# Tutorial - 05

U19CS009

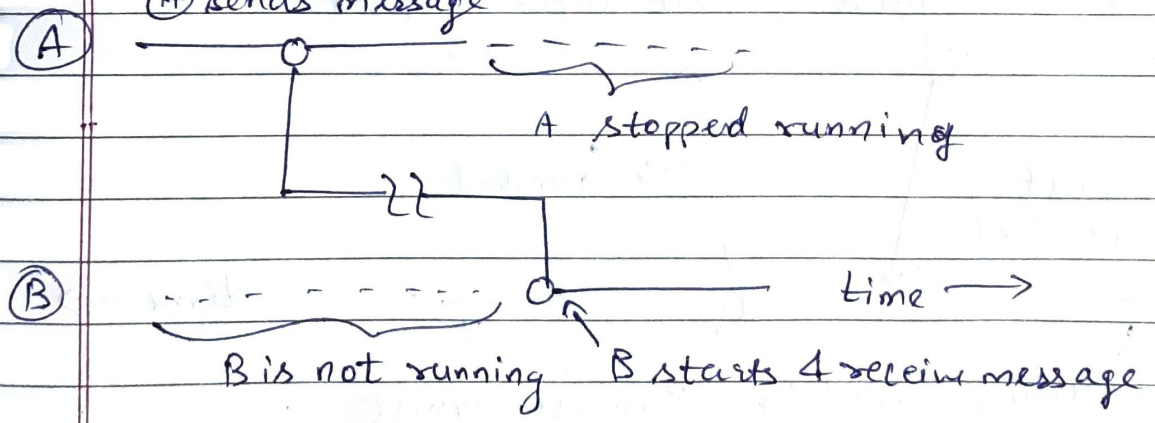
Page No.:
Date : / /

Q-1 Answer the following questions.

a). Give explanation of each figure :-
b). Mention application scenario where each one is application.

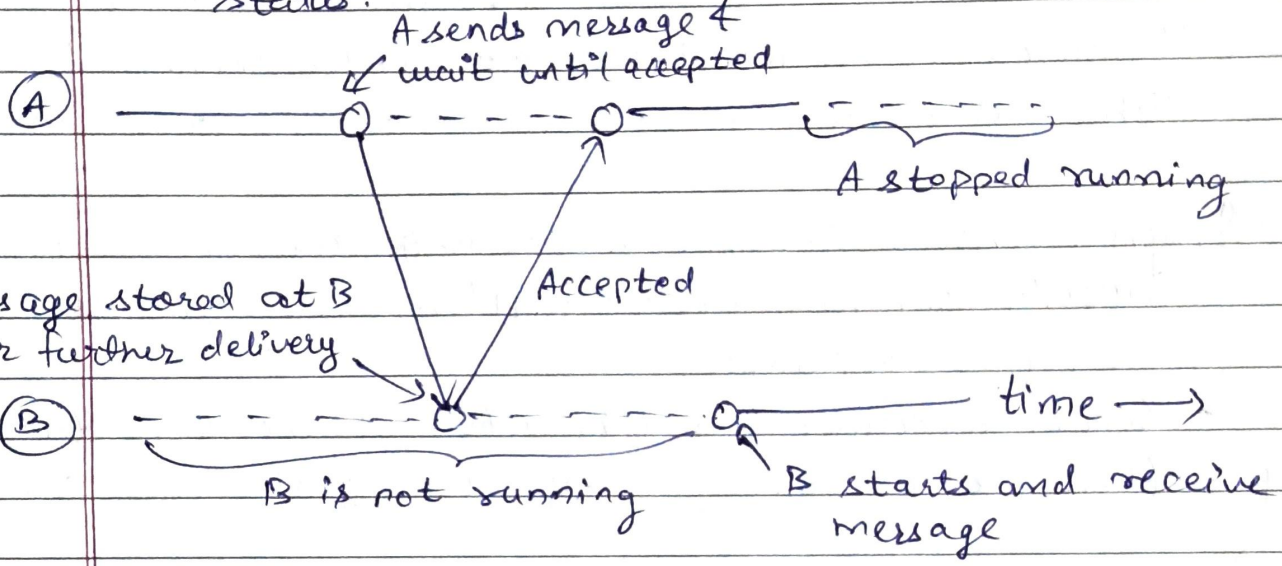
Ans ⇒ Persistent Asynchronous → (A is sender), (B is receiver).
Dark line depicts execution. (A) sends a message and keeps executing without blocking. The message sent from A will take an arbitrary amount of time to reach (B). (A) may or may not be running by the time message reaches (B). Disk of multiple memory queues could be used for storage at (A)'s side. There is a guarantee that the message will eventually reach (B). (B) receives this request and process it then.

→ Eg → Email can take an arbitrary amount of time to reach, and when the receiver ~~want~~ ready the email, the sender might not even be running



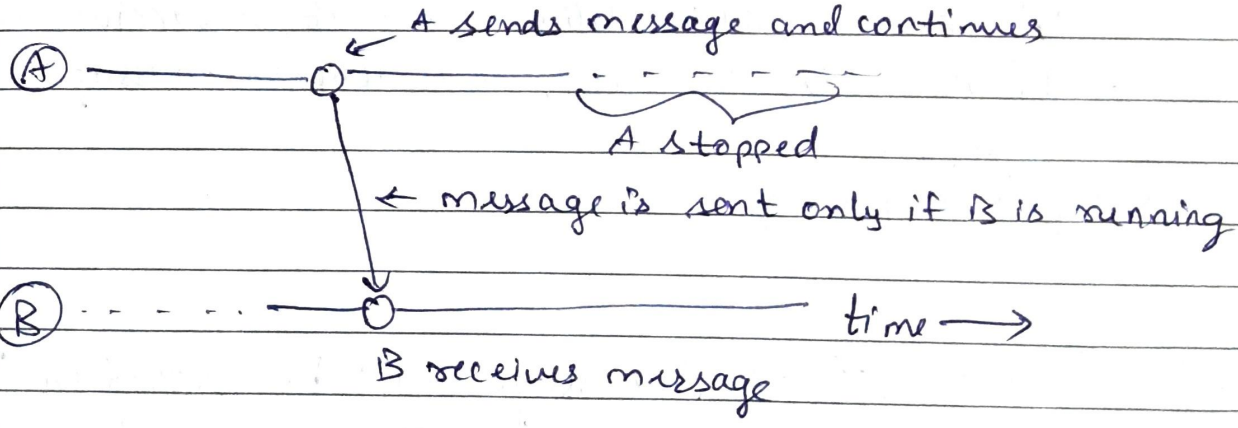

⇒ Persistent Synchronous → (A) sends message and is than blocked (depicted by dotted line). The acknowledgement is for receipt (not delivery response). Because this model is persistent the message may stay in (B)'s queue (or in any router along the way) for an arbitrary amount of time

→ (Eg) Message and Chat applications → many messaging systems are persistent and can tell us the delivery status.



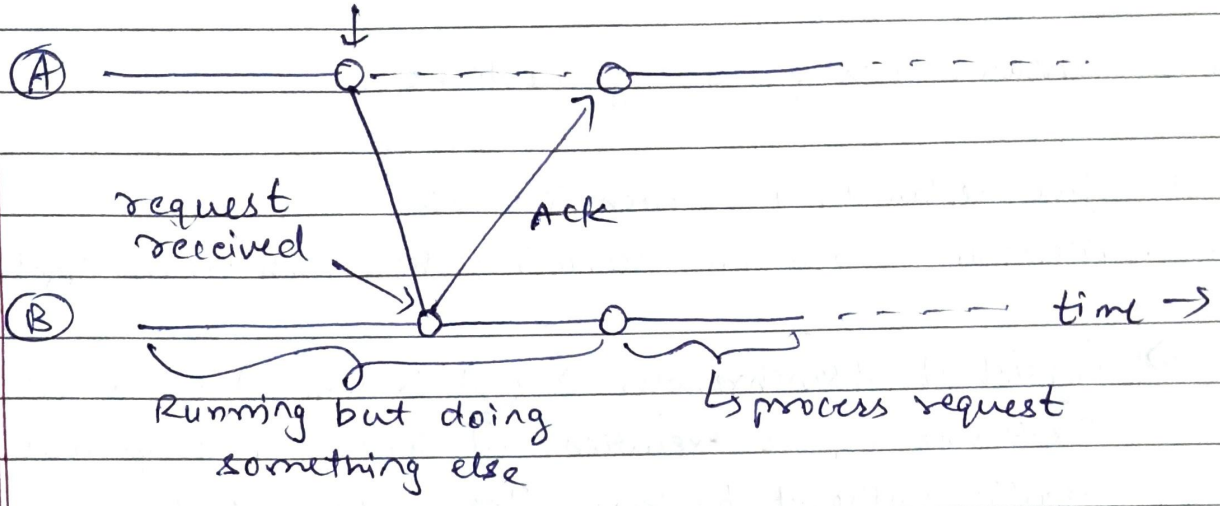

⇒ Transient Asynchronous → (A) sends the message and continues execution (non-blocking). (B) has to be running because if it is not running the message will be discarded. Even if any routers along the way is down, the message will be discarded.

→ (Eg) UDP communication → The function MPI_bsend() is the implementation of this.




⇒ Transient Synchronous (Receipt-based) → (A)'s message to (B) is blocking until an ack is received. This ack simply tells us that the message was received at the other end. It does not tells us anything about whether the process has started.

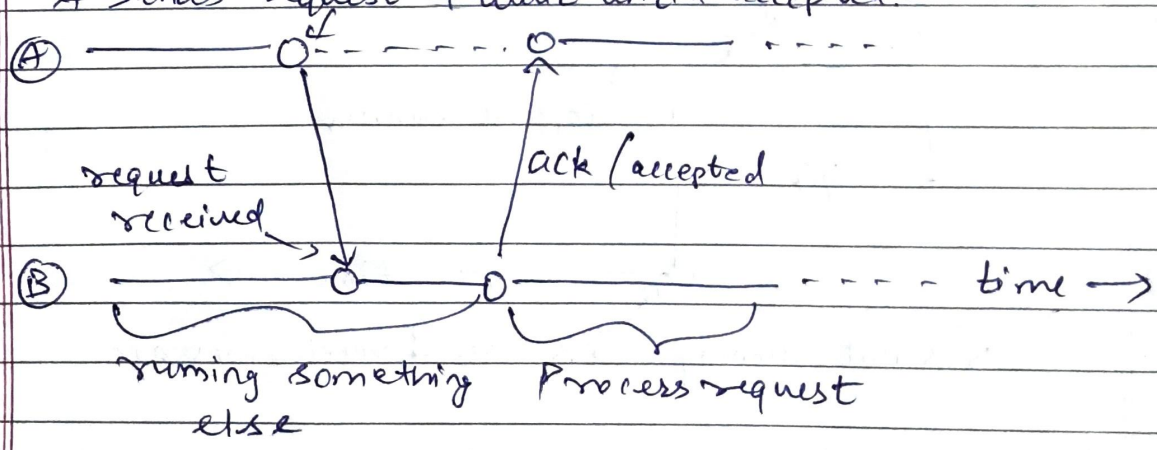
A sends ~~requests 4 wait until received~~




⇒ Transient Synchronous (Delivery-based)

Extension of receipt-based transient synchronous comm. A will resume running when B takes delivery of the message. The ack comes a little bit later than the previous method. This is essentially asynchronous RPC. because from the perspective of an RPC, we are not blocking for the reply.

A sends request 4 wait until accepted.



Page No.:
Date : / /

## ⇒ Transient Synchronous (Response-based)

In this (A) resumes execution upon receiving a response. That is not only has the message been delivered, but it has been processed and the ack comes back in the form of reply. This is transient RPC. The client is blocked for entire duration the reply come back