APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

STUDY MATERIALS

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSIT3Y

SEVENTH SEMESTER B.TECH DEGREE EXAMINATION, DECEMBER 2018

Course Code : CS407

Course Name : DISTRIBUTED COMPUTING

MODULES (1 2 3)

PART A (Each carries 4 marks)

1.List any 4 issues in the design of a distributed system (4)

- Transparency
- Heterogeneity
- Openness
- Scalability

3.Explain the key techniques used for indirect communication (4)

Group communication – Group communication is concerned with the delivery of messages to a set of recipients and hence it is a multiparty communication paradigm supporting one to many communication.

Publisher/subscribe – In this pattern, senders of messages called publishers, do not program the messages to be sent directly to specific receivers called subscribers.

Message queues/forwarding – In this model, producer processes can send messages to a specified queue and consumer processes can receive messages from the queue.

Tuple space – Tuple space is a collection of ordered sequences of data called tuples. It is an associative memory.

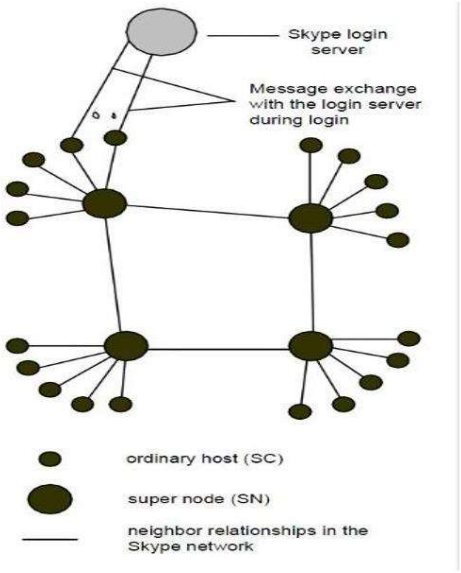
4.Why Skype is called an Overlay network?

SKYPE : an example of an overlay network.

- Skype is a virtual network that establishes connection between people.(Skype subscribers who are currently active.)
- No IP address or port is required to establish a call.
- The architecture of the VN supporting Skype is not widely publicized ,but researchers have studied Skype through variety of methods, including traffic analysis of it.

- Skype architecture: peer to peer architecture consisting of ordinary nodes and super nodes.

- **Skype Client (SC):** Each SC needs to build and refresh a table of reachable nodes. In Skype, this table is called host cache (HC) and it contains IP address and port number of super nodes.
- **Super Node (SN):** Super nodes are the end points where Skype clients connect to. Any node with a public IP address having sufficient CPU, memory, and network bandwidth is a candidate to become a super node . If a SC cannot establish a TCP connection with a SN then it will report a login failure.



- **User connection**: User's authenticated via login servers.
- **Search for users :** super nodes to search of the global index of users distributed across the super nodes.

  on average, 8 super nodes are contacted.
- **Voice connection**: TCP for signalling call request and termination and UDP or TCP for streaming audio.

6.What is "send_group" group communication primitive referred over "send" primitive?

• In one to one and one to many send or has to specify destination address and pointer to data so some send primitives can be used for both.

• But some systems use send group primitives because

1) It simplifies design and implementation
2) It provides greater flexibilities.
3) Can include extra parameters to specify degree of reliability or atomicity
4)Indicates whether to use name server or group server

10.Define mobile agents. How can they be potential security threat?

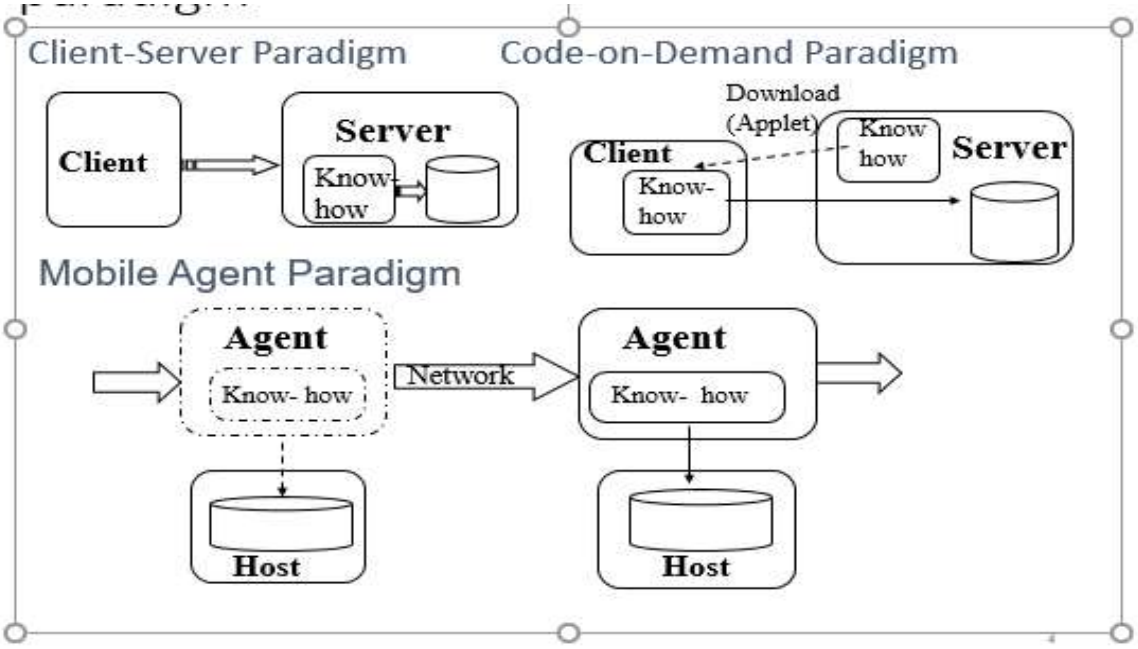
Mobile agents

- Mobile agents are defined as active objects (or clusters of objects) that have behavior, state and location.
  - **Mobility:** *Agents* that can travel in network
  - **Autonomy:** Agent itself decides when and where to migrate next
  - Examples of the tasks that can be done by mobile agents are:

To collect information.

To install and maintain software maintained on the computers within an organization.

To compare the prices of products from a number of vendors.



- Mobile agents are a potential security threat to the resources in computers that they visit.
- The environment receiving a mobile agent should decide on which of the local resources to be allowed to use.
- Mobile agents themselves can be vulnerable

They may not be able to complete their task if they are refused access to the information they need.

PART B

11. a) What are the two variants of the interaction model in distributed systems? On what points do they differ?

Synchronous communication

Asynchronous communication

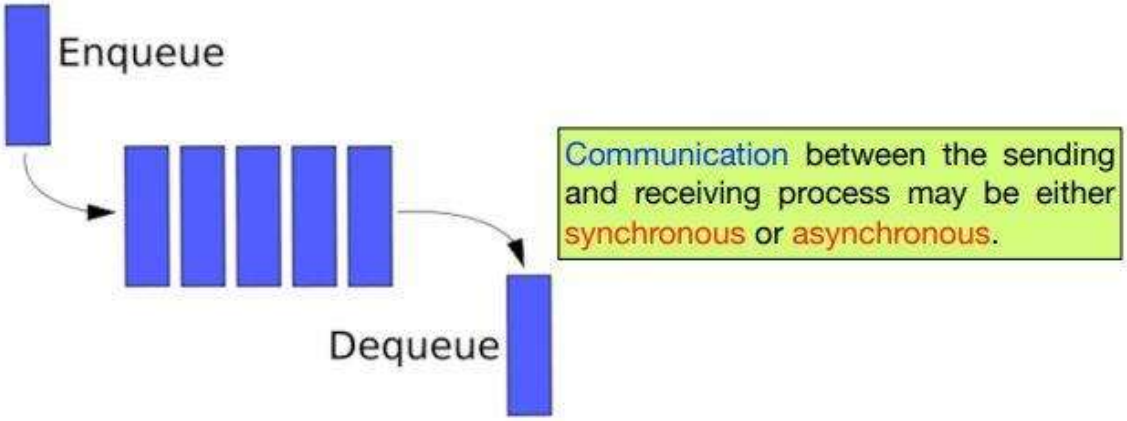
- Message passing between a pair of processes can be supported by two message communication operations, *send* and *receive*, defined in terms of destinations and messages.
- To communicate, one process sends a message (a sequence of bytes) to a destination and another process at the destination receives the message.

- This activity involves the communication of data from the sending process to the receiving process and may involve the synchronization of the two processes.

Synchronous and asynchronous communication

Sending VS Receiving

- A queue is associated with each message destination.
- Sending processes cause messages to be added to *remote* queues.
- Receiving processes remove messages from *local* queues.



Synchronous Communication

- The sending and receiving processes synchronize at every message.
- In this case, both send and receive are *blocking operations*:
  - whenever a send is issued the sending process is *blocked* until the corresponding receive is issued;
  - whenever a receive is issued the receiving process *blocks* until a message arrives.

Asynchronous Communication

- The send operation is *non-blocking*:
  - the sending process is allowed to proceed as soon as the message has been copied to a local buffer;
  - the transmission of the message proceeds in parallel with the sending process.
- The receive operation can have *blocking* and *non-blocking* variants:
  - [non-blocking] the receiving process proceeds with its program after issuing a receive operation;
  - [blocking] receiving process blocks until a message arrives.

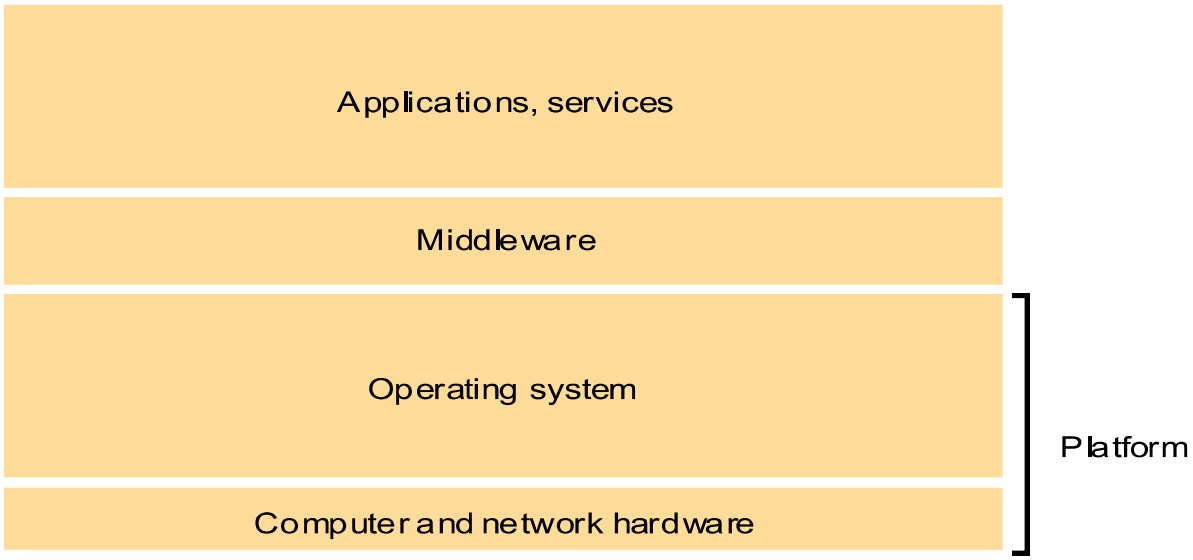
b) Describe any 4 key architectural patterns used in distributed systems

- Architectural patterns in distributed system includes

1. Layering
2. Tiered architectures
3. concept of thin client

1.**Layering**

- In a layered approach, a complex system is partitioned into a number of layers. Each layer offers software abstraction, below and above layers are unaware of implementation details.
- Software and hardware service layers in distributed system.



- **Platform**:
- A platform for distributed systems and application consists of the lowest-level hardware and software layers.
- These low-level layers provide services to the layers above them, which are implemented independently in each computer, bringing the system's programming interface up to a level that facilitates communication and coordination between processes.
- Examples: Intel x86/Windows, Intel x86/Linux
- **Middleware:**
- A layer of software whose purpose is to mask heterogeneity present in distributed systems and to provide a convenient programming model to application developers.
- Major Examples:

  IBM WebSphere

  Microsoft .NET

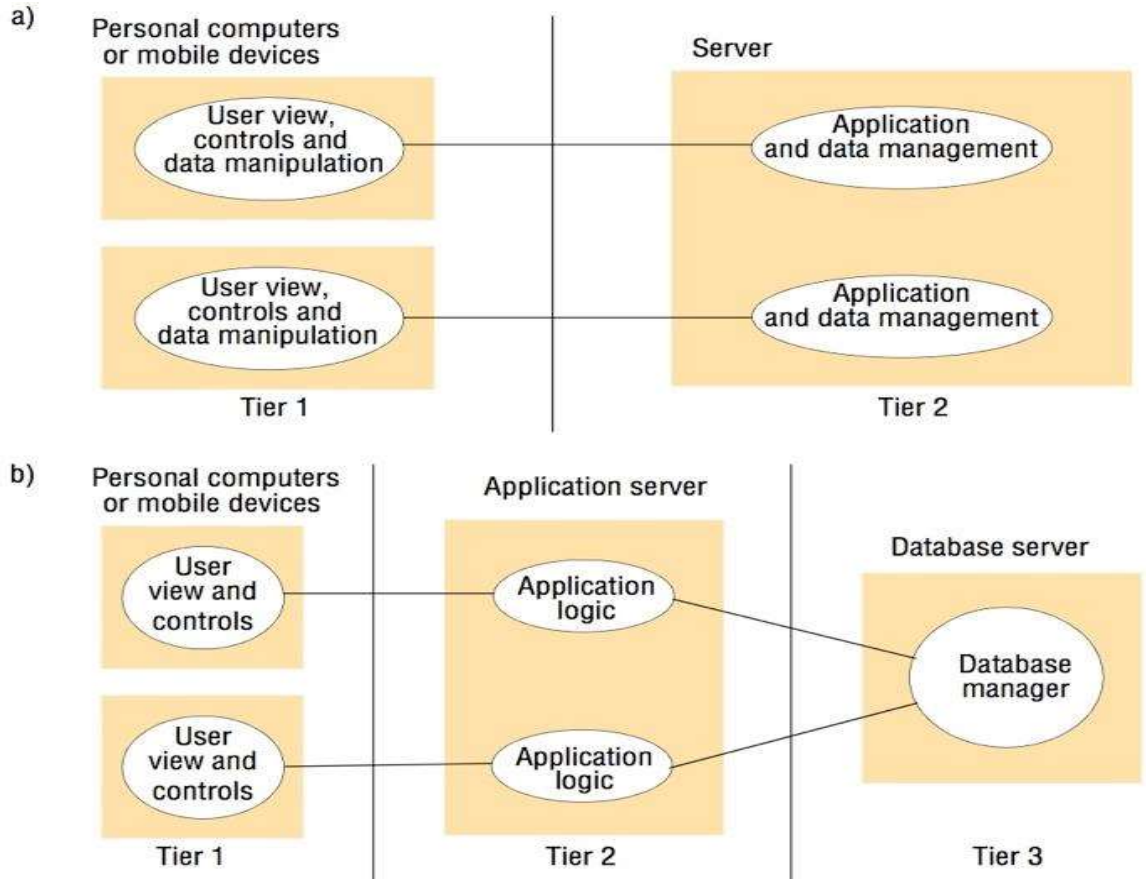
  Sun J2EE
- Helps providing useful building blocks for the construction of software components that can work with one another in a distributed system.
- It raises the level of the communication activities of application programs through the support of abstractions such as RMI, communication between a group of processes, replication of shared data objects etc.

2.**Tired Architectures**

Tiering is a technique to organize functionality of a given layer and place this functionality onto appropriate servers and to other physical nodes

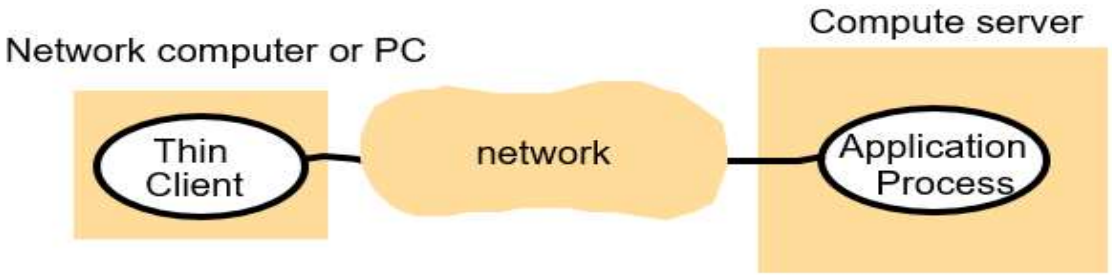
Two tier architecture

Three tier architecture



3.**Thin Client**

- A trend in distributed computing is towards moving complexity away from the end user device towards services in the internet.
- The term thin client refers to a software layer that supports a window-based user interface that is local to the user while executing application programs on a remote computer.
- A thin client is a stateless desktop terminal that has no hard drive. All features typically found on the desktop PC, including applications, sensitive data, memory, etc., are stored back in the data center when using a thin client.



12.a) List and explain the different types of communication paradigms used within distributes systems.

Three types of communication paradigms include:

1.Interprocess communication

2.Remote invocation.

3.Indirect communication

1.Interprocess communication

Is a set of programming interfaces that allow a programmer to coordinate activities among different program processes in a distributed system.

IPC refers to relatively low level support for communication between processes in distributed system.

Examples :

Message passing primitives(send(destination,&msg),Receive(source,&msg)),socket programming.

2.Remote invocation

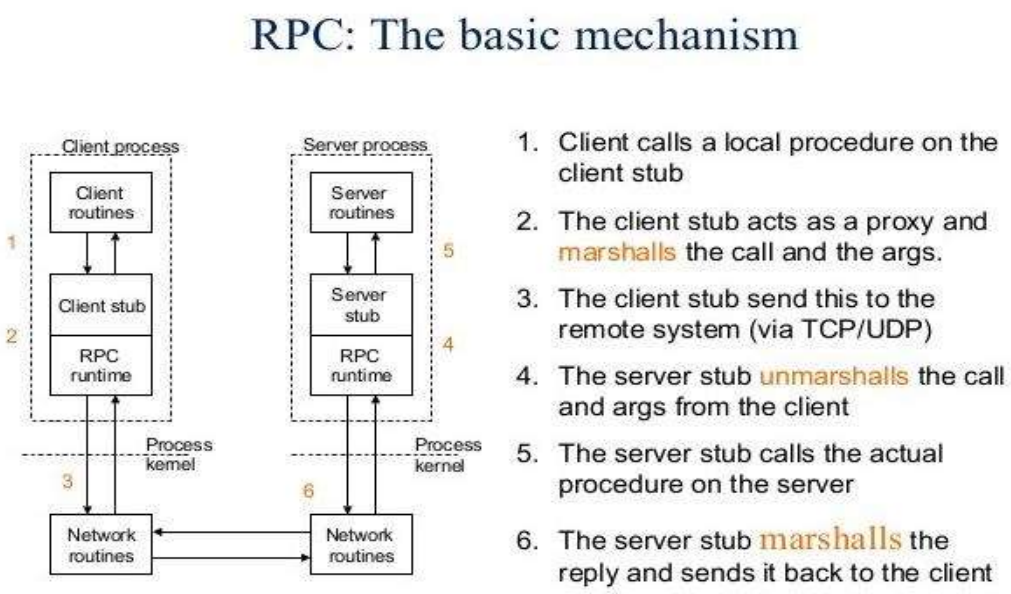
Most common communication paradigm covering a range of techniques based on a two way exchange between communicating entities in the distributed systems, and resulting in the calling of a remote operation, procedure or method eg: RPC

Request Reply Protocol

-are effectively a pattern imposed on an underlying message-passing service to support client-server computing.

RRP is used with HTTP protocol.

Both RPC and RMI approaches are supported by underlying request-reply exchanges.



3.Indirect communication

-is defined as communication between entities in a distributed system through an intermediary with no direct coupling between the sender and receiver.

Sender -> Middleman -> Receiver

Group communication –

Group communication is concerned with the delivery of messages to a set of recipients and hence it is a multiparty communication paradigm supporting one to many communication.

Each group in the system is represented by a group identifier

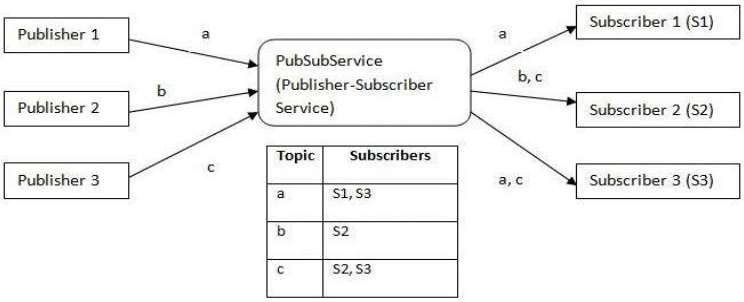
Recipients elect to receive messages sent to a group via group identifier.

Senders then send messages to the group via group identifier and hence do not need to know the recipient of the message.

Publisher/subscribe – In this pattern, senders of messages called publishers, do not program the messages to be sent directly to specific receivers called subscribers.

- The programmer “publishes” messages (events), without any knowledge of any subscribers there may be.
- Similarly, subscribers express interest in one or more events, and only receive messages that are of interest, without any knowledge of any publishers.
- To accomplish this, an intermediary, called a “message broker” or “event bus”, receives published messages, and then forwards them on to those subscribers who are registered to receive them.

- Publish-subscribe systems:



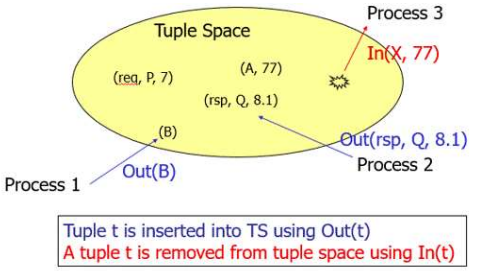
- **Message Publisher** sends messages to the PubSub Service without any knowledge of message subscribers
- **Message Subscribers** will only get the messages for which topics they are registered with the PubSubService. **E.g.** say we have three different topics (message types): a, b, c but only topic a is of interest to Subscriber 1, topics b and c are of interest to Subscriber 2 and Subscriber 3 wants messages for topics a and c. So subscriber 1 will be notified for topic a, Subscriber 2 will be notified for topics b and c, and Subscriber 3 will be notified for topics a and c by the PubSub Service.
- **Publishers** tag each message with a topic, and send it to the **PubSubService which acts like a middleman** between Publishers and receivers.
- **Subscriber** registers itself with PubSubService (middleman) and tells that it's interested in messages related to a particular topic.
- Publisher-Subscriber pattern is highly **loosely coupled architecture** where the publishers don't know who the subscribers are and subscribers don't know who the publishers of topic are.

Message queues/forwarding –

- Message queues:
- Publish-subscribe systems offer a one-to-many style of communication, message queues offer a point-to-point service.
- In messages queue model, producer processes can send messages to a specified queue and consumer processes can receive messages from the queue or be notified of the arrival of new messages in the queue.
- Queue offer an indirection between the producer and consumer processes.

Tuple space –

Communication paradigms: Indirect communication techniques



- Tuple space is an associative memory.
- Tuple space is a collection of ordered sequences of data called tuples.
- Each field of a tuple contains data in the form of one of the valid types (including aggregates like arrays, structures etc.)
- Each process can then find a tuple by expressing pattern of interest and remove/read it from the tuple space.
- Tuple space can be viewed as a long term data memory - once installed, tuples remain in tuple space until they are explicitly removed by some process. Thus, processes can interact through time as well as space , since the producer and consumer of a tuple need never coexist simultaneously.

12 b) A distributed system is defined as one in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages, what are the consequences of defining a distributed system in this manner?

1.Web Search

- The web search engine is a software to search information on the world wide web(WWW).
- This a very complex task, as current estimates state that the web consists of over 63 billion pages and one trillion unique web addresses.
- When you Google something ,behind the scene many computers communicates and co-ordinates to provide you an answer.
- But the user perceives the distributed system as a single entity rather than as a collection of autonomous systems, which are cooperating.
- The users are unaware of where the services are located and data transferring from a local machine to a remote and vice versa.

2.Massively multiplayer online games (MMOGs)

- A massively multiplayer online game is an online game with large numbers of players, typically from hundreds to thousands.
- Previously, client server architecture was used for gaming.
- This architecture has the advantage that a single authority orders events, resolves conflicts in the simulation , and act as a central repository for data and is easy to secure.
- Issues:

- First, it introduces delay because message between players are forwarded through servers.
- Second, traffic at the server increases with the number of players, creating localized congestion.
- Third, the size of the virtual world and the quantity the data it contains is limited by the storage capacity of the server.
- Another distributed architecture to overcome these problems:
- Peer-to-peer : Players can send messages directly to each other, thereby reducing delay and eliminating localized congestion

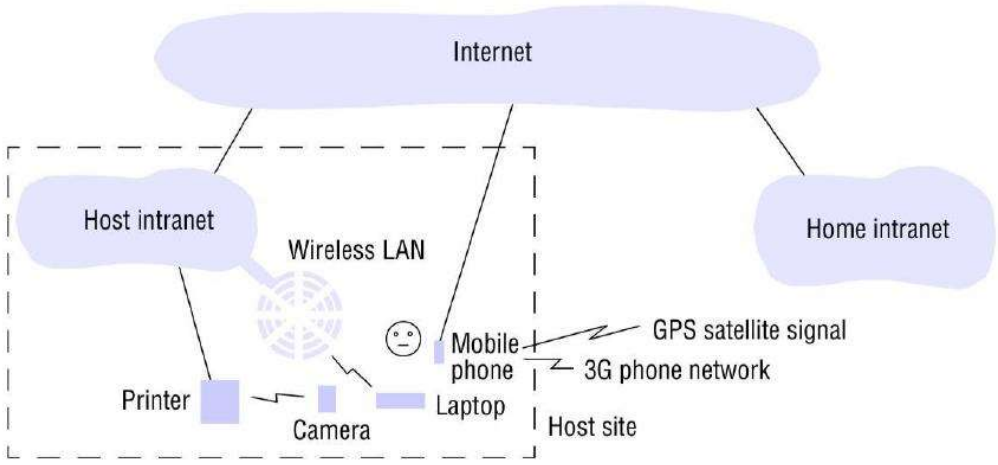
13.a) Write notes on mobile and ubiquitous computing. (4)

The portability of many of these devices, together with their ability to connect conveniently to networks in different places, makes *mobile computing* possible. Mobile computing is the performance of computing tasks while the user is on the move, or visiting places other than their usual environment. Mobility introduces a number of challenges for distributed systems, including the need to deal with variable connectivity and indeed disconnection, and the need to maintain operation in the face of device mobility.

Ubiquitous computing is the harnessing of many small, cheap computational devices that are present in users' physical environments, including the home, office and even natural settings. The term 'ubiquitous' is intended to suggest that small computing devices will eventually become so pervasive in everyday objects that they are scarcely noticed. That is, their computational behaviour will be transparently and intimately tied up with their physical function.

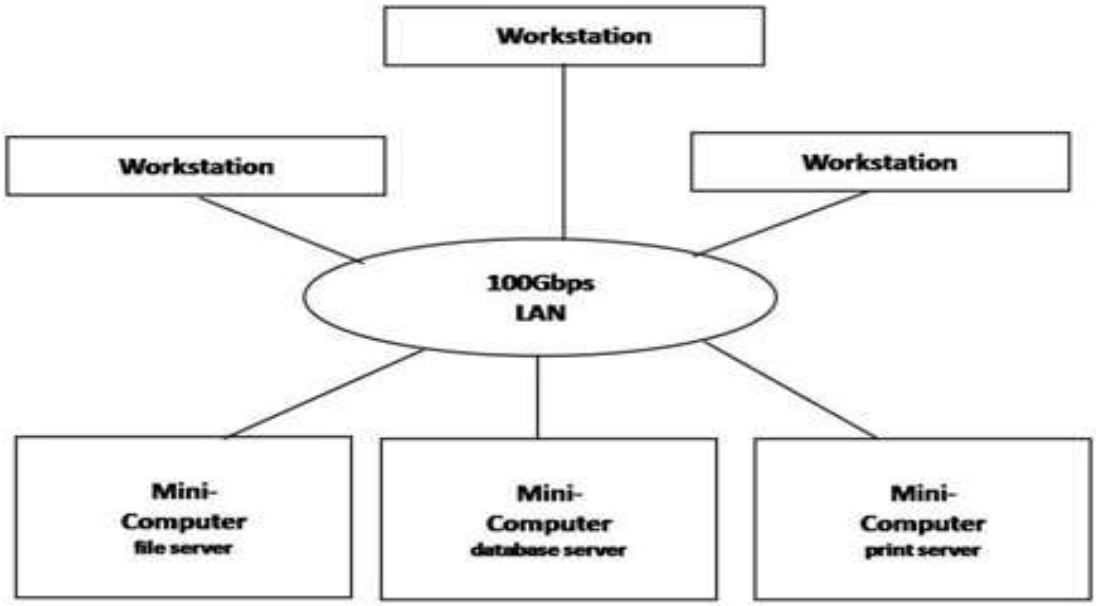
The presence of computers everywhere only becomes useful when they can communicate with one another. For example, it may be convenient for users to control their washing machine or their entertainment system from their phone or a 'universal remote control' device in the homee.

Figure below shows a user who is visiting a host organization. The figure shows the user's home intranet and the host intranet at the site that the user is visiting. Both intranets are connected to the rest of the Internet.



13.b) Compare workstation server model with process pool model?

Workstation Server Model:



- This model consists of a few minicomputers ad several ws interconnected by a communication network.
- Most of the ws in this model are diskless ws and a few of which are diskful.
- Minicomputers are used for implementing the file system and other type of services such as database service ad print service.

WORKING :

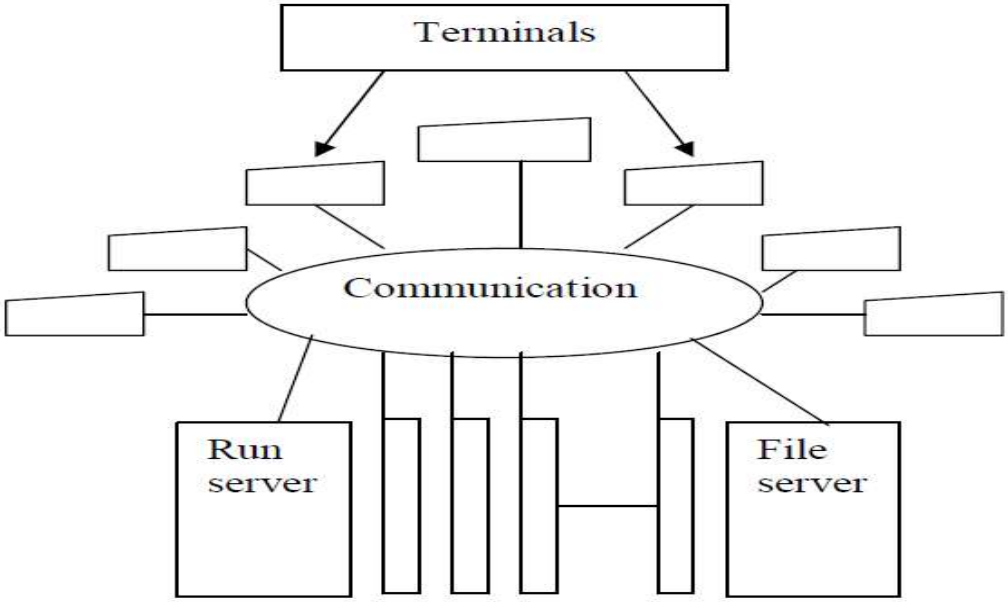
- User can perform normal computation activities at his home ws.

- Request for services are provided by servers.(client server model)
- For better performance local disk of diskful ws can be used as storage of shared files.
- Advantages:

Faster response(servers)

Diskless workstations . Softwares need to install only on servers.

Processor Pool model:



- Consists of multiple processors: a pool of processors and a group of terminals
- This model has no concept of home work station.
- The processors are powerful microcomputers and minicomputers.
- A specialized server called Run Server(RS) handles the scheduling of the processors (P1, P2… PN-1, PN)
- If a user logs into his/her terminal and tries to execute a parallel program on X number of computers.
- Run server will allot X number of processors to the job. They will be returned to the pool after the parallel job completes.
- The run server decides on which processor the job will run.

If one of the processors in the pool fails, the run server allocates another processor in the pool

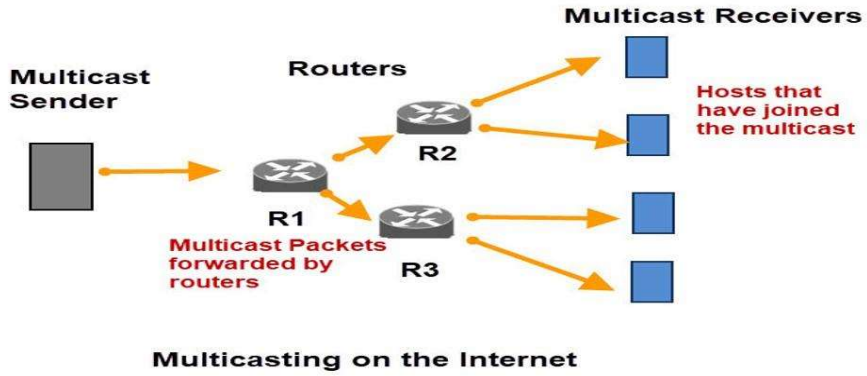
PART C

14 a) Describe IP multicast in detail

IP Multicast

- **Multicast group**: The set of receivers for a multicast transmission is called multicast group.
- Multicast group is identified by **multicast addresses**.
- Multicast addresses are in the range 224.0.0.0 to 239.255.255.255.
- i.e **class D internet address.**
- The membership of multicast groups is dynamic.
- Since TCP supports only the unicast mode, at the application programming level, IP multicast is available via UDP.

IP multicast refers to the implementation of multicast communication in internet.



Multicasting on the Internet

- The following details are specific to IPv4:
  - **Multicast routers** – IP packets can be multicast on a local network and on the Internet.
    - The IP multicast routing protocol uses the **Time To Live (TTL)** field of IP datagrams to decide how "far" from a sending host a given multicast packet should be forwarded.
    - The default TTL for multicast datagrams is 1, which will result in multicast packets going only to other hosts on the local network.
    - Local multicasts use the multicast capability of the local network .
    - Multicast in the Internet make use of **multicast routers**.

As the values of the TTL field increase, routers will expand the number of hops they will forward a multicast packet. To provide meaningful scope control, multicast routers enforce the following "thresholds" on forwarding based on the TTL field:

0
restricted to the same host
1
restricted to the same subnet
32
restricted to the same site
64
restricted to the same region
128
restricted to the same continent
255
unrestricted

- **Multicast address allocation** – Multicast address can be permanent or temporary:
  - 224.0.0.1 to 244.0.0.255 are assigned to be permanent multicast addresses. i.e. reserved for use of routing protocols and other topology discovery or maintenance protocol.
  - A temporary multicast group requires a free multicast address to avoid accidental participation in an existing group before use and cease after use.
  - A client-server solution is adopted whereby clients request a multicast address from a multicast address allocation server(MAAS), which must then communicate across domains to ensure allocations are unique for a given life time and scope.

b) Give notes on failure model for multicast datagrams

Failure model for multicast datagrams

- Same failure characteristics as UDP.
- suffer from omission failures: it does not guarantee that a message will be delivered to any member of a group.
- This is a unreliable multicast.

RELIABILITY and Ordering of multicast

- The following situations could occur:
  - **Omission failure** – A recipient could drop a message because its buffer is full. A datagram sent from on multicast router to another might be lost.
  - **Process failure** – If a multicast router fails, the group members beyond that router won't receive the message.
  - **Ordering issue** – The packets could arrive in a different order.

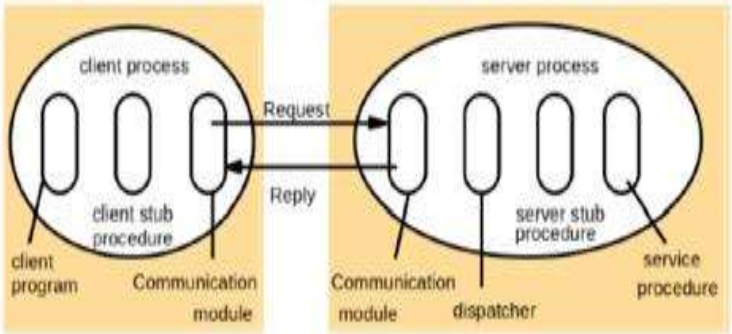
- The effects of reliability and ordering:
  - **Fault tolerance based on replicated services** – In replicated servers, multicast requires that either all or none should receive the request to perform an operation to remain consistent.
  - **Finding the discovery servers in spontaneous networking** – An occasional lost request is not an issue when locating a discovery server.
  - **Better performance through replicated data** – The effect of lost messages and inconsistent ordering depends on the method of replication.
  - **Propagation of event notifications** – The particular application determines the qualities required of multicast.

- These examples suggest that some applications require a more reliable multicast.
  - **Reliable multicast** – Any message transmitted is either received by all members of a group or by none of them.
  - **Totally ordered multicast** – All of the messages transmitted to a group reach all of the members in the same order.

15 a) Explain the implementation of RPC mechanism with a neat diagram

5.3.2 Implementation of RPC

Figure 5.10 Role of client and server stub procedures in RPC



Remote Procedure Call

- **Remote Procedure Call (RPC)** is a protocol that allows programs to call procedures located on other machines.
- RPC uses the client/server model. The requesting program is a client and the service-providing program is the server.
- The client stub acts as a proxy for the remote procedure.
- The server stub acts as a correspondent to the client stub.

The following steps take place during an RPC: Working

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*** 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(**communication module**)transport layer, which sends it **to the remote server machine**.

4. On the server , **the dispatcher** selects one of the server stub procedures according to the procedure identifier in the request message.

5. ***The server stub procedure***, then **unmarshalls** the parameters and calls the desired server routine (**server procedure**) 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 **communication module** (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. The client stub demarshalls the return parameters and execution returns to the caller.**

The client and server stub procedures and the dispatcher can be generated automatically by an interface compiler from the interface definition of the service.

SOLVED QUESTIONS – MODULE 6

Regular Paper

18b) Write an algorithm to implement mutual exclusion between N processes that is based upon multicast and logical clocks. Illustrate the algorithm using the situation involving three processes p1,p2, p3.

Ans) Ricart and Agrawala [1981] developed an algorithm to implement Mutual exclusion between N peer processes based upon multicast.

• Processes that require entry to a critical section multicast a request message, and can enter it only when all the other processes have replied to this message.

• The condition under which a process replies to a request are designed to ensure ME1, ME2 and ME3 are met.

• Each process pi keeps a Lamport clock. Message requesting entry are of the form<T, pi>.

• Each process records its state of either RELEASE, WANTED or HELD in a variable state.

• If a process requests entry and all other processes is RELEASED, then all processes reply immediately.

• If some process is in state HELD, then that process will not reply until it is finished.

• If some process is in state WANTED and has a smaller timestamp than the incoming request, it will queue the request until it is finished.

• If two or more processes request entry at the same time, then whichever bears the lowest timestamp will be the first to collect N-1 replies.

ALGORITHM

On initialization

state := RELEASED;

To enter the section

state := WANTED;

Multicast request to all processes; //**request processing deferred here**

T := request's timestamp;

Wait until (number of replies received = (*N* – 1));

state := HELD;

On receipt of a request <Ti, pi> at pj(i ≠ j)

if (state = HELD or *(state =* WANTED *and (T, pj) < (Ti, pi)))*

then

queue *request* from *pi* without replying;

else

reply immediately to *pi*;

end if

To exit the critical section

state := RELEASED;

reply to any queued requests;

Algorithm by Ricart and Agrawala

• Satisfies ME1

  • if it were possible for two processes pi and pj (i ≠ j) to enter the CS at the same time, then both of those processes would have to have replied to the other.

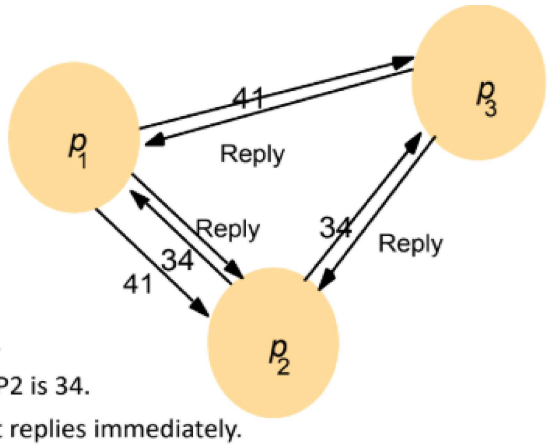
  • But since the pairs <Ti, pi> are totally ordered, this is impossible

• Satisfies ME2 and ME3

• Bandwidth consumption is high (Multicast).

• Client delay is again 1 round trip time

• Throughput: Synchronization delay is only one message transmission time. Both the previous algorithms incurred a round-trip synchronization delay.



- P1 and P2 request CS concurrently.
- The timestamp of P1 is 41 and for P2 is 34.
- When P3 receives their requests, it replies immediately.
- When P2 receives P1's request, it finds its own request has the lower timestamp, and so does not reply, holding P1 request in queue.
- However, P1 will reply. P2 will enter CS. After P2 finishes, P2 reply P1 and P1 will enter CS.
- Granting entry takes 2(N-1) messages, N-1 to multicast request and N-1 replies.

19) With an example and suitable figure describe the operation of bully algorithm. Justify whether it meets the requirements of election, during run of the algorithm. Also evaluate the performance of the above algorithm.

Ans) The bully algorithm devised by Garcia-Molina (1982).

Algorithm assumptions:

1. Message delivery between processes is reliable.
2. This algorithm assumes that the system is synchronous: it uses timeouts to detect a process failure.
3. each process has a unique ID
4. The bully algorithm assumes that each process knows which processes have higher identifiers, and that it can communicate with all such processes.

Goal

Find the non-crashed process with the highest ID

Three types of messages

- an **election message** is sent to announce an election.
- an **answer message** is sent in response to an election message .
- a **coordinator message** is sent to announce the identity of the elected process- the new coordinator.

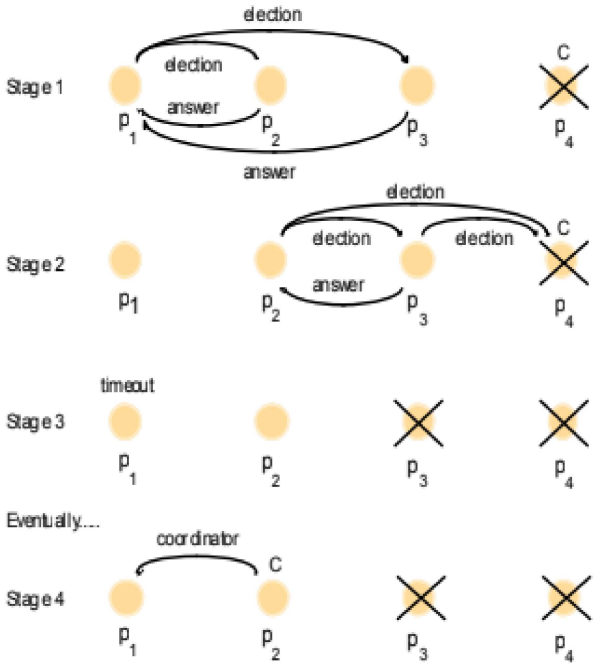
Explanation

- A process begins an election when it notices, through timeouts, that the coordinator has failed.
- Several processes may discover this concurrently– Need consistent result.
- Since the system is synchronous, we can construct a reliable failure detector.
- There is a maximum message transmission delay, Ttrans , and a maximum delay for processing a message Tprocess .
- Therefore, we can calculate a time T = 2Ttrans + Tprocess which is an upper bound on the time that can elapse between sending a message to another process and receiving a response.

- If no response arrives within time T, then the local failure detector can report that the intended recipient of the request has failed.
- The process that knows it has the highest identifier can elect itself as simply by sending a coordinator message to all processes with lower identifiers.
- A process with a lower identifier can begin an election by sending an election message to those processes that have a higher identifier and awaiting answer messages in response.
- If none arrives within time T, the process considers itself the coordinator and sends a coordinator message to all processes with lower identifiers announcing this.
- Otherwise, the process waits a further period Tc for a coordinator message to arrive from the new coordinator.
- If a process pi receives a coordinator message, it sets its variable elected(i) to the identifier of the coordinator contained within it and treats that process as the coordinator.
- If a process receives an election message, it sends back an answer message and begins another election – unless it has begun one already.

Example

1. There are four processes, *p*1 –*p*4 .
2. Process *p*1 detects the failure of the coordinator *p*4 and announces an election **(stage 1).**
3. On receiving an *election* message from *p*1 , processes *p*2 and *p*3 send *answer* messages to *p*1 and begin their own elections;
4. *p*3 sends an *answer* message to *p*2 , but *p*3 receives no *answer* message from the failed process *p*4 **(stage 2).**
5. p3 therefore decides that it is the coordinator. But before it can send out the *coordinator* message, it too fails **(stage 3).**
6. When *p*1 's timeout period *T'* expires (which we assume occurs before *p*2 's timeout expires), it deduces the absence of a *coordinator* message and begins another election.
7. Eventually, *p*2 is elected coordinator **(stage 4).**



The election of coordinator p_2, after the failure of p_4 and then p_3

Justification

1. **Liveness Condition**
   - This algorithm clearly meets the liveness condition E2, by the assumption of reliable message delivery.
   - It is impossible for two processes to decide that they are the coordinator, since the process with the lower identifier will discover that the other exists and defer to it.

2. **Safety Condition**
   - The algorithm is not guaranteed to meet the safety condition E1 if processes that have crashed are replaced by processes with the same identifiers.
   - Example : p3 had failed but was then replaced. Just as p2 sends its coordinator message, p3 does the same. p2 receives p3 ‘s coordinator message after it has sent its own and so sets elected(2)= p3 . Due to variable message transmission delays, p1 receives p2 ’s coordinator message after p3 ’s and so eventually sets elected(1) = p2 . Condition E1 has been broken.

Performance Analysis

Best case

The node with second highest identifier detects failure

Total messages = N-2

- One message for each of the other processes indicating the process with the second highest identifier is the new coordinator.

Worst case

The node with lowest identifier detects failure. This causes N-1 processes to initiate the election algorithm each sending messages to processes with higher identifiers.

Total messages = $O(N^2)$

try it now

A KTU
STUDENTS
PLATFORM

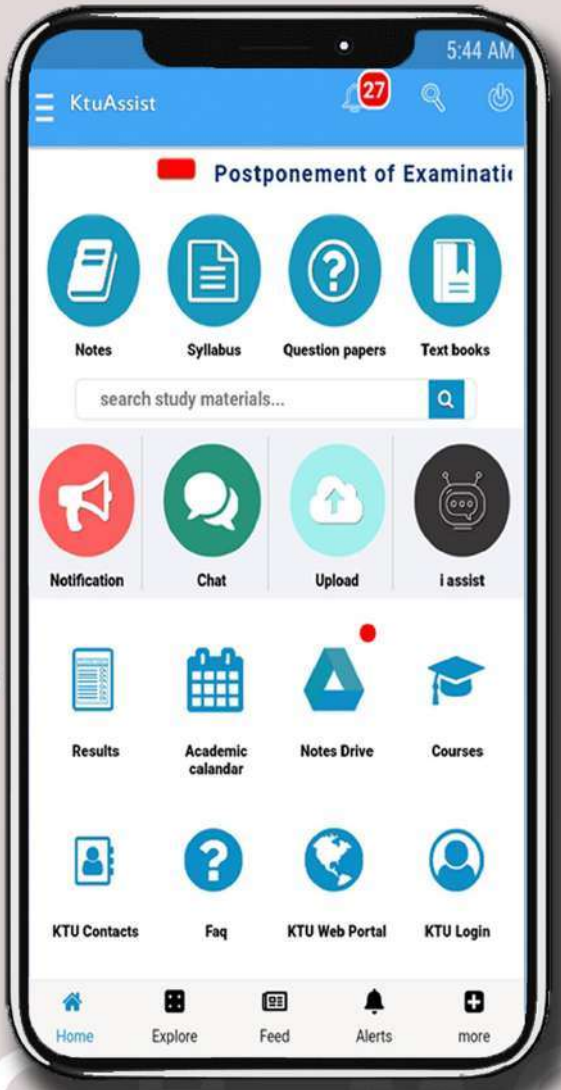
SYLLABUS

NOTES

TEXT BOOKS

QUESTION PAPERS

KTU NOTIFICATION



DOWNLOAD
IT
FROM
GOOGLE PLAY

CHAT

A FAQ

LOGIN

CALENDAR

MUCH MORE

DOWNLOAD APP

ktuassist.in

instagram.com/ktu_assist

facebook.com/ktuassist