COMPUTER ENGINEERING DEPARTMENT

ASSIGNMENT NO. 1

Subject: Distributed Computing

COURSE: B.E Year: 2021-2022 Semester: VIII
DEPT: Computer Engineering
SUBJECT CODE: CSC802 DUE DATE: 11/03/2022

Roll No.: 50 Name: Amey Mahendra Thakur

Class: BE-Comps B Date of Submission: 11/03/2022

DC Assignment - 1

Sr. No.	Questions
1	Discuss the common issues with which the designer of a heterogeneous distributed system must deal.
2	Discuss different models of middleware.
3	Discuss the call semantics of RPC.
4	Discuss the stream-oriented communication in detail.
5	Explain Berkeley Algorithm and Cristian Algorithm along with advantages and disadvantages.
6	Discuss Meakawa Algorithm with an example along with deadlock handling messages.

Student Signature:

AMEY THAKUR B-50 Amey.

Q1. Discuss the common issues with which the designer of a heterogeneous distributed system must deal.

Ans:

Common Design Issues:

① Transparency
- User should not be aware of hidden facts of the systems
- Types of transparency are:
  - Location
  - Access
  - Migration
  - Relocation
  - Replication
  - Failure
  - Concurrency
  - Persistence

② Reliability
- The system should be fault tolerant to handle failures in DS.

③ Flexibility
- The system should be flexible enough to adopt new changes. i.e. ease of enhancement.

④ Scalability
- Increasing number of devices should not hamper basic working of the system.

⑤ Performance
- The performance of DS should be at least as good as centralized system.

AMEY THAKUR B-50 Amey.

⑥ Heterogeneous Environment
- Since the DS is forming a heterogeneous environment, there is a need to break into consideration heterogeneity.

⑦ Emulation of Existing Software!
- Architecture should support all software supported by centralized system.

⑧ Security
- Data should be secured and and should be transmitted securely over the network.

AMEY THAKUR B-50 Amey.

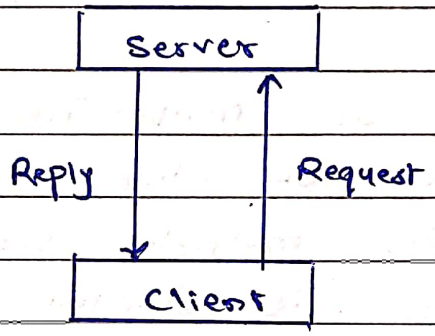
Q2. Discuss different models of middleware.

Ans:

Middleware is computer software that provides services to software applications beyond those available from the Operating system.

Models of Middleware

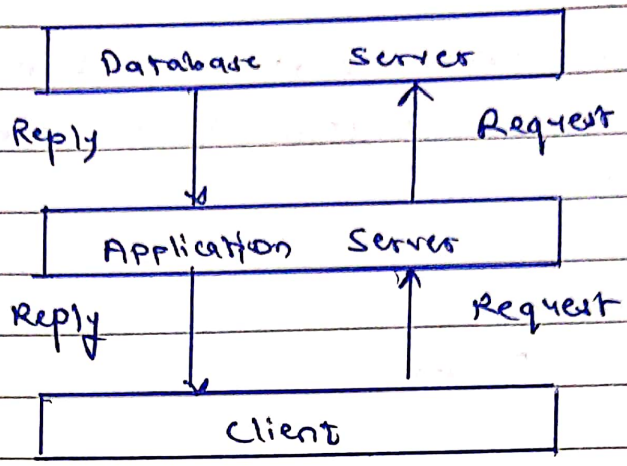
① Client Server Model



- It is the most widely used model in middleware
- It is used for communication between processes.
- In this model, one process acts as a server while all other processes acts as a client.
- Clients request for the services.
- Server provides the services to client.

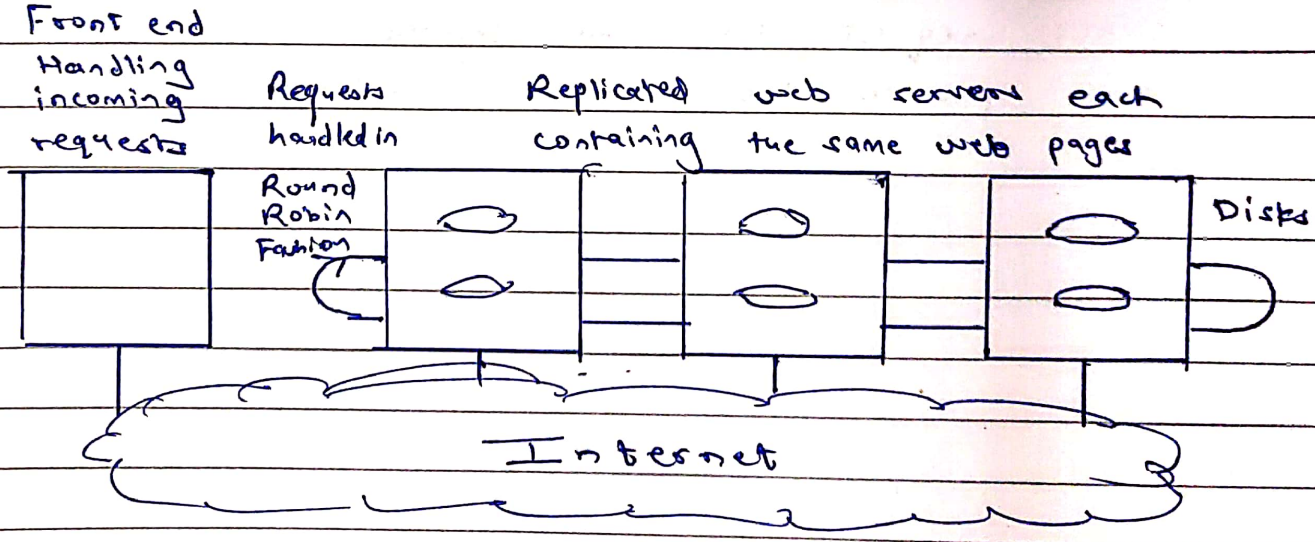
AMEY THAKUR B-50 Amey.

② Vertical Distribution Model



- It is also known as multi-Tier model.
- It is an extension of client server model.
- This model includes multiple servers
- The client request is directed to first server
- This request is proceed to next server, until the final server is reached

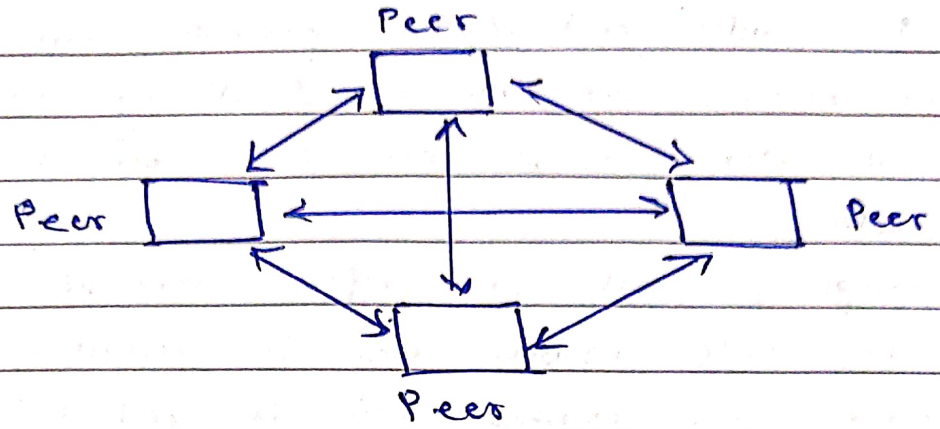
③ Horizontal Distribution model



- It is used to improve scalability and reliability
- It involves replicating a server functionality over multiple computers
- In this model, each server machine contains a complete copy of all hosted web pages and client's requests are passed on to the servers.

AMEY THAKUR B-50 Amey

④ Peer to Peer model



- It is a decentralized communication model.
- In this model, each node functions as both a client and server.
- It can request the services as well as send a respond to the request.
- Example: Collection of PCs in an office or home.

⑤ Hybrid Model

- It can be combination of any of the above models
- Examples:
  - ⓐ Super-peer networks
    - It is combination of client server and peer to peer model. Example - Bit Torrent.
  - ⓑ Edge-Server networks
    - In edge-server networks servers are placed at the edge of the internet.
    - Example - Internet service Provider (ISP)

AMEY THAKUR B-50 Amey

Q3. Discuss the call semantics of RPC.

Ans:

- In RPC, the caller and callee processes can be situated on different nodes. The normal functioning of an RPC may get disrupted due to one or more reasons mentioned below:
  1. Call message is lost or response message is lost.
  2. The callee node crashes and is restarted
  3. The caller node crashes and is restarted.
- In RPC system the call semantics determines how often the remote procedure may be executed under fault conditions
- Types of RPC call semantics:

1. May - Be Call Semantics
   - This is the weakest semantics in which a timeout mechanism is used that prevents the caller from waiting indefinitely for a response from the callee
   - This means that the caller waits until a pre-determined timeout period and then continues to execute.
   - Hence the semantics does not guarantee the receipt of call message nor the execution. This semantic is applicable where the response message is less important and applications that operate within a local network with successful transmission of messages

AMEY THAKUR B-50 Amey

(2) Last - Once call semantics

- This call semantics uses the idea of retransmitting the call message based on timeouts until the caller receives a response.
- The call, execution and result of will keep repeating until the result of procedure execution is received by the caller
- The results of the last executed call are used by the caller, hence it is known as last-one semantics.
- Last one semantics can be easily achieved only when two nodes are involved in the RPC, but it is tricky to implement it for nested RPCs and cases by orphan calls.

(3) Last - of - Many call Semantics

- This semantics neglects orphan calls unlike least-once call semantics. Orphan call is one whose caller has expired due to node crash.
- To identify each call, unique call identifiers are used which to neglect orphan calls.
- When a call is repeated, it is assigned to a new call identifier and each response message has a corresponding call identifier.
- A response is accepted only if the call identifier associated with it matches the identifier of the most recent call else it is ignored.

AMEY THAKUR B-50 Amey

④ At-least-once call Semantics

- This semantics guarantees that the call is executed one or more times but does not specify which results are returned to the caller.
- It can be implemented using timeout based retransmission without considering the orphan calls.

⑤ Exactly-Once call semantics

- This is the strongest and most desirable call semantics. It eliminates possibility of a procedure being executed more than once irrespective of the number of retransmitted call.
- The implementation of exactly-once call semantics is based on the use of timeouts, retransmission, call identifiers with same identifier for repeated calls and a reply cache associated with the callee.

AMEY THAKUR B-50 Amey

Q4. Discuss the stream oriented communication in detail.

Ans:

Stream-oriented communication

- RPC and message-oriented communication are based on the exchange of discrete messages.
- Timing might affect performance, but not correctness.
- In stream-oriented communication the message content must be delivered at a certain rate, as well as correctly
- Example: Music or Video
- Stream oriented communication supports continuous media communication.

Transmission modes in stream-oriented communication

① Asynchronous Mode
- No restrictions with respect to when data is to be delivered.

② Synchronous Mode
- Define a maximum end-to-end delay for individual data packets.

③ Isochronous Mode
- Define a maximum end-to-end delay and maximum delay variance

AMEY THAKUR B-50 Amey

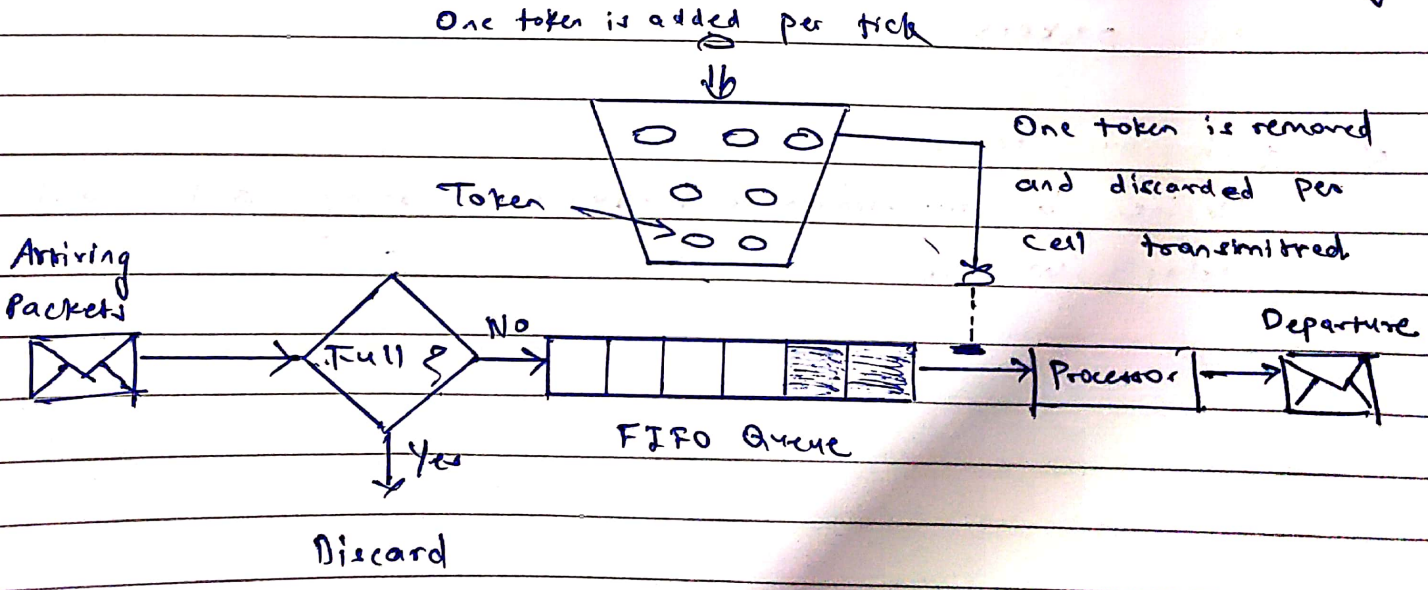
Stream:

- A data stream is a connection oriented communication facility
- It supports isochronous data transmission
- Some common stream characteristics:
  - (a) stream are unidirectional
  - (b) There is generally a single source.
- Stream Types:
  - (a) Simple: It consists of a single flow of data. (eg audio or video)
  - (b) Complex: It consists of multiple data flows (eg stereo audio or combination audio/video)

Example:

Token Bucket Algorithm

- The token bucket can be easily implemented with a counter.
- The token is initialized to zero.
- Each time a token is added, counter is incremented by 1.
- Each time a unit of data is dispatched, counter is decremented by 1.
- If the counter contains zero, the host cannot send any data.

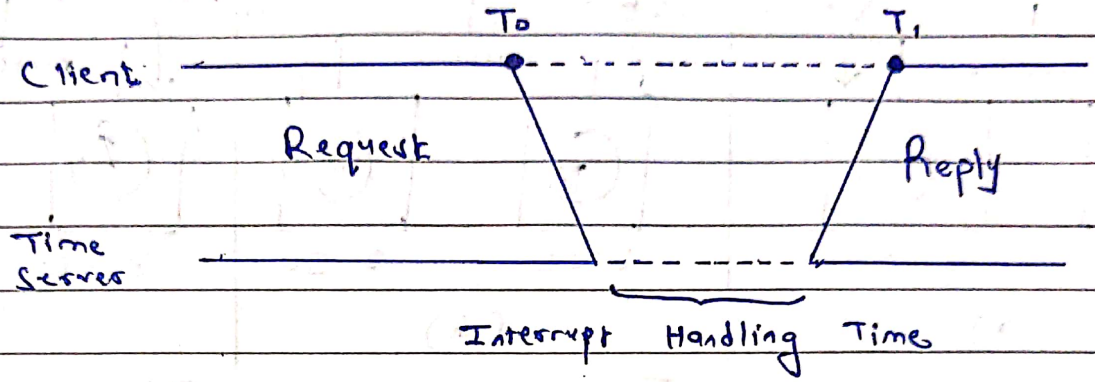


AMEY THAKUR B-50 Amey

Q5 Explain Berkeley Algorithm and Cristian Algorithm along with advantages and disadvantages.

Ans:

Cristian's Algorithm



- Cristian's algorithm is a clock synchronization algorithm 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.
- Here round trip time refers to the time duration between start of a request and end of corresponding response.
- In this method, each node periodically sends a message to server.
- When the time server receives the message it responds with a message T,

  T = Current time of server node.
- Assume the clock time of client be To when it sends the message T, when it receives the message from server
- T, and To are measured using same clock.

  $T_1 - T_0 / 2$
- When reply is received at clients node

  clock is readjusted to $T + (T_1 - T_0)/2$

Advantage:

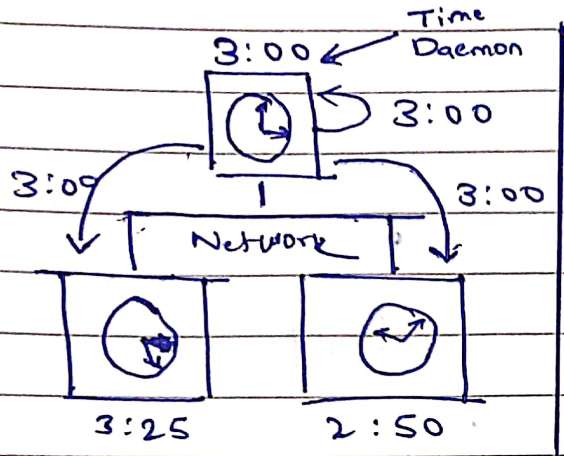
- It assumes that no additional information is available

Disadvantage:

- It restricts the number of measurements for estimating the value.

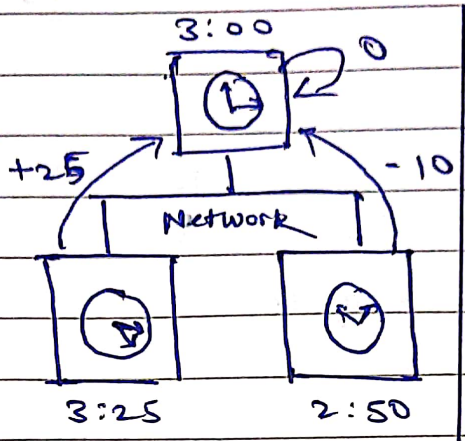
AMEY THAKUR B-50 Amey

Berkeley Algorithm:



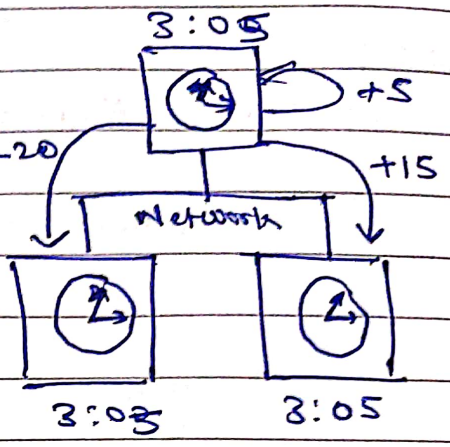
(a)

The time daemon asks the other machines for their clock values



(b)

The machine answer



(c)

The time daemon tells everyone how to adjust their clock

- The Berkeley algorithm is a method of clock synchronization in distributed computing which assumes no machine has an accurate time source.
- It is an active time server approach.
- In this approach, the time server periodically sends a message to all the computers in the group of computers.
- When this message is received each computer send back its own clock value to the time server.
- The time server has a prior knowledge of the approximate time required for propogation of a message which is used to readjust the clock values.
- It then takes average of clock values of all the computers.
- The calculated average is the current time to which all clocks should be readjusted.

- The time server readjusts its own clock to this value and instead of sending the current time to other computers it sends amount of time each computer needs for readjustment.
- This can be positive or negative value.

AMEY THAKUR B-50 Amey

Q6. Discuss Maekawa algorithm with an example along with deadlock handling message

Ans:

Maekawa Algorithm

- It is quorum based approached to ensure mutual exclusion in distributed systems.
- In this algorithm,
  1. Three types of messages (Request, Reply, Release) are used.
  2. A site send a request message to all other site in its request set or quorum to get their permission to enter critical section
  3. A site send a reply message to requesting site to give its permission to enter the critical section
  4. A site send a release message to all other site in its request set or quorum upon exiting the critical section

To enter critical section.

- When a site S_i wants to enter the critical section, it sends a request message REQUEST (i) to all other sites in the request set R_i.
- When a site S_j receives the request message REQUEST (i) from site S_i, it returns a REPLY message to site S_i if it has not sent a REPLY message to site from time it received the last RELEASE message. Otherwise, it queues up the requests.

AMEY THAKUR B-50 Amey.

To execute the critical section

- A site S_i can enter the critical section if it has received the REPLY message from all the site in request set R_i.

To release the critical section

- When a site S_i exits the critical section, it sends RELEASE (i) message to all other sites in request set R_i.
- When a site S_j receives the RELEASE (i) message from site S_i, it send REPLY message to the next site waiting in the queue and deletes that entry from the queue.
- In case queue is empty, site S_j updates its status to show that it has not sent any REPLY message since the receipt of the last release message

Message Complexity.

- Maekawa's algorithm requires invocation of $3\sqrt{N}$ messages per critical section executing as the size of a request set is $\sqrt{N}$. These $3\sqrt{N}$ message involves
  - → $\sqrt{N}$ request message
  - → $\sqrt{N}$ reply message
  - → $\sqrt{N}$ release message

AMEY THAKUR B-50 Amey.

Handling Deadlocks

- Maekawa's algorithm handles deadlocks by requiring a site to yield a lock if the timestamp of its request is larger than the timestamp of some other request waiting for the same lock.
- A site suspects a deadlock whenever a higher priority request arrives and waits at a site because the site has sent a REPLY message to a lower priority request.

Deadlock handing requires three types of messages

1. FAILED
2. INQUIRE
3. YIELD