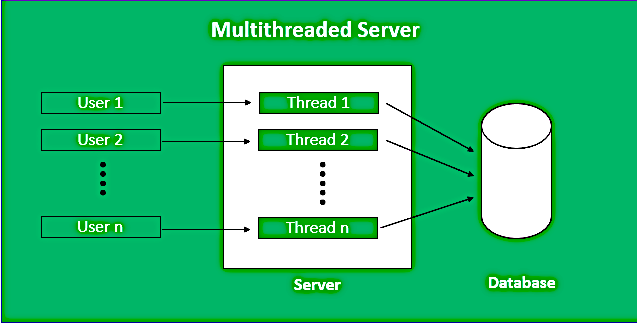
Code migration is the movement of programming code from one system to another. There are three distinct levels of code migration with increasing complexity, cost and risk. Simple migration involves the movement from language to a newer version. A second, more complicated level of migration involves moving to a different programming language. Migrating to an entirely new platform or operating system is the most complex type of migration.

Diagram

Description automatically generated

**Multithreaded Server Architecture**

A server having more than one thread is known as Multithreaded Server. When a client sends the request, a thread is generated through which a user can communicate with the server. We need to generate multiple threads to accept multiple requests from multiple clients at the same time.



**Advantages of Multithreaded Server:**

**->Quick and Efficient:** Multithreaded server could respond efficiently and quickly to the increasing client queries quickly.

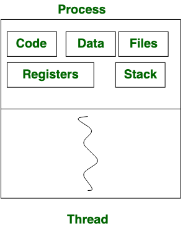
**->Waiting time for users decreases:** In a single-threaded server, other users had to wait until the running process gets completed but in multithreaded servers, all users can get a response at a single time so no user has to wait for other processes to finish.

**Disadvantages of Multithreaded Server:**

**->Complicated Code**: It is difficult to write the code of the multithreaded server. These programs can not be created easily

**->Debugging is difficult**: Analyzing the main reason and origin of the error is difficult.

**Process and thread**

**Processes** are basically the programs that are dispatched from the ready state and are scheduled in the CPU for execution. PCB(Process Control Block) holds the concept of process. A process can create other processes which are known as Child Processes. The process takes more time to terminate and it is isolated means it does not share the memory with any other process. The process can have the following states new, ready, running, waiting, terminated, and suspended.

**Thread:**

Thread is the segment of a process means a process can have multiple threads and these multiple threads are contained within a process. A thread has three states: Running, Ready, and Blocked. The thread takes less time to terminate as compared to the process but unlike the process, threads do not isolate.

|  |  |
| --- | --- |
| **Process** | **Thread** |
| Process means any program is in execution. | Thread means a segment of a process. |
| The process takes more time to terminate | The thread takes less time to terminate |
| It takes more time for creation | It takes less time for creation |
| It also takes more time for context Switching | It takes less time for context switching |
| The process has its own Process Control Block, Stack, and Address Space. | Thread has Parents’ PCB, its own Thread Control Block, and Stack and common Address space. |
| Process switching uses an interface in an operating system. | Thread switching does not require calling an operating system and causes an interrupt to the kernel. |

**REMOTE PROCEDURE CALL(RPC)**

A remote procedure call is an interposes communication technique that is used for client-server-based applications. It is also known as a subroutine call or a function call. A client has a request message that the RPC translates and sends to the server. This request may be a procedure or a function call to a remote server. When the server receives the request, it sends the required response back to the client. The client is blocked while the server is processing the call and only resumed execution after the server is finished.

The sequence of events in a remote procedure call are given as follows −

1.The client stub is called by the client.

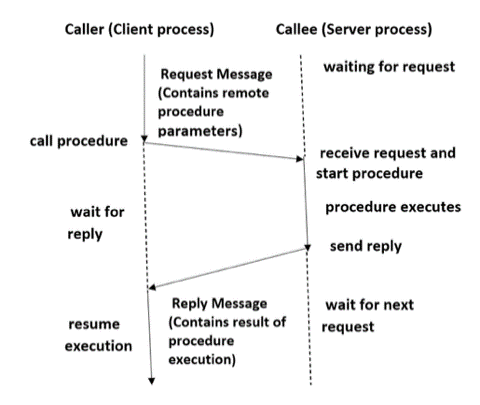
2.The client stub makes a system call to send the message to the server and puts the parameters in the message.

3.The message is sent from the client to the server by the client’s operating system.

4.The message is passed to the server stub by the server operating system.

5.The parameters are removed from the message by the server stub.

6.Then the server procedure is called by server stub



**User Datagram Protocol (UDP)**

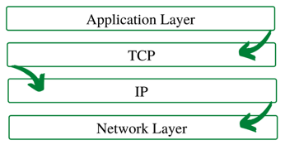
User Datagram Protocol (UDP) is a Transport Layer protocol. UDP is a part of the Internet Protocol suite, referred to as UDP/IP suite. Unlike TCP, it is an unreliable and connectionless protocol. So, there is no need to establish a connection prior to data transfer.

Table

Description automatically generated

* Source Port: Source Port is a 2 Byte long field used to identify the port number of the source.
* Destination Port: It is a 2 Byte long field, used to identify the port of the destined packet.
* Length: Length is the length of UDP including the header and the data. It is a 16-bits field.
* Checksum: Checksum is 2 Bytes long field. It is the 16-bit one’s complement of the one’s complement sum of the UDP header, the pseudo-header of information from the IP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

**Transmission Control Protocol**

TCP (Transmission Control Protocol) is one of the main protocols of the Internet protocol suite. It lies between the Application and Network Layers which are used in providing reliable delivery services. It is a connection-oriented protocol for communications that helps in the exchange of messages between the different devices over a network.

For example, When a user requests a web page on the internet, somewhere in the world, the server processes that request and sends back an HTML Page to that user. The server makes use of a protocol called the HTTP Protocol. The HTTP then requests the TCP layer to set the required connection and send the HTML file.

|  |  |
| --- | --- |
| TCP is a connection-oriented protocol. Connection-orientation means that the communicating devices should establish a connection before transmitting data and should close the connection after transmitting the data. | UDP is the Datagram-oriented protocol. This is because there is no overhead for opening a connection, maintaining a connection, and terminating a connection. UDP is efficient for broadcast and multicast types of network transmission. |
| TCP is reliable as it guarantees the delivery of data to the destination router | The delivery of data to the destination cannot be guaranteed in UDP |
| Comparatively Slower than UDP | Faster, simpler and more efficient than TCP |
| 20-60 bytes variable length header | 8 bytes fixed length header |
| Heavy weight | Lightweight |

**Multicasting in Computer Network**

Multicast is a method of group communication where the sender sends data to multiple receivers or nodes present in the network simultaneously. Multicasting is a type of one-to-many and many-to-many communication as it allows sender or senders to send data packets to multiple receivers at once across LANs or WANs. This process helps in minimizing the data frame of the network.

Chart, bubble chart

Description automatically generated

Multicasting works in like Broadcasting, but in Multicasting, the information is sent to the targeted or specific members of the network. This task can be accomplished by transmitting individual copies to each user or node present in the network but sending individual copies to each user is inefficient and might increase the network latency. To overcome these shortcomings, multicasting allows a single transmission that can be split up among the multiple users, consequently, this reduces the bandwidth of the signal.

Applications:

Multicasting is used in many areas like:

* Internet protocol (IP)
* Streaming Media

It also supports video conferencing applications and webcasts.

**IP Multicast:**

Multicasting that takes place over the Internet is known as IP Multicasting. These multicasts follow the internet protocol (IP) to transmit data. IP multicasting uses a mechanism known as ‘Multicast trees’ to transmit to information among the users of the network. Multicast trees: allows a single transmission to branch out to the desired receivers. The branches are created at the Internet routers, the branches are created such that the length of the transmission will be minimum.

**Remote Method Invocation (RMI)**

Remote Method Invocation (RMI) is an API that allows an object to invoke a method on an object that exists in another address space, which could be on the same machine or on a remote machine. Through RMI, an object running in a JVM present on a computer (Client-side) can invoke methods on an object present in another JVM (Server-side). RMI creates a public remote server object that enables client and server-side communications through simple method calls on the server object.

Stub Object: The stub object on the client machine builds an information block and sends this information to the server.

The block consists of

* An identifier of the remote object to be used
* Method name which is to be invoked
* Parameters to the remote JVM

Skeleton Object: The skeleton object passes the request from the stub object to the remote object. It performs the following tasks

* It calls the desired method on the real object present on the server.
* Diagram

  Description automatically generatedIt forwards the parameters received from the stub object to the method.

**Group Communication in distributed Systems**

Communication between two processes in a distributed system is required to exchange various data, such as code or a file, between the processes. When one source process tries to communicate with multiple processes at once, it is called Group Communication. This abstraction is to hide the message passing so that the communication looks like a normal procedure call. Group communication also helps the processes from different hosts to work together and perform operations in a synchronized manner, therefore increases the overall performance of the system.

**Types of Group Communication in a Distributed System :**

1**.Broadcast Communication :**

Diagram

Description automatically generatedWhen the host process tries to communicate with every process in a distributed system at same time. Broadcast communication comes in handy when a common stream of information is to be delivered to each and every process in most efficient manner possible. Since it does not require any processing whatsoever, communication is very fast in comparison to other modes of communication. However, it does not support a large number of processes and cannot treat a specific process individually.

Diagram

Description automatically generated**2.Multicast Communication:**When the host process tries to communicate with a designated group of processes in a distributed system at the same time. This technique is mainly used to find a way to address problem of a high workload on host system and redundant information from process in system. Multitasking can significantly decrease time taken for message handling.

A multicast Communication: P1 process communicating with only a group of the process in the system

**Diagram

Description automatically generated3.Unicast Communication:**

When the host process tries to communicate with a single process in a distributed system at the same time. Although, same information may be passed to multiple processes. This works best for two processes communicating as only it has to treat a specific process only. However, it leads to overheads as it has to find exact process and then exchange information/data.

**Client server communication**

Clients and servers exchange messages in a request response messaging pattern. The client sends a request, and the server returns a response. This exchange of messages is an example of inter-process communication.

To communicate, the computers must have a common language, and they must follow rules so that both the client and the server know what to expect. The language and rules of communication are defined in a communication.

Diagram

Description automatically generatedAll client-server protocols operate in the application layer. The application-layer protocol defines the basic patterns of the dialogue.

To formalize the data exchange even further, the server may implement an API(such as a web service).The API is an abstraction layer for such resources as databases and custom software. By restricting communication to a specific content format. By abstracting access, it facilitate cross platform data exchange.

Example

* When a bank customer accesses online banking services with a web browser (the client), the client initiates a request to the bank’s web server.
* The customer’s login credentials may be stored in a database, and the web server accesses the database server as a client.

**Name Resolution**

Name resolution is the process of mapping an object's name to the object's properties, such as its location. Since an object's properties are stored and maintained by the authoritative name servers of that object, name resolution is basically the process of mapping an object's name to the authoritative name servers of that object. Once an authoritative name server of the object has been located, operations can be invoked to read or update the object's properties. Each name agent in a distributed system knows about vat least one name server apriorism. To get a name resolved, a client first contacts its name agent, which in turn contacts a known name server, which may in turn contact other name servers.

**The DNS Name Space**

The DNS name space is the set of all names used with DNS. This space is partitioned hierarchically and is case insensitive.

The current DNS name space is a tree of domains with an unnamed root at the top. The top echelons of the tree are the so-called top-level domains (TLDs), which include:

* Generic TLDs (gTLDs)
* Country-code TLDs (ccTLDs)
* Internationalized country-code TLDs (IDN ccTLDs)
* A special infrastructure TLD called, for historical reasons, ARPA [RFC3172].

**What Is LDAP?**

Companies store usernames, passwords, email addresses, printer connections, and other static data within directories. LDAP is an open, vendor-neutral application protocol for accessing and maintaining that data. LDAP can also tackle authentication, so users can sign on just once and access many different files on the server.

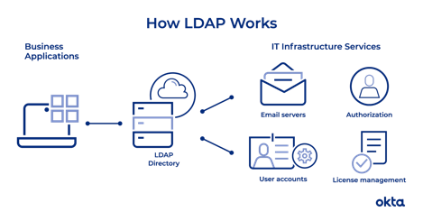
An LDAP query typically involves:

1.Session connection. The user connects to the server via an LDAP port.

2.Request. The user submits a query, such as an email lookup, to the server.

3.Response. The LDAP protocol queries the directory, finds the information, and delivers it to the user.

4.Completion. The user disconnects from the LDAP port.



**Name Services and the Domain Name System**

A name service stores a collection of one or more naming contexts, sets of bindings between textual names and attributes for objects such as computers, services, and users.

The major operation that a name service supports is to resolve names.

**Uniform Resource Identifiers**

Uniform Resource Identifiers (URIs) came about from the need to identify resources on the Web, and other Internet resources such as electronic mailboxes. An important goal was to identify resources in a coherent way, so that they could all be processed by common software such as browsers.

**Uniform Resource Locators:** Some URIs contain information that can be used to locate and access a resource; others are pure resource names. The familiar term Uniform Resource Locator (URL) is often used for URIs that provide location information and specify the method for accessing the resource.

Uniform Resource Names: Uniform Resource Names (URNs) are URIs that are used as pure resource names rather than locators. For example, the URI:

=============================================================

**What is DNS? How does it works?**

The domain name system (DNS) is a naming database in which internet domain names are located and translated into Internet Protocol (IP) addresses. The domain name system maps the name people use to locate a website to the IP address that a computer uses to locate that website.

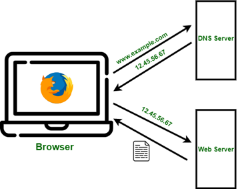
For example, if someone types "example.com" into a web browser, a server behind the scenes maps that name to the corresponding IP address. An IP address is similar in structure to 203.0.113.72.

**Working of DNS**

1.DNS is a client/server network communication protocol. DNS clients send requests to the. server while DNS servers send responses to the client.

2.Client requests contain a name which is converted into an IP address known as a forward DNS lookups while requests containing an IP address which is converted into a name known as reverse DNS lookups.

3.DNS implements a distributed database to store the name of all the hosts available on the internet.



**What christian’s algorithm of physical clock synchronization.?**

Cristian’s Algorithm is a clock synchronization algorithm is used to synchronize time with a time server by client processes. This algorithm works well with low-latency networks where Round Trip Time is short as compared to accuracy while redundancy-prone distributed systems/applications do not go hand in hand with this algorithm. Here Round Trip Time refers to the time duration between the start of a Request and the end of the corresponding Response.

**Algorithm:**

1) The process on the client machine sends the request for fetching clock time(time at the server) to the Clock Server at time T\_0 .

2) The Clock Server listens to the request made by the client process and returns the response in form of clock server time.

3) The client process fetches the response from the Clock Server at time T\_1 and calculates the synchronized client clock time using the formula given below.

\[ T\_{CLIENT} = T\_{SERVER} + (T\_1 - T\_0)/2 \]

where T\_{CLIENT} refers to the synchronized clock time,

T\_{SERVER} refers to the clock time returned by the server,

T\_0 refers to the time at which request was sent by the client process,

T\_1 refers to the time at which response was received by the client process

**Explain Logical Clocks in detail?**

Logical Clocks refer to implementing a protocol on all machines within your distributed system, so that the machines are able to maintain consistent ordering of events within some virtual timespan. A logical clock is a mechanism for capturing chronological and causal relationships in a distributed system. Distributed systems may have no physically synchronous global clock, so a logical clock allows global ordering on events from different processes in such systems.

Example:

If we go outside then we have made a full plan that at which place we have to go first, second and so on. We don’t go to second place at first and then the first place. We always maintain the procedure or an organization that is planned before. In a similar way, we should do the operations on our PCs one by one in an organized way.

Suppose we have more than 10 PCs in a distributed system and every PC is doing it’s own work but then how we make them work together. There comes a solution to this i.e., LOGICAL CLOCK.

**Explain Recart-Agrawala algorithm of distributed mutual exclusion.**

Ricart–Agrawala algorithm is an algorithm to for mutual exclusion in a distributed system proposed by Glenn Ricart and Ashok Agrawala. This algorithm is an extension and optimization of Lamport’s Distributed Mutual Exclusion Algorithm. Like Lamport’s Algorithm, it also follows permission based approach to ensure mutual exclusion.

**In this algorithm:**

1.Two type of messages ( REQUEST and REPLY) are used and communication channels are assumed to follow FIFO order.

2.A site send a REQUEST message to all other site to get their permission to enter critical section.

3.A site send a REPLY message to other site to give its permission to enter the critical section.

4.A timestamp is given to each critical section request using Lamport’s logical clock.

5.Timestamp is used to determine priority of critical section requests. Smaller timestamp gets high priority over larger timestamp. The execution of critical section request is always in the order of their timestamp.

**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 coordinator does not respond to it within a time interval T, then it is assumed that coordinator has failed.

2.Now process P sends election message 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,

6.Process P again waits for time interval T’ to receive another message from Q that it has been elected as coordinator.

7.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 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 neighbor 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:

3.If message received does not contain 1 in active list then P1 adds 2 to its active list and forwards the message.

4.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.

5.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.

**What Does Distributed System Mean? goals and challenges of DS ?**

A distributed system is any network structure that consists of autonomous computers that are connected using a distribution middleware. Distributed systems facilitate sharing different resources and capabilities, to provide users with a single and integrated coherent network.

The opposite of a distributed system is a centralized system. If all of the components of a computing system reside in one machine, as was the case with early mainframes such as Von Neumann machines, it is not a distributed system.

**The key goals of a distributed system include:**

1.Transparency: Achieving the image of a single system image without concealing the details of the location, access, migration, concurrency, failure, relocation, persistence and resources to the users

2.Openness: Making the network easier to configure and modify

3.Reliability: Compared to a single system, a distributed system should be highly capable of being secure, consistent and have a high capability of masking errors.

4.Performance: Compared to other models, distributed models are expected to give a much-wanted boost to performance.

5.Scalability: Distributed systems should be scalable with respect to geography, administration or size.

**Challenges for distributed systems include:**

1.Security is a big challenge in a distributed environment, especially when using public networks.

2.Fault tolerance could be tough when the distributed model is built based on unreliable components.

3.Coordination and resource sharing can be difficult if proper protocols or policies are not in place.

4.Process knowledge should be put in place for the administrators and users of the distributed model.

## Comparison Table Between Absolute and Relative Path

| **Parameter of Comparison** | **Absolute Path** | **Relative Path** |
| --- | --- | --- |
| By definition | specifies the location from the root directory | related to the location from current directory |
| Function of delimiting character | Begins with a delimiting character | Never begins with a delimiting character |
| Navigates to | Content from other domains | Content from the same domain |
| URL used | Uses absolute URL | Used relative URL |
| Other names | Full-path or File path | Non-absolute path |

**Global Name Service (GNS)**  
Designed and implemented by Lampson and colleagues at the DEC Systems Research Center (1986)Provide facilities for resource location, addressing and authenticationWhen the naming database grows from small to large scale, the structure of name space may changethe service should accommodate itCache consistency ?

GNS Structure

->Tree of directories holding names and values

->Muti-part pathnames refer to the root or relative working directory (like Unix file system)

->Unique Directory Identifier (DI)

->A directory contains list of names and references

->Leaves of tree contain value trees (structured values)

Diagram

Description automatically generated

**Name Spaces**

A name space is a collection of all valid names recognized by a particular serviceAllow simple but meaningful names to be usedPotentially infinite number of namesStructuredto allow similar subnames without clashesto group related namesAllow re-structuring of name treesfor some types of change, old programs should continue to workManagement of trust.

Diagram

Description automatically generated

**What is Thread? What are it’s types ?**

A thread is a flow of execution through the process code, with its own program counter that keeps track of which instruction to execute next, system registers which hold its current working variables, and a stack which contains the execution history.

**Types of Thread**

**1.User Level Threads**

In this case, the thread management kernel is not aware of the existence of threads. The thread library contains code for creating and destroying threads, for passing message and data between threads, for scheduling thread execution and for saving and restoring thread contexts. The application starts with a single thread.

Advantages

1.Thread switching does not require Kernel mode privileges.

2.User level thread can run on any operating system.

3.Scheduling can be application specific in the user level thread.

4.User level threads are fast to create and manage.

Disadvantages

1.In a typical operating system, most system calls are blocking.

2.Multithreaded application cannot take advantage of multiprocessing.

**2.Kernel Level Threads**

In this case, thread management is done by the Kernel. There is no thread management code in the application area. Kernel threads are supported directly by the operating system. Any application can be programmed to be multithreaded. All of the threads within an application are supported within a single process.

The Kernel maintains context information for the process as a whole and for individuals threads within the process. Scheduling by the Kernel is done on a thread basis. The Kernel performs thread creation, scheduling and management in Kernel space. Kernel threads are generally slower to create and manage than the user threads.

**Advantages**

1.Kernel can simultaneously schedule multiple threads from the same process on multiple processes.

2.If one thread in a process is blocked, the Kernel can schedule another thread of the same process.

3.Kernel routines themselves can be multithreaded.

**Disadvantages**

1.Kernel threads are generally slower to create and manage than the user threads.

2.Transfer of control from one thread to another within the same process requires a mode switch to the Kernel.

|  |  |  |
| --- | --- | --- |
| **S.N.** | **User-Level Threads** | **Kernel-Level Thread** |
| 1 | User-level threads are faster to create and manage. | Kernel-level threads are slower to create and manage. |
| 2 | Implementation is by a thread library at the user level. | Operating system supports creation of Kernel threads. |
| 3 | User-level thread is generic and can run on any operating system. | Kernel-level thread is specific to the operating system. |
| 4 | Multi-threaded applications cannot take advantage of multiprocessing. | Kernel routines themselves can be multithreaded. |

**Computer Network Architecture**

Computer Network Architecture is defined as the physical and logical design of the software, hardware, protocols, and media of the transmission of data. Simply we can say that how computers are organized and how tasks are allocated to the computer.

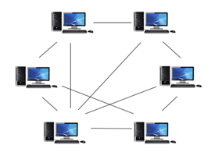
The two types of network architectures are used:

Computer Network Architecture

Peer-To-Peer network

* Client/Server network

**Peer-To-Peer network**

Peer-To-Peer network is a network in which all the computers are linked together with equal privilege and responsibilities for processing the data.

Peer-To-Peer network is useful for small environments, usually up to 10 computers.

Peer-To-Peer network has no dedicated server.

Special permissions are assigned to each computer for sharing the resources, but this can lead to a problem if the computer with the resource is down.

**Advantages Of Peer-To-Peer Network:**

1.It is less costly as it does not contain any dedicated server.

2.If one computer stops working but, other computers will not stop working.

3.It is easy to set up and maintain as each computer manages itself.

**Disadvantages Of Peer-To-Peer Network:**

In the case of Peer-To-Peer network, it does not contain the centralized system . Therefore, it cannot back up the data as the data is different in different locations.

It has a security issue as the device is managed itself.

**Client/Server Network**

Client/Server network is a network model designed for the end users called clients, to access the resources such as songs, video, etc. from a central computer known as Server.

The central controller is known as a server while all other computers in the network are called clients.

A server performs all the major operations such as security and network management.

A server is responsible for managing all the resources such as files, directories, printer, etc.

All the clients communicate with each other through a server. For example, if client1 wants to send some data to client 2, then it first sends the request to the server for the permission. The server sends the response to the client 1 to initiate its communication with the client 2.

**Computer Network Architecture**

**Advantages Of Client/Server network**:

1.A Client/Server network contains the centralized system. Therefore we can back up the data easily.

2.A Client/Server network has a dedicated server that improves the overall performance of the whole system.

3.Security is better in Client/Server network as a single server administers the shared resources.

4.It also increases the speed of the sharing resources.

**Disadvantages Of Client/Server network:**

1.Client/Server network is expensive as it requires the server with large memory.

2.A server has a Network Operating System(NOS) to provide the resources to the clients, but the cost of NOS is very high.

3.It requires a dedicated network administrator to manage all the resources.

**Digital Signature**

• A digital signature—a type of electronic signature—is a

mathematical algorithm routinely used to validate the

authenticity and integrity of a message (e.g., an email, a credit

card transaction, or a digital document).

• Digital signatures create a virtual fingerprint that is unique to

a person or entity and are used to identify users and protect

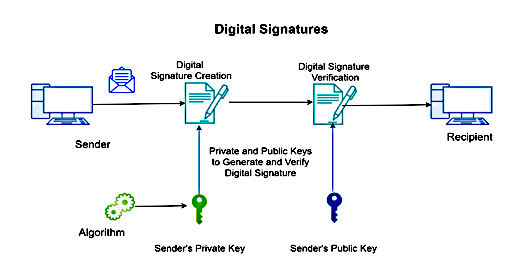
information in digital messages or documents.

• In emails, the email content itself becomes part of the digital

signature.

• Digital signatures are significantly more secure than other

forms of electronic signatures.



**Vector Clocks in Distributed Systems**

Vector Clock is an algorithm that generates partial ordering of events and detects causality violations in a distributed system. These clocks expand on Scalar time to facilitate a causally consistent view of the distributed system, they detect whether a contributed event has caused another event in the distributed system. It essentially captures all the causal relationships. This algorithm helps us label every process with a vector(a list of integers) with an integer for each local clock of every process within the system. So for N given processes, there will be vector/ array of size N.

**How does the vector clock algorithm work :**

1.Initially, all the clocks are set to zero.

2.Every time, an Internal event occurs in a process, the value of the processes’s logical clock in the vector is incremented by 1

3.Also, every time a process sends a message, the value of the processes’s logical clock in the vector is incremented by 1.

Every time, a process receives a message, the value of the processes’s logical clock in the vector is incremented by 1, and moreover, each element is updated by taking the maximum of the value in its own vector clock and the value in the vector in the received message (for every element).

Example :

Consider a process (P) with a vector size N for each process: the above set of rules mentioned are to be executed by the vector clock:

Chart

Description automatically generated

The above example depicts the vector clocks mechanism in which the vector clocks are updated after execution of internal events, the arrows indicate how the values of vectors are sent in between the processes (P1, P2, P3).

To sum up, Vector clocks algorithms are used in distributed systems to provide a causally consistent ordering of events but the entire Vector is sent to each process for every message sent, in order to keep the vector clocks in sync.

|  |  |  |
| --- | --- | --- |
| **Basis of Comparison** | **Centralized database** | **Distributed database** |
| **Definition** | It is a database that is stored, located as well as maintained at a single location only. | It is a database that consists of multiple databases which are connected with each other and are spread across different physical locations. |
| **Access time** | The data access time in the case of multiple users is more in a centralized database. | The data access time in the case of multiple users is less in a distributed database. |
| **Management of data** | The management, modification, and backup of this database are easier as the entire data is present at the same location. | The management, modification, and backup of this database are very difficult as it is spread across different physical locations. |
| **View** | This database provides a uniform and complete view to the user. | Since it is spread across different locations thus it is difficult to provide a uniform view to the user. |
| **Data Consistency** | This database has more data consistency in comparison to distributed database. | This database may have some data replications thus data consistency is less. |

**Explain the middleware with its importance in distributed system with suitable diagram.**

Distributed computing problems are simplified by many vendors as they are offering distributed system services that have standard programing interfaces and protocols. These distributed system services are called middleware. As they sit in the middle, layering above the operating system and networking software below industry specific applications. Middleware may have multiple roles, but the most common one is usually handling communication between components of the system. At a high level, it's analogous to the postal system - you can send [almost] anything through the mail by packaging and addressing it in a standard manner. The carrier makes no assumptions about the contents, on the assumption that the receiver knows what they represent.

Importance:

1. Focus on end-to-end support and integration, not just individual components: There is now widespread recognition that effective development of large-scale distributed systems requires the use of COTS infrastructure and service components. Moreover, the usability of the resulting products depends heavily on the weaving of the properties of the whole as derived from its parts. In its most useful forms, middleware provides the end-to-end perspective extending across elements applicable to the network substrate, the platform operating systems and system services, the programming system in which they are developed, the applications themselves, and the middleware that integrates all these elements together.
2. The increased viability of open systems architectures and open-source availability: By their very nature, distributed systems developed by composing separate components are more open than systems conceived and developed as monolithic entities. The focus on interfaces for integrating and controlling the component parts leads naturally to standard interfaces. In turn, this yields the potential for multiple choices for component implementations, and open engineering concepts.
3. Advanced common infrastructure sustaining continuous innovation: Middleware: supporting component integration and reuse is a key technology to help amortize software life-cycle costs by leveraging previous development expertise, e.g., component middleware helps to abstract commonly reused low-level OS concurrency and networking details away into higher level, more easily used artifacts. Likewise, middleware also focuses efforts to improve software quality and performance by combining aspects of a larger solution together, e.g., component middleware combines fault tolerance for domain-specific elements with real-time QoS properties.

Diagram

Description automatically generated

|  |  |
| --- | --- |
| **Client-Server Network** | **Peer-to-Peer Network** |
| In Client-Server Network, Clients and server are differentiated, Specific server and clients are present. | In Peer-to-Peer Network, Clients and server are not differentiated. |
| Client-Server Network focuses on information sharing. | While Peer-to-Peer Network focuses on connectivity. |
| In Client-Server Network, Centralized server is used to store the data. | While in Peer-to-Peer Network, Each peer has its own data. |
| In Client-Server Network, Server respond the services which is request by Client. | While in Peer-to-Peer Network, Each and every node can do both request and respond for the services. |
| Client-Server Network are costlier than Peer-to-Peer Network. | While Peer-to-Peer Network are less costly than Client-Server Network. |
| Client-Server Network are more stable than Peer-to-Peer Network. | While Peer-to-Peer Network are less stable if number of peers is increase. |
| Client-Server Network is used for both small and large networks. | While Peer-to-Peer Network is generally suited for small networks with fewer than 10 computers. |

**What is interposes communication in distributed system? Explain. Between RPC and RMI which one is better in terms of communication with suitable example.**

Inter process Communication is a process of exchanging the data between two or more independent process in a distributed environment is called as Inter process communication. Inter process communication on the internet provides both Datagram and stream communication.

Examples Of Inter Process Communication:

1.N number of applications can communicate with the X server through network protocols.

2.Servers like Apache spawn child processes to handle requests.

3.Pipes are a form of IPC: grep foo file | sort

|  |  |
| --- | --- |
| **RPC** | **RMI** |
| RPC is a website for libraries and OS. | It is a forum for java. |
| RPC facilitates the programming of the procedures. | RMI supports programming oriented to the object. |
| RPC is less powerful. | RMI is more powerful. |
| There is no protection for RPC. | It offers protection at the customer level. |
| For basic RPC applications, several codes are required. | Multiple codes for basic RMI applications are not necessary. |

Conclusion:

RPC and RMI are frameworks that permit a client to invoke server processing or method by communicating with the client-server. The common distinction between RPC and RMI is that RPC only supports proceedings, while the other big difference between RMI and RPC is the fact that the parameters transferred to remote operations have regular data structures.

| **S.NO** | **Authentication** | **Authorization** |
| --- | --- | --- |
| 1. | In authentication process, the identity of users are checked for providing the access to the system. | While in authorization process, person’s or user’s authorities are checked for accessing the resources. |
| 2. | In authentication process, users or persons are verified. | While in this process, users or persons are validated. |
| 3. | It is done before the authorization process. | While this process is done after the authentication process. |
| 4. | It needs usually user’s login details. | While it needs user’s privilege or security levels. |

**Explain transient Asynchronous communication with suitable example.**

Diagram, schematic

Description automatically generatedAns: A sends the message and continues execution (nonblocking). B has to be running, because if it is not running the message will be discarded. Even if any router along the way is down, the message will be discarded. UDP communication is an example of transient asynchronous communication. The function MP I b send() is an implementation of this.

Example:

**Why it is difficult to synchronous physical clock? Explain how synchronous can be solved using logical clock.**

Ans: Communication between processes in a distributed system can have unpredictable delays, processes can fail, messages may be lost ▪ Synchronization in distributed systems is harder than in centralized systems because the need for distributed algorithms.

Properties of distributed algorithms:

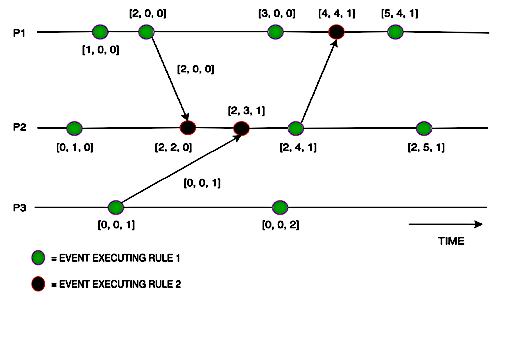
1 The relevant information is scattered among multiple machines.

2 Processes make decisions based only on locally available information.

3 A single point of failure in the system should be avoided.

4 No common clock or other precise global time source exists.

Logical Clocks refer to implementing a protocol on all machines within your distributed system, so that the machines are able to maintain consistent ordering of events within some virtual timespan. A logical clock is a mechanism for capturing chronological and causal relationships in a distributed system. Distributed systems may have no physically synchronous global clock, so a logical clock allows global ordering on events from different processes in such systems. If we go outside then we have made a full plan that at which place we have to go first, second and so on. We don’t go to second place at first and then the first place. We always maintain the procedure or an organization that is planned before. In a similar way, we should do the operations on our PCs one by one in an organized way. Suppose, we have more than 10 PCs in a distributed system and every PC is doing it’s own work but then how we make them work together. There comes a solution to this i.e. LOGICAL CLOCK.

**What do you understand by Caching and Replication in web? Explain.**

A cache is a temporary storage location for copied information. There are over a billion pages (or objects) on the internet. Many users request the same popular objects. An example of that would be the top logo image of Yahoo.com which appears in almost all Yahoo pages. The image must be delivered to the browser each time the browser accesses any of Yahoo's pages these pages are requested a number of times each day by different users. A Web cache is a dedicated computer system which will monitor the object requests and stores objects as it retrieves them from the server. On subsequent requests the cache will deliver objects from its storage rather than passing the request to the origin server. Every Web object changes over time and therefore has a useful life or "freshness”. If the freshness of an object expires it is the responsibility of the Web cache to get the new version of the object. The more the number of requests for the same object the more effective will the Web cache be in reducing upstream traffic and will also help reducing server load, resulting in less latency.

Replication is a technique similar to caching but is generally considered to be more active. The process of replication copies cache content and pushes it on to one or more cache servers across the network. Replication is required to distribute objects among the servers to maintain the freshness of content across servers, which results in reduced upstream network traffic. Typically, the same content is pushed across several machines making it more efficient to use Multicast. Replication is critical in global operations, where cost of international traffic is high and ways have to be found to mirror data without using too much bandwidth.

============================================================

**Describe different types of security mechanisms to protect against security threats in distributed systems.**

Ans: Network Security is field in computer technology that deals with ensuring security of computer network infrastructure. As the network is very necessary for sharing of information whether it is at hardware level such as printer, scanner, or at software level. Therefore, security mechanism can also be termed as is set of processes that deal with recovery from security attack. Various mechanisms are designed to recover from these specific attacks at various protocol layers.

Types of Security Mechanism are

Encipherment:

This security mechanism deals with hiding and covering of data which helps data to become confidential. It is achieved by applying mathematical calculations or algorithms which reconstruct information into not readable form.

Access Control:

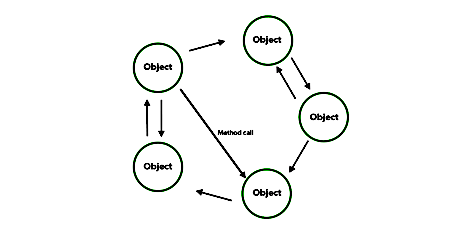
This mechanism is used to stop unattended access to data which you are sending. It can be achieved by various techniques such as applying passwords, using firewall, or just by adding PIN to data.

Notarization:

This security mechanism involves use of trusted third party in communication. It acts as mediator between sender and receiver so that if any chance of conflict is reduced. This mediator keeps record of requests made by sender to receiver for later denied.

Data Integrity:

This security mechanism is used by appending value to data to which is created by data itself. It is similar to sending packet of information known to both sending and receiving parties and checked before and after data is received.



Authentication exchange:

This security mechanism deals with identity to be known in communication. This is achieved at the TCP/IP layer where two-way handshaking mechanism is used to ensure data is sent or not

Bit stuffing:

This security mechanism is used to add some extra bits into data which is being transmitted. It helps data to be checked at the receiving end and is achieved by Even parity or Odd Parity.

Digital Signature:

This security mechanism is achieved by adding digital data that is not visible to eyes. It is form of electronic signature which is added by sender which is checked by receiver electronically. This mechanism is used to preserve data which is not more confidential but sender’s identity is to be notified.

**Describe Lamport’s algorithm with its benefits and drawbacks.**

Ans: Lamport’s Distributed Mutual Exclusion Algorithm is a permission-based algorithm proposed by Lamport’s as an illustration of his synchronization scheme for distributed systems.

In permission-based timestamp is used to order critical section requests and to resolve any conflict between requests.

In Lamport’s Algorithm critical section requests are executed in the increasing order of timestamps i.e. a request with smaller timestamp will be given permission to execute critical section first than a request with larger timestamp.

Algorithm:

To enter Critical section:

* When a site Si wants to enter the critical section, it sends a request message Request (tsi, i) to all other sites and places the request on request\_queuei. Here, Tsi denotes the timestamp of Site Si
* When a site Sj receives the request message REQUEST (tsi, i) from site Si, it returns a timestamped REPLY message to site Si and places the request of site Si on request\_queuei

.

To execute the critical section:

* A site Si can enter the critical section if it has received the message with timestamp larger than (tsi, i) from all other sites and its own request is at the top of request\_queuei

To release the critical section:

* When a site Si exits the critical section, it removes its own request from the top of its request queue and sends a timestamped RELEASE message to all other sites
* When a site Sj receives the timestamped RELEASE message from site Si, it removes the request of Si from its request queue

Drawbacks of Lamport’s Algorithm:

* •Unreliable approach: failure of any one of the processes will halt the progress of entire system.
* High message complexity: Algorithm requires 3(N-1) messages per critical section invocation.

Performance:

* Synchronization delay is equal to maximum message transmission time
* It requires 3(N – 1) messages per CS execution.
* Algorithm can be optimized to 2(N – 1) messages by omitting the REPLY message in some situations.

**What is vector clock? Why it is important? Explain vector clock algorithm by its example.**

Ans: Vector Clock is an algorithm that generates partial ordering of events and detects causality violations in a distributed system. These clocks expand on Scalar time to facilitate a causally consistent view of the distributed system, they detect whether a contributed event has caused another event in the distributed system. It essentially captures all the causal relationships. This algorithm helps us label every process with a vector (a list of integers) with an integer for each local clock of every process within the system. So for N given processes, there will be vector/ array of size N.

How does the vector clock algorithm work?

* Initially, all the clocks are set to zero.
* Every time, an Internal event occurs in a process, the value of the processes’ logical clock in the vector is incremented by 1
* Also, every time a process sends a message, the value of the processes’ logical clock in the vector is incremented by 1.

Every time, a process receives a message, the value of the processes’ logical clock in the vector is incremented by 1, and moreover, each element is updated by taking the maximum of the value in its own vector clock and the value in the vector in the received message (for every element).

Example:

Consider a process (P) with a vector size N for each process: the above set of rules mentioned are to be executed by the vector clock:

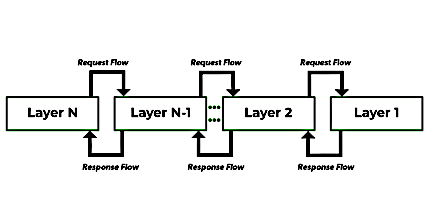
===========================================================

**Explain different types of architecture styles in distributed system.**

1. Layered Architecture:

In Layered architecture, different components are organized in layers. Each layer communicates with its adjacent layer by sending requests and getting responses.  The layered architecture separates components into units. It is an efficient way of communication. Any layer cannot directly communicate with another layer. A layer can only communicate with its neighboring layer and then the next layer transfers information to another layer and so on the process goes on.

In some cases, layered architecture is in cross-layer coordination. In a cross-layer, any adjacent layer can be skipped until it fulfils the request and provides better performance results. Request flow from top to bottom(downwards) and response flow from bottom to top(upwards). The advantage of layered architecture is that each layer can be modified independently without affecting the whole system. This type of architecture is used in Open System Interconnection (OSI) model.



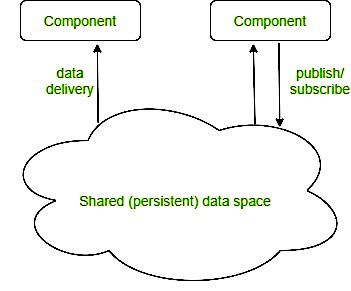
2. Object-Oriented Architecture:

In this type of architecture, components are treated as objects which convey information to each other. Object-Oriented Architecture contains an arrangement of loosely coupled objects. Objects can interact with each other through method calls. Objects are connected to each other through the Remote Procedure Call (RPC) mechanism or Remote Method Invocation (RMI) mechanism. Web Services and REST API are examples of object-oriented architecture.

3. Data Centered Architecture:

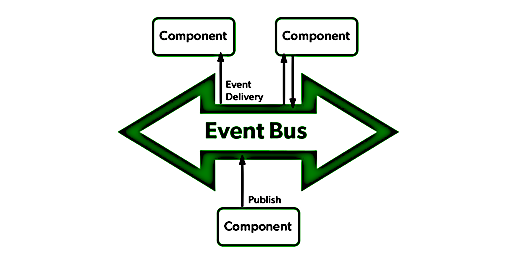
Data Centered Architecture is a type of architecture in which a common data space is present at the center. It contains all the required data in one place a shared data space. All the components are connected to this data space and they follow publish/subscribe type of communication. It has a central data repository at the center. Required data is then delivered to the components.

For example, Producer-Consumer system. The producer produces data in common data space and consumers request data.



4. Event-Based Architecture:

Event-Based Architecture is almost similar to Data centered architecture just the difference is that in this architecture events are present instead of data. Events are present at the center in the Event bus and delivered to the required component whenever needed. In this architecture, the entire communication is done through events. When an event occurs, the system, as well as the receiver, get notified. Data, URLs etc. are transmitted through events. The components of this system are loosely coupled that’s why it is easy to add, remove and modify them. Heterogeneous components can communicate through the bus.



**Explain Remote procedure call and its working process in detail with a suitable diagram. How RPC is different from RMI?**

Remote Procedure Call (RPC) is a powerful technique for constructing distributed, client-server-based applications. It is based on extending the conventional local procedure calling so that the called procedure need not exist in the same address space as the calling procedure. The two processes may be on the same system, or they may be on different systems with a network connecting them.

Diagram

Description automatically generatedWorking of RPC.

The following steps take place during an RPC:

* 1. A client invokes a client stub procedure, passing parameters in the usual way. The client stub resides within the client’s own address space.
  2. The client stub Marshalls(pack) the parameters into a message. Marshalling includes converting the representation of the parameters into a standard format, and copying each parameter into the message.
  3. The client stub passes the message to the transport layer, which sends it to the remote server machine.
  4. On the server, the transport layer passes the message to a server stub, which demarshalls(unpack) the parameters and calls the desired server routine using the regular procedure call mechanism.
  5. When the server procedure completes, it returns to the server stub (e.g., via a normal procedure call return), which Marshalls the return values into a message. The server stub then hands the message to the transport layer.
  6. The transport layer sends the result message back to the client transport layer, which hands the message back to the client stub.
  7. The client stub demarshalls the return parameters and execution return to the caller.

**Three component of security? CIA triad?**

When we discuss data and information, we must consider the CIA triad. The CIA triad refers to an information security model made up of the three main components: confidentiality, integrity and availability. Each component represents a fundamental objective of information security.

The three components of the CIA triad are discussed below:

Confidentiality:

This component is often associated with secrecy and the use of encryption. Confidentiality in this context means that the data is only available to authorized parties. When information has been kept confidential it means that it has not been compromised by other parties; confidential data are not disclosed to people who do not require them or who should not have access to them. Ensuring confidentiality means that information is organized in terms of who needs to have access, as well as the sensitivity of the data. A breach of confidentiality may take place through different means, for instance hacking or social engineering.

Integrity:

Data integrity refers to the certainty that the data is not tampered with or degraded during or after submission. It is the certainty that the data has not been subject to unauthorized modification, either intentional or unintentional. There are two points during the transmission process during which the integrity could be compromised: during the upload or transmission of data or during the storage of the document in the database or collection.

Availability: This means that the information is available to authorized users when it is needed. For a system to demonstrate availability, it must have properly functioning computing systems, security controls and communication channels. Systems defined as critical (power generation, medical equipment, safety systems) often have extreme requirements related to availability. These systems must be resilient against cyber threats, and have safeguards against power outages, hardware failures and other events that might impact the system availability.

**What is data- centric consistency model? Explain its type in details.**

Ans: Traditionally consistency has been discussed in the context of read and write operations on shared data, available by means of shared memory. A shared database, or a file system. Here, we use the broader term data store. A data store may be physically distributed across multiple machines.

Types: -

i. Consistency model: a contract between a (distributed) data store and processes, in which the data store specifies precisely what the results of read and write operations are in the presence of concurrency.

ii. Continuous Consistency:

Observation: We can actually talk a about a degree of consistency:

· Replicas may differ in their numerical value

· Replicas may differ in their relative staleness

· There may differences with respect to (number and order) of performed update operations

Conit: consistency unit) specifies the data unit over which consistency is to be measured.

· e.g., stock record, weather report, etc.

Conit example: numerical and ordering deviations

iii. Strict Consistency

Any read on a data item ‘x’ returns a value corresponding to the result of the most recent write on ‘x’ (regardless of where the write occurred). With Strict Consistency, all writes are instantaneously visible to all processes and absolute global time order is maintained throughout the distributed system. This is the consistency model “Holy Grail” – not at all easy in the real world, and all but impossible within a DS.

iv. Sequential Consistency

· A weaker consistency model, which represents a relaxation of the rules.

It is also must easier (possible) to implement.

Sequential Consistency:

The result of any execution is the same as if the (read and write) operations by all processes on the data-store were executed in the same sequential order and the operations of each individual process appear in this sequence in the order specified by its program.

v. Causal Consistency

· Writes that are potentially causally related must be seen by all processes in the same order.

· Concurrent writes (i.e., writes that are NOT causally related) may be seen in a different order by different processes.

vi. FIFO Consistency

Writes done by a single process are seen by all other processes in the order in which they were issued, but writes from different processes may be seen in a different order by different processes.

· Also called “PRAM Consistency” – Pipelined RAM.

· Easy to implement -There are no guarantees about the order in which different processes see writes – except those two or more writes from a single process must be seen in order.

vii. Weak Consistency

· Not all applications need to see all writes, let alone seeing them in the same order.

· Leads to Weak Consistency (which is primarily designed to work with distributed critical sections).

· This model introduces the notion of a synchronization variable”, which is used update all copies of the data-store.

Properties Weak Consistency:

1. Accesses to synchronization variables associated with a data-store are sequentially consistent.

2. No operation on a synchronization variable is allowed to be performed until all previous writes have been completed everywhere.

3. No read or write operation on data items are allowed to be performed until all previous operations to synchronization variables have been performed.

viii. Release Consistency

· When a process does an acquire, the data-store will ensure that all the local copies of the protected data are brought up to date to be consistent with the remote ones if needs be.

· When a release is done, protected data that have been changed are propagated out to the local copies of the data-store.

ix. Entry consistency

● Acquire and release are still used, and the data-store meets the following conditions:

● An acquire access of a synchronization variable is not allowed to perform with respect to a process until all updates to the guarded shared data have been performed with respect to that process.

● Before an exclusive mode access to a synchronization variable by a process is allowed to perform with respect to that process, no other process may hold the synchronization variable, not even in nonexclusive mode.

● After an exclusive mode access to a synchronization variable has been performed, any other process's next nonexclusive mode access to that synchronization variable may not be performed until it has performed with respect to that variable's owner.

**Cryptography and its Types and features!**

Cryptography is technique of securing information and communications through use of codes so that only those person for whom the information is intended can understand it and process it. Thus preventing unauthorized access to information. The prefix “crypt” means “hidden” and suffix graphy means “writing”.

I**n general there are three types Of cryptography:**

Symmetric Key Cryptography:

It is an encryption system where the sender and receiver of message use a single common key to encrypt and decrypt messages. Symmetric Key Systems are faster and simpler but the problem is that sender and receiver have to somehow exchange key in a secure manner. The most popular symmetric key cryptography system is Data Encryption System(DES).

Hash Functions:

There is no usage of any key in this algorithm. A hash value with fixed length is calculated as per the plain text which makes it impossible for contents of plain text to be recovered. Many operating systems use hash functions to encrypt passwords.

Asymmetric Key Cryptography:

Under this system a pair of keys is used to encrypt and decrypt information. A public key is used for encryption and a private key is used for decryption. Public key and Private Key are different. Even if the public key is known by everyone the intended receiver can only decode it because he alone knows the private key.

**Features Of Cryptography are as follows:**

Confidentiality:

Information can only be accessed by the person for whom it is intended and no other person except him can access it.

Integrity:

Information cannot be modified in storage or transition between sender and intended receiver without any addition to information being detected.

Non-repudiation:

The creator/sender of information cannot deny his intention to send information at later stage.

Authentication:

The identities of sender and receiver are confirmed. As well as destination/origin of information is confirmed.

===========================================================