~~EVENT DRIVEN SERVICES PUBSUB 11 lec~~

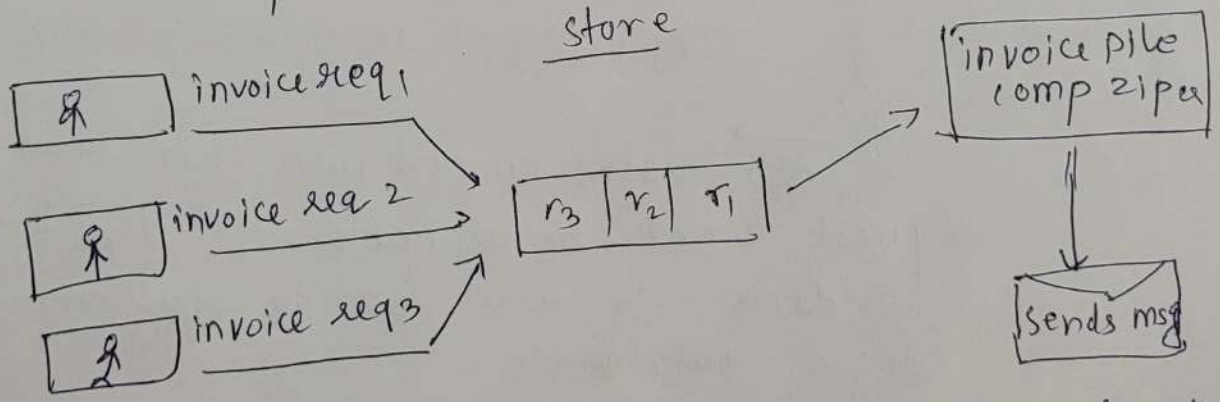
Sudocode SYNCHRONOUS & ASYNCHRONOUS COMMUN

Synchronous commun → when we send the request abve get response within sec.
we get response as soon as we send request
Eg phone call → there is no wait or lag in response from any party

Asynchronous commun → we dont expect response as soon as we send the request. (ie instantaneous or real time not needed)
Eg mails, msg → we dont reply to msg or mails as soon as we get it.

also a person sending a msg doesnot hopes to gets respons in millisec.
↳ Asynchronous commun

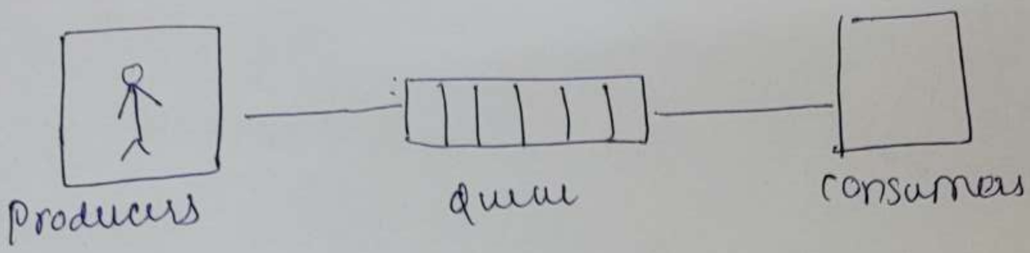
Msg Queue → lining up the msges betn 2 compononents to help them to communicate is msg queue.



Eg in a store online → if 3 members requests for them last 1 wants invoice.
eqn req will be enqueue, when it gets slot it will pick 1 req process it and sends it on mail
here the user dont ~~too~~ expect immediate resp till he gets mail he can do his own work

Eg 2 → when we order a restaurant →

SQS , kafka , RabbitMQ - Eg of msg queue for scalable architecture



Producers → components that sends msges to queue it tells what has to be done.

Consumers → components that carry out the particular operation told by producer.

adv → queue can handle any amount of req

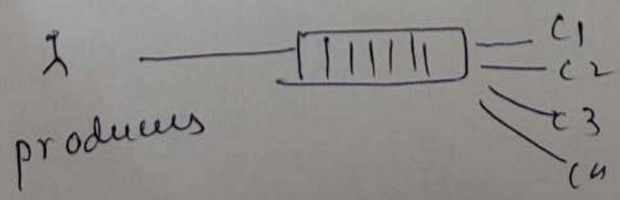
(1) if queue won't be there

Eg then — req → [C] process time req 10min

→ [ ] req is sent on 2nd min then this request won't be acceptle till the first one is done. So user has to stay on page for longer duration

queue → helps to handle loads of load.

(2) queue has consumers that consumes the req if no of req increases. then the no of (producer) consumers can also be increase a. / or deceased Viceversa



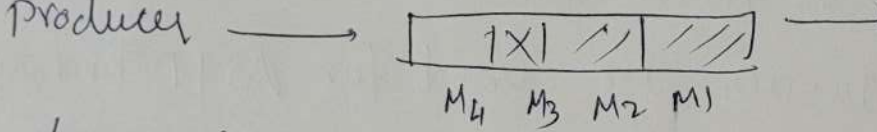
③ If any consumer fails the req would still remain in the queue. and it would be ~~assigned~~ to ~~some other~~ consumer when consumer is up. So actual request would never be lost.

FEATURES OF MSG QUEUE (in producer consumer Model)

① Msg order → FIFO
→ No FIFO
} it depends on our use case

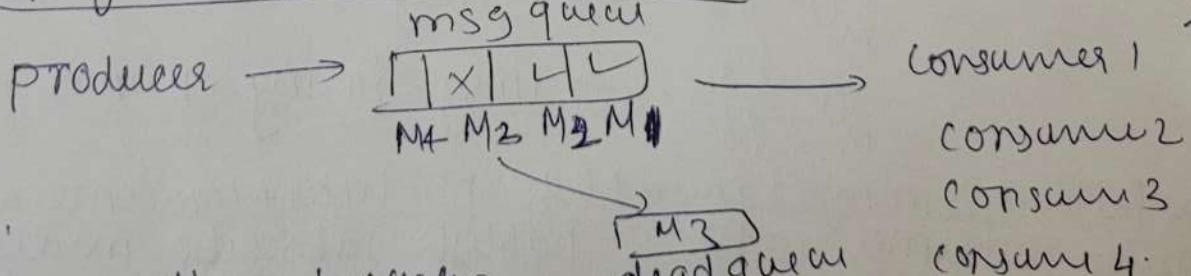
Eg, msg appln → uses FIFO
Eg, Restaurant order → No FIFO or invoice.

→ Msg order FIFO - msg appln (ordered) Eg chat appln



here. if consumer recives M1 & M2 & so m1 & m2 will come out of queue & then consumer 1 fails so M3 would not be sent it will still stay in queue

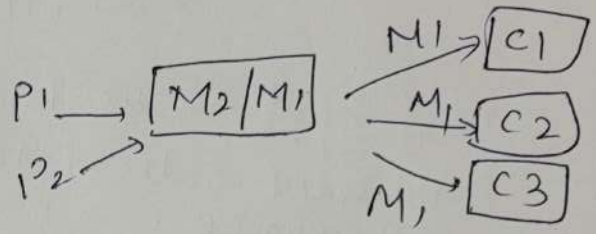
→ Msg non order - NON FIFO (unordered) Eg mail



M1 → will go to cust 2
M2 → goes to cust 2
Now consumer 3 breaks / not available then M3 will go to dead queue, & from dead queue it it will be again pushed to msg main queue → till that time consume 4 will pick up M4

Msg queue → Here one msg will only be gone to one consumer.

* ~~when we want multiple msg to~~
* when we want same msg to be consumed by multiple consumer we will use publish-subscribe model



SUMMARY

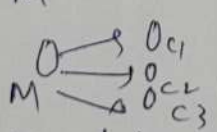
→ SYNCHRONOUS VS ASYNCHRONOUS COMMUNICN

→ HOW MSG Queues are used for Asynchronous COMMUN ie scaling.

→ we can inc/dec no of consumers acc to need

→ Features of msg queue → FIFO & non FIFO.
→ acc to unit case choose. (Ordered or unordered)

$_M O \rightarrow O_c$ → 1 msg → 1 con → model



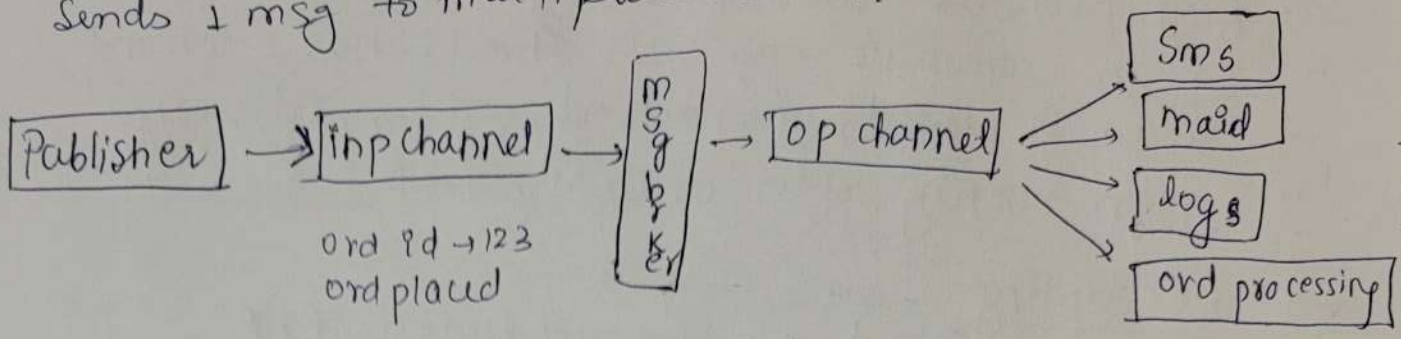
→ 1 msg → many consumer - publish subscribe

→ How failure is managed in ~~publish~~ producer & consumer model

produces consumer model	Publish subscribe model
1 msg → 1 consumer we can inc no of consumer but each msg will go to only one consumer.	1 msg → consm1 consm2 consm3 1 msg to many consum

sudocode PUBLISH SUBSCRIBE MODEL - PUBSUB

Sends 1 msg to multiple consumers.



Services
Consumers

① Publisher ~~requests~~ places order
it goes to inp channel msg broker takes other details of that ord id (ie id name) sends to op channel.
Now for placed ord servie all the 4 subscriber services wants to read, so all 4 services would be called

place order → SMS, mail, log, ord process

② ~~cart~~ add to cart
goes to inp channel → msg brk - op channel.
only log service need to access this req.

Add to cart → log

③ Subscribe to apps newsletter
inp channel - msg ~~queue~~ brker - op chan
mail & Sms wants to acess

Subscribe to app → Sms, mail

PUB SUB USECASE

→ Asynchronous workflow
1 Eg in ecommerce app it should have comms ordr is placed, dispached, out for delivery.
Asyn → only aftr order is placed.

→ Decoupling
2 ord 1 → placed → ~~this~~ yet todespached
ord 2 → this order also can be placed, user donot have to wait for order 1 to get delivered completely

→ Load balancing
3 if large req comes, then msg would be queued.

→ Diffred processing
a specific msg or req we can set its time ie we know during 12 pm-4 pm there is huge traffic so some msg we can Schedule its timing to 4 am, so dirty that time the msg would be consumed.

→ DATA STREAMING

WHEN TO NOT USE PUBSUB Patt

→ when we have small no of bushisu ie msg req eg Go

→ when our API or any feature needs instantaneous response; in this case donot use Pubsub