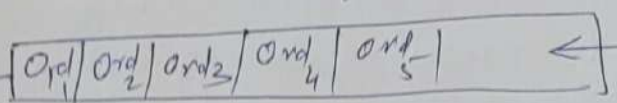
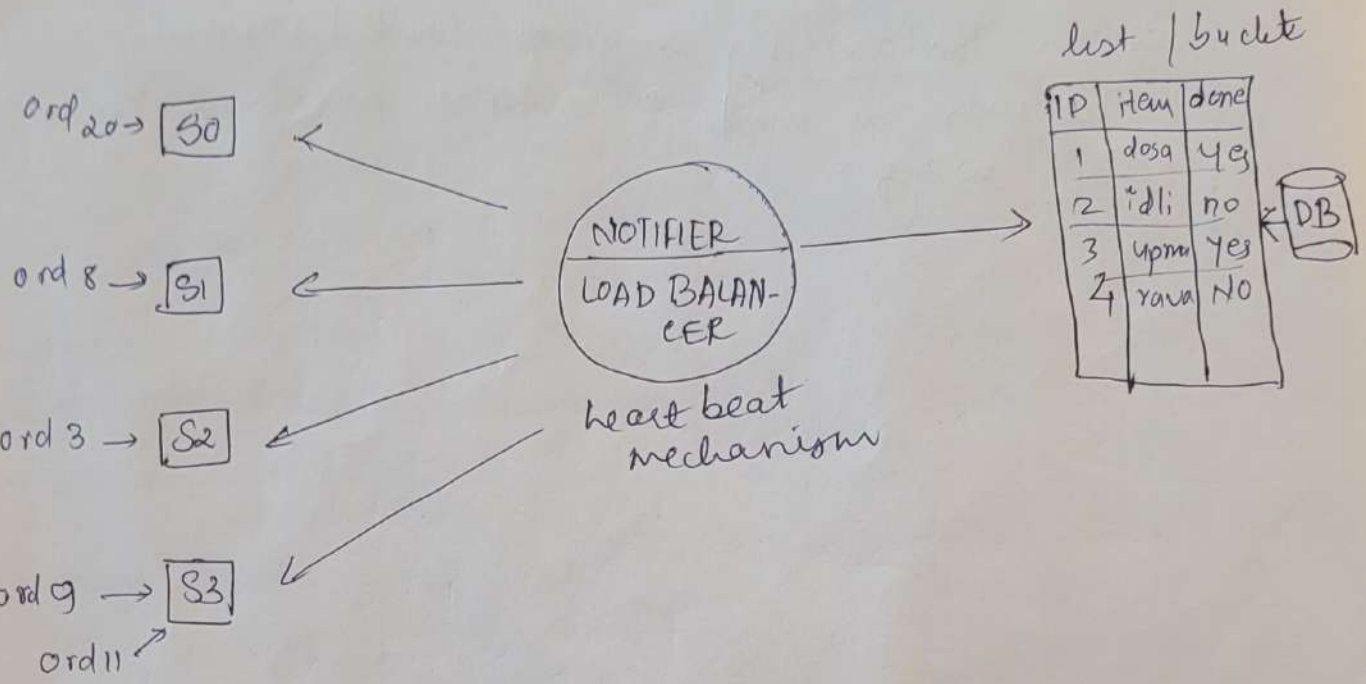
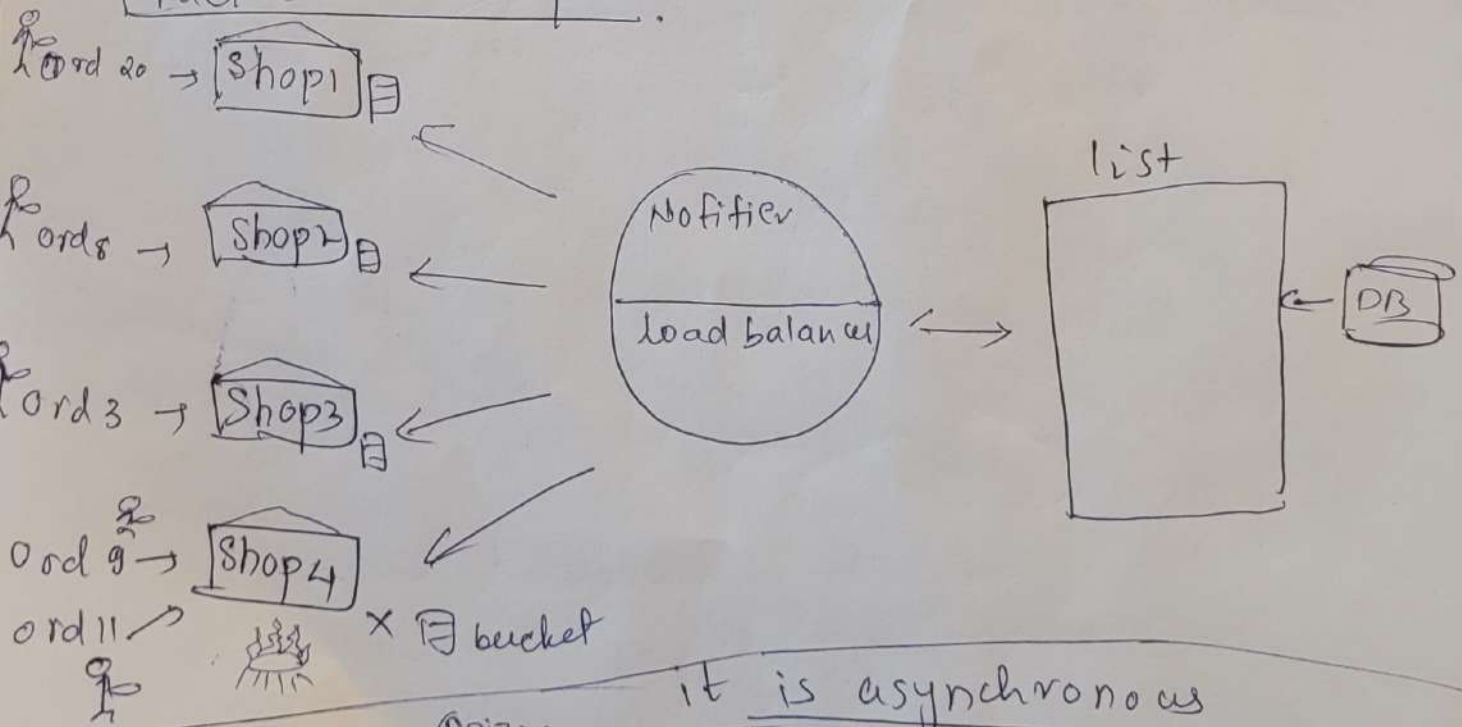
VID 5 → Message Queue & where it is used



once the order is prepared the rest pays us and we remove the order out of queue



Practical Example



ie Shop 3

① Pizza
② burger
③ juice

it is asynchronous

So he can make juice as it is quick even though its order came late

→ Each shop gets a order
→ Suppose shop 4 breaks / catches fire
→ With help of notifier it will notify the DB that there is failture in server / shop 4
→ Request / order 9 + 11 (of shop 4) needs to be rerouted.
→ Load balancer used to route the req also makes sure that duplicate order doesn't goes to same shop.
→ Each shop will have its own bucket where the order is kept
→ Suppose shop 4 gets break and its order 9 goes to shop 3 then a new bucket / order would be added to shop 3.

→ Message / Task queue.
→ it has all the features of Notifier. Load balancer & heart beat mechanism

What it does.

→ It takes request from each server
→ Notifies if any server breaks / fails
→ Balances the load among rest of server and makes sure that ~~does~~ the redistribution of failed server request ~~doesn't goes to the~~ gets distribution ~~equa~~ in a way that there is no duplicate on same server.

ie req 3 do bar ek hi server pe nahi jani chaiye

use of Notifier

it keeps on checking the server.
So it talks to each server and checking if they are alive or not.
If notifier doesn't gets any response from server for longer time it assumes that it is dead.

Lecture 6

Q) what are Microservices
& when & why it is used.

EVENT DRIVEN SERVICES PUBSUB ~~11 lec~~

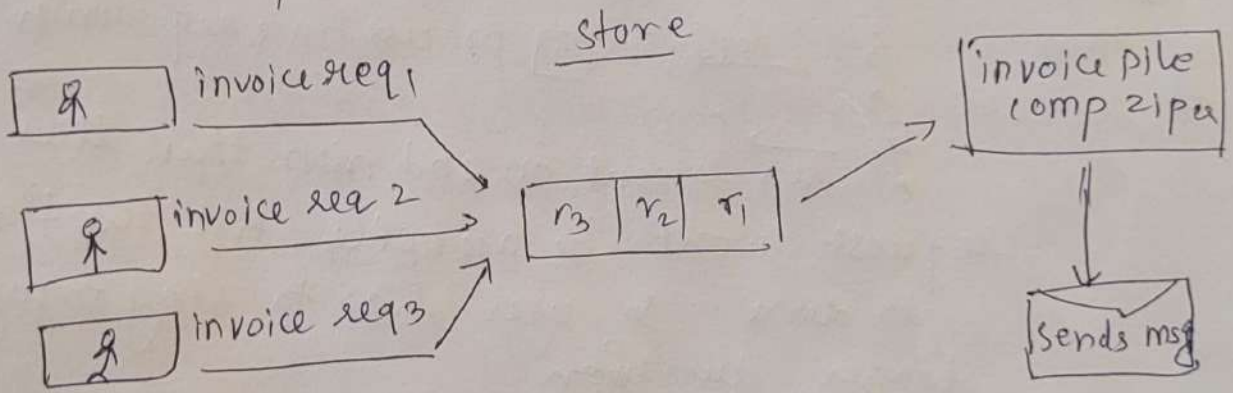
Sudocode

Synchronous commun → when we send the request atwe 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 → Asynchronous commun in millisec.

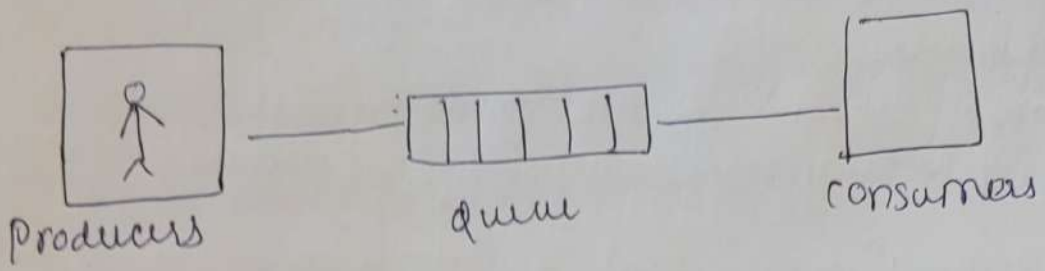
Msg Queue → lining up the msges betn 2 compononents to help them to communicate is msg queue.



Eg in a store online → if 3 members requests for thier last 1 months 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 ~~be~~ expect immediate resp till he gets mail he can do his own work

Eg 2 → when we order a restaurant →

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

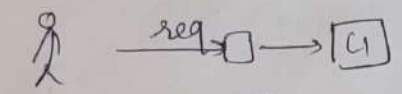


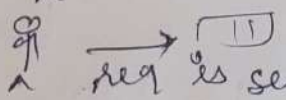
Producers → components that sends mgges to queue it tells what has to be done.

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

adv → queue can handle any amount of req

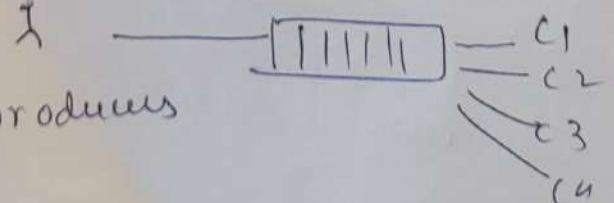
(1) If queue won't be there

Eg then  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 or decreased / vice versa



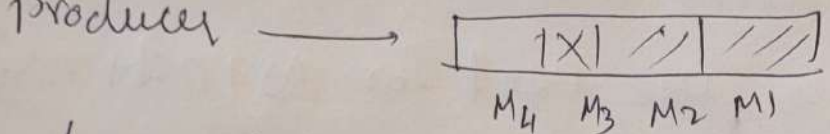
③ If any consumer fails the req would still remain in the queue. and it would be ~~assigned~~ to ~~some other~~ consumer when consumer is up. So actual request would never be lost.

FEATURES OF MSG QUEUE in producer consumer Model

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

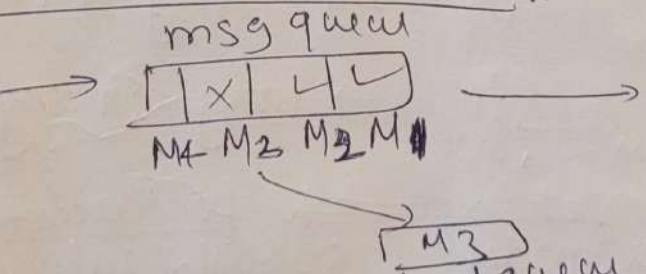
Eg, msg appln → uses FIFO
Eg. Restaurant order → No FIFO or invoice.

→ 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 wouldnot be sent it will still stay in queue

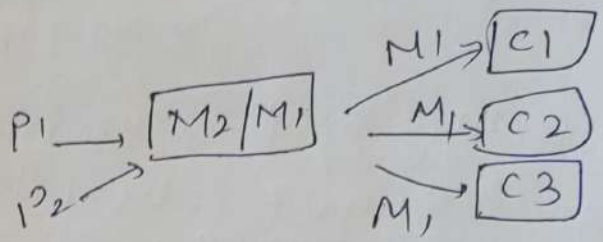
→ Msg non order - NON FIFO (unordered) Eg mail



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

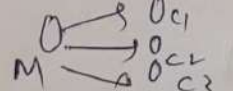
→ SYNCHRONOUS VS ASYNCHRONOUS COMMUNICN

→ 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)

→ 1 msg → 1 con → model



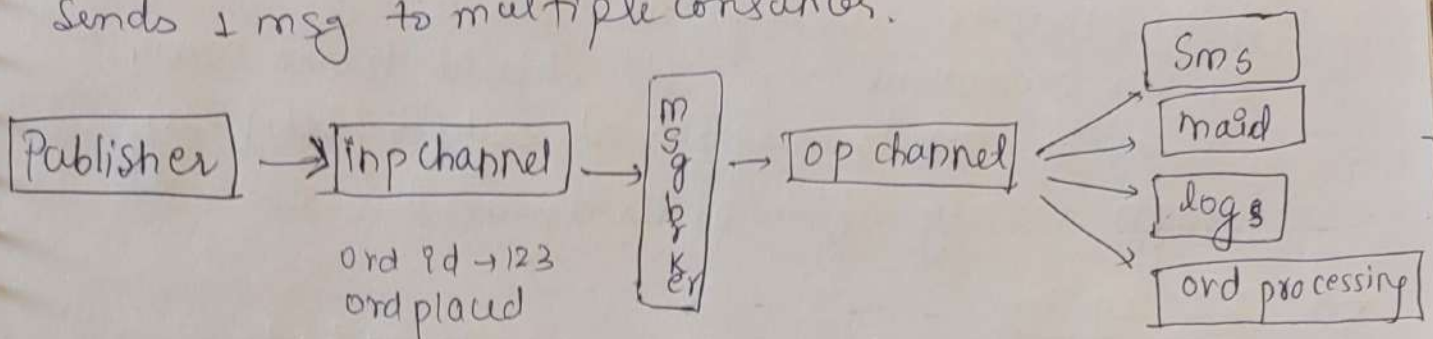
→ 1 msg → many consumer - publish subscribe

→ ~~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 service 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 access

Subscribe to app → SMS, mail