Arunima Singh Thakur, 180905218, Sec C, Roll no 31, Branch CSE, DS mid sem,

1) Typical middleware services :-

a) Communication :- A common communication service is the so-called Remote Procedure Call (RPC). An RPC service allows an application to invoke a function that is implemented & executed on a remote computer as if it was locally available.

b) Transactions :- Many applications make use of multiple services that are distributed among several computers. Middleware generally offers special support for executing such services in all-or-nothing fashion, commonly referred to as an atomic transaction.

c) Service composition :- Web-based middleware can help by standardizing the way web services are accessed and providing the means to generate their functions in a specific order.

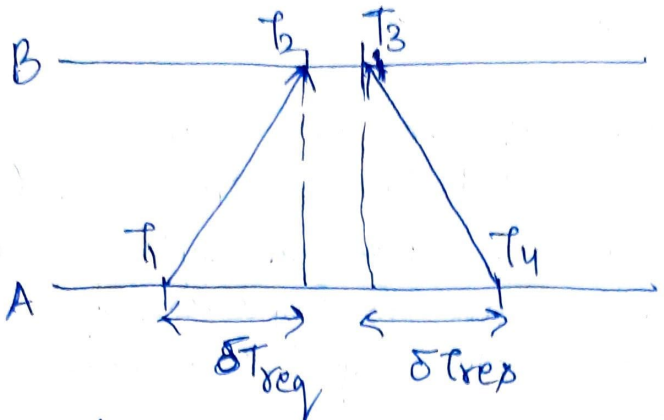
d) Reliability :- It allows a developer to build an application as a group of processes such that any message sent by one process is guaranteed to be received by all or no other process

2) a) Size scalability - A system can be scalable with respect to its size, meaning that we can easily add more users & resources to the system without any noticeable loss of performance.

b) Geographical scalability - This system is one one in which the users and resources may lie far apart, but the fact that communication delays may be significant is hardly noticed.

(c) Administrative scalability – This system is one that can still be easily managed even if it spans many independent administrative organizations

4)



Getting the current time from a time server

In this case, A will send a request to B, timestamped with value T_1. B will record the time of receipt T_2 (taken from its own clock), and returns a response timestamped with value T_3, and piggybanking the previously recorded value T_2. Finally A records the time of the response's arrival, T_4. Assume propagation delay from A to B & B to A is roughly same i.e $T_2 - T_1 \approx T_4 - T_3$. Then A can estimate its offset relative to B as, –

$$\theta = T_3 + \frac{(T_2 - T_1) + (T_4 - T_3)}{2} - T_4$$

$$= \frac{(T_2 - T_1) + T_3 - T_4}{2}$$

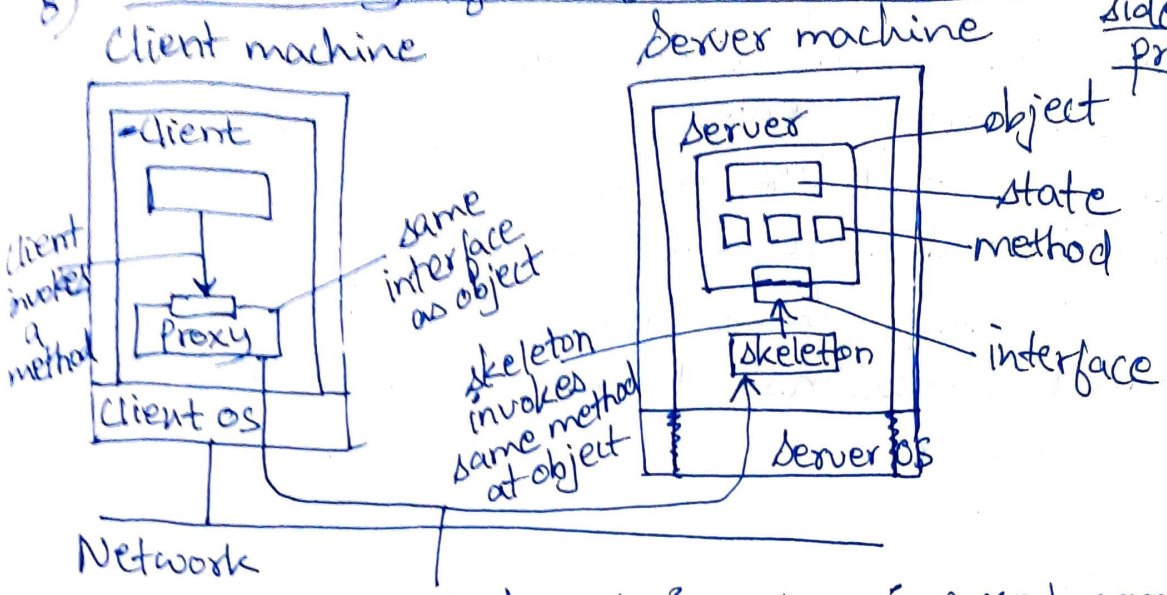
NTP

- A protocol is set up pairwise between servers
- In NTP, B will also probe A for its current time. In this θ as above method and

$$\delta = \frac{(T_2 - T_1) + (T_4 - T_3)}{2}$$

- Eight pairs of (θ, δ) values are buffered & minimum value of δ is taken as the best estimation for delay & associated θ value as reliable estimation of offset.

8) Common Organization of a remote object with client side proxy

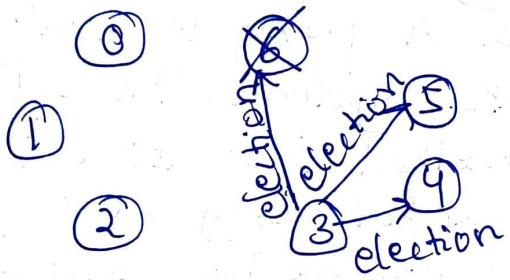


When a client binds to a distributed object or a remote object, an implementation of the object's interface, called a proxy, is then loaded into the client address space. A proxy is analogous to a client stub in RPC systems. The only thing it does is marshal method invocation into messages and unmarshal reply messages to return the result of the method invocation to the client. The server stub is also responsible for marshaling replies and forwarding reply messages to the client-side proxy. The server-side stub is often referred to as a skeleton as it provides the bare means for letting the server middleware access the user-defined objects. A characteristic, but somewhat counterintuitive feature of most distributed objects is that their state is not distributed: it resides at a single machine. Only the interfaces implemented by the object are made available on other machines. Such objects are also referred to as remote objects. In a general distributed object, the state itself may be physically distributed across multiple machines, but this distribution is also hidden from clients behind the objects interfaces.

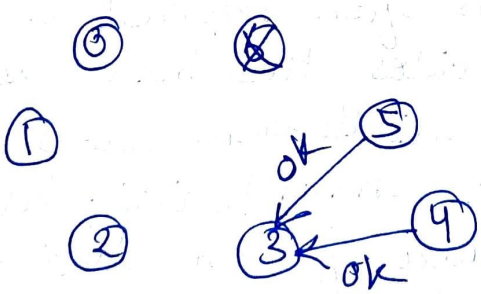
After returning the result of the method invocation to the client, the actual object which resides at a server machine, offers the same interface as it does on the client machine. Incoming invocation requests are first passed to a server stub, which unmarshals them to make method invocations at the object's interface at the server.

---

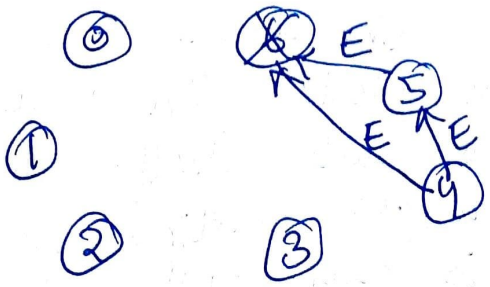
6) Given, 7 processes and 3 noticed that 6 has crashed. Now process 3 will generate an election message and send to higher-numbered colleagues.



Now process 4 & 5 respond telling process 3 to stop by sending ok acknowledgement.

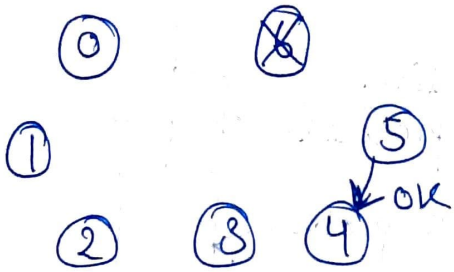


Next 5 and 4 will hold an election

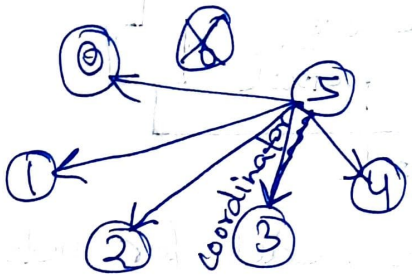


[E indicates election message]

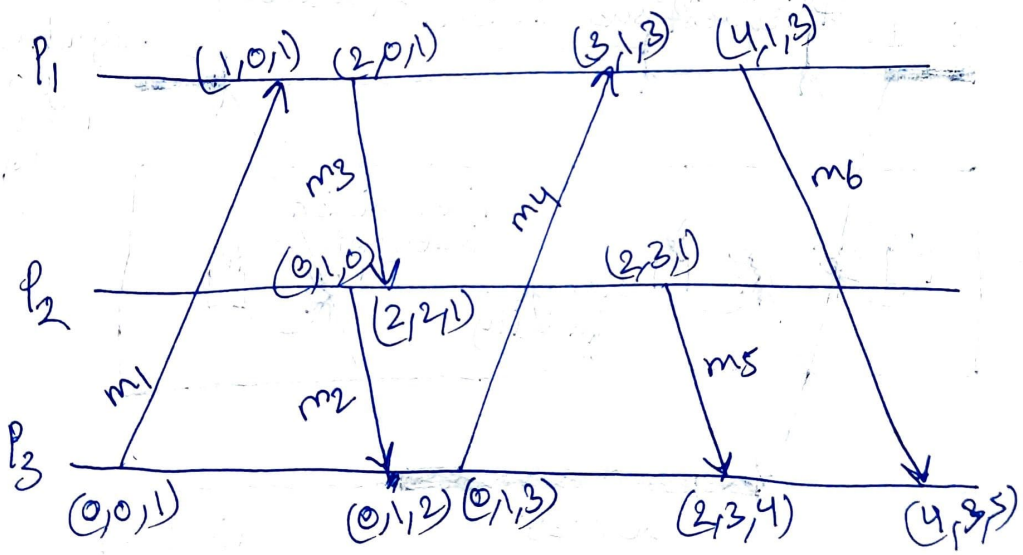
Process 5 sends an ok acknowledgement to Process 4 and tells it to stop.



Thus process 5 wins and tells everyone it is the coordinator

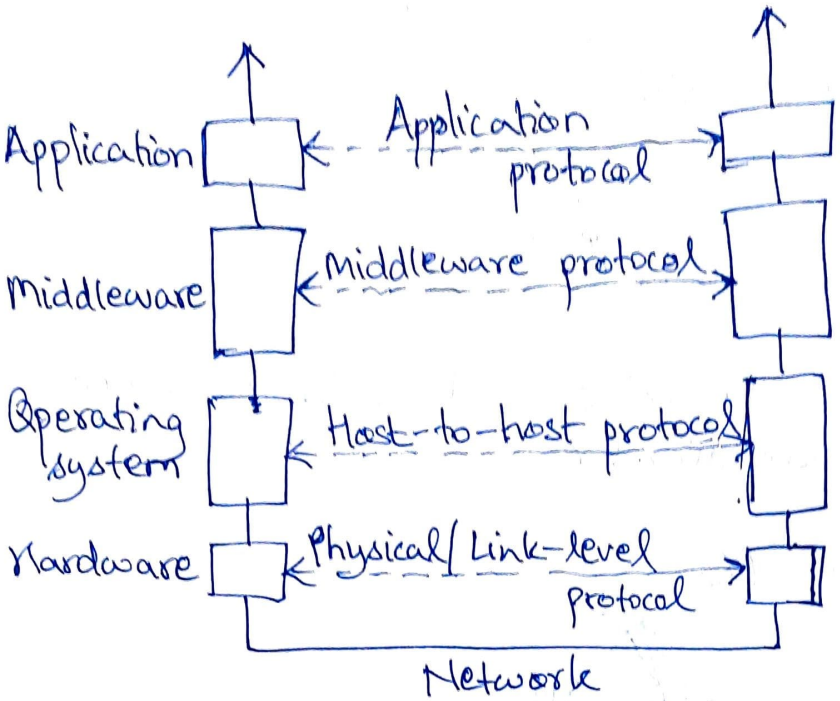


5) Initial (0,0,0)



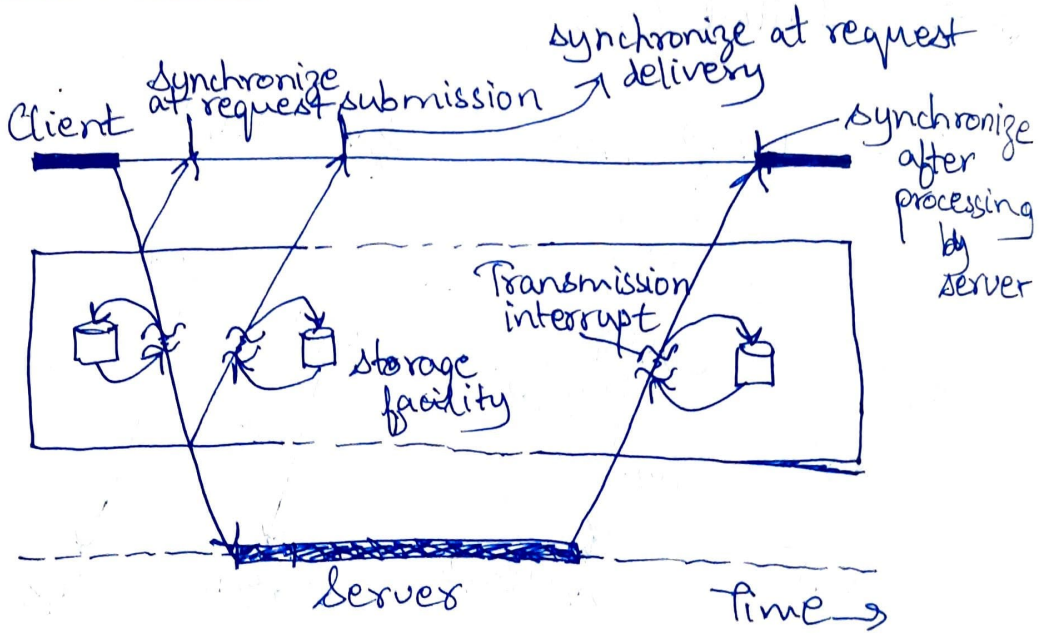
7) In this model, the session and presentation layer have been replaced by a single middleware layer that contains application-independent protocols. Network & transport services have been grouped into communication services as normally offered by an operating system, which, in turn, manages the specific

lowest-level hardware used to establish communication.



An adapted reference model for networked communication

b)



Viewing middleware as an intermediate (distributed) service in application-level communication

Types of communication:-

1) Persistent - A message that has been submitted for transmission is stored by the communication middleware as long as it takes

to deliver it to the receiver.

2) Transient: A message is stored by the communication system only as long as the sending and receiving application are executing.

3) Asynchronous: A sender continues immediately after it has submitted its message for transmission.

4) Synchronous: The sender is blocked until its request is known to be accepted.

---

3) Both clocks tick 1000 times per ms

UTC = Update once in a minute

Maximum clock skew = 1200 ms
= 1.2 seconds

so,

$C_1(t)$ = 1000 times/ms

$C_2(t)$ let it be t times/ms

Also given, $C_2(t) < C_1(t)$ {clock is slower}

$\Rightarrow [C_1(t) - C_2(t)] \times 60 = 1.2$

$\Rightarrow (1000 - t) \times 0.001 \times 60 = 1.2$

$\Rightarrow 1000 - t = 20$

$t = 980$

∴ Clock 2 ticks 980 times/ms
or 980000 times/second.

---

9)

~~a)~~ In centralized mutual exclusion algorithm take an issue on distinguishing a dead co-ordinator from permission denied, because in processes normally block after making a request, they cannot distinguish as in both cases

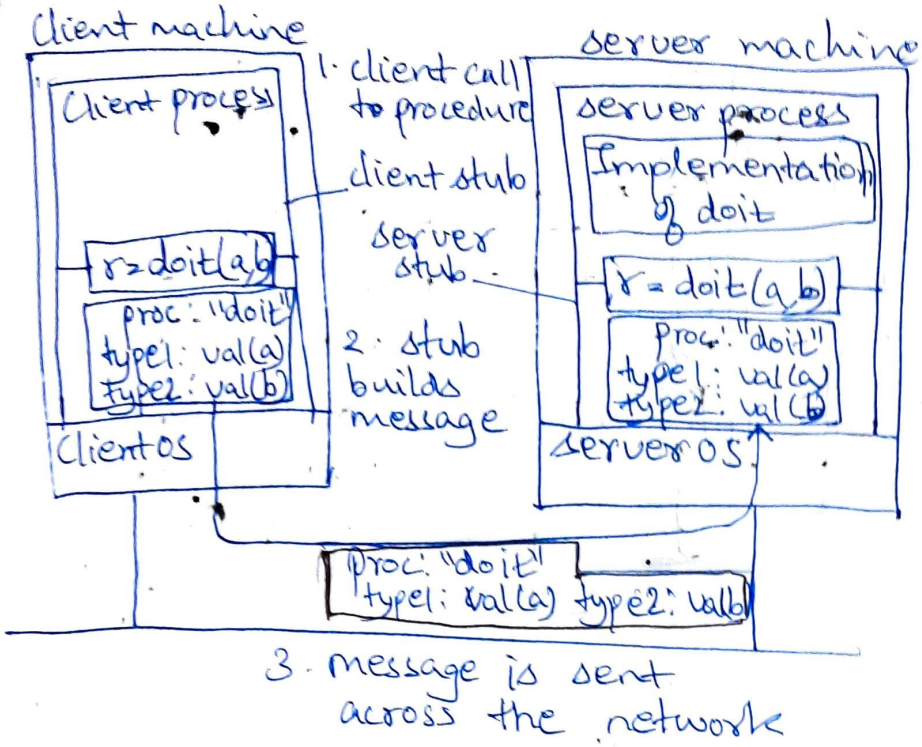
a) The centralized approach faces this shortcomming. This is because in case of both dead coordination & permission denied, no message comes back.

b) In ring algo, eventually an election message gets back to the process that started it all that process recognises this event when it receives an incoming message contains its own identifier. At this point, the message type is changed to COORDINATOR & circulated once again to inform who coordinator is & who ~~is~~ the members of the new ring are.

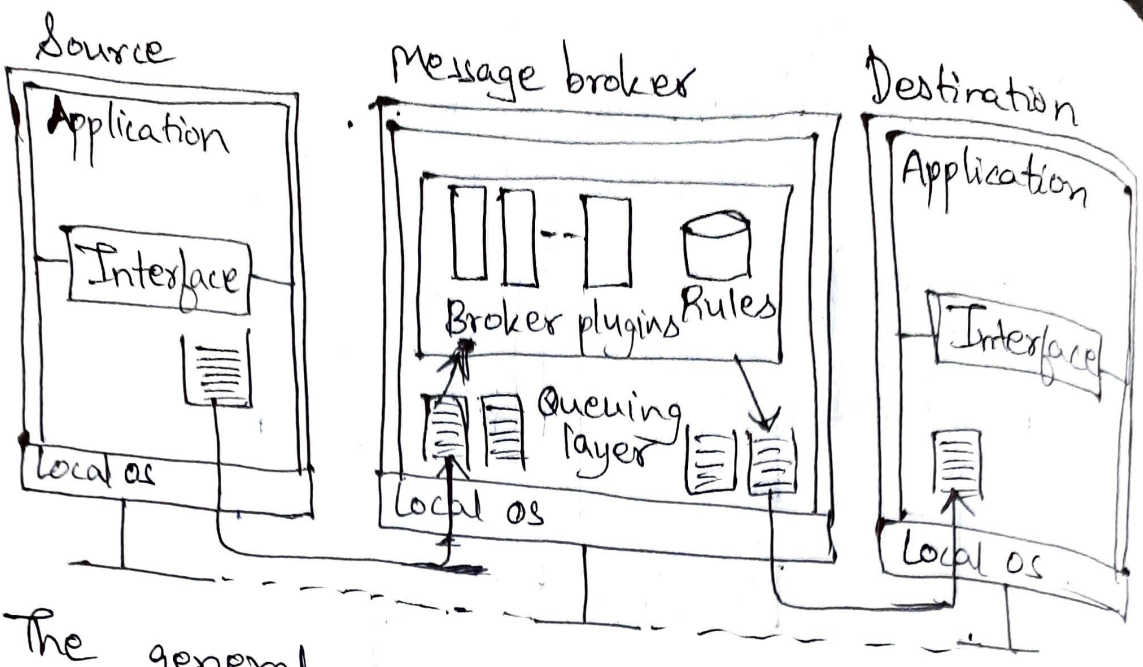
c) In a distributed algo, Lamports logical clock helps in total ordering of all events & helps eliminate ambiguity. It provides the current (logical time) to any process which ~~asks~~ wants to access a shared resource It follows total ordered multicast

d) Each token represents a repelling force by which another token is inclined to move away. The net effect is that if all tokens exert the same repulsion force, they will move away from each other & speed themselves evenly in a m-dimensional geometric space. To make all nodes holding a token learn about other tokens, we can use a gossiping protocol by which a token's force is disseminated through the network.

10)



1. The client procedure calls the client stub in the normal way
2. The client stub builds a message & calls the local OS
3. The client's os sends the message to the remote OS
4. The remote os gives the message to the server stub
5. The server stub unpacks the parameters(s) & calls the server
6. The server does the work & returns the result to the stub
7. The server stub packs the result in a message & calls its local OS
8. The server's os sends the message to the client's OS
9. The client's os gives the message to the client stub
10. The stub unpacks the result & returns it to the client.



The general organization of a message broker in a message queuing system

---

(4th ❸ Continued)

- Applying NTP symmetrically, i.e, allow B to adjust its clock to that of A.
- If B's clock is more accurate, then adjusting B's clock is meaningless. So NTP divides servers into strata.
- Server with a ref. clock such as UTC receiver or an atomic clock is known to be a stratum-1 server.
- When A contacts B, it will only adjust its time if its own stratum level is higher than that of B. Moreover after the synchronization, A's stratum level will become one higher than that of B. In other words, if B is a stratum-k server, then A will become a stratum-(k+1) server if its orginal stratum level was already larger than k.