System Design Notification Service

F.R → 1) Service used by third party clients to notify customers

2) Notification type → Low priority (ad/promotion etc)

→ High priority (alerts/order status etc)

3) Message order from same publisher should be gauranteed

4) Message delivery status to be informed to publisher. Can be a daily report or a background task.

NFR → 1) High availability

2) High priority notifications → Realtime

→ Atleast once

3) Low priority → Can expect delay

↓ Atmost once for better UX

Capacity estimation $\rightarrow$

Storage $\rightarrow$

Assume 10K publisher clients. Each client has 10M subscribers. Each publisher pushes 10 high priority & 2 low priority messages/day/subscriber

$\Rightarrow$ Messages to be delivered in 1 day at max & deleted from storage (to be confirmed from interviewer)

each message size $\simeq$ 10 KB. $\Rightarrow$ (10+2) $\times$ 10 K message storage/day $\Rightarrow$ 10 MB/day

each message entry = (pubID + subscriber ID + message ID) $\simeq$ 1 KB max

total entries/day $\simeq$ 10K $\times$ 10 M

$\Rightarrow$ total storage size req $\simeq 10^4 \times 10^7 \times 10^3$ B $= 10^{14}$ B = 100 TB

B/W $\rightarrow$ High priority messages in almost realtime & low prio in few hrs

Main bottleneck in storing messages to temp storage after processing, loading reciever details & then forwarding the messages

$\Rightarrow$ Assume 100 MBPS storage speed & .10 MBPS network latency

$\Rightarrow$ 10K publishers publish messages & 10M × 10K subscribers read those messages

$\Rightarrow$ 10×10^{11} messages dispatched / day $\Rightarrow$ $10^{12}\times 10$ KB $\Rightarrow$ 10^{16} B / day $\Rightarrow$ $10^{16}/(24\times3600)$ B/sec

$\Rightarrow$ $\simeq 10^{10}$ B/S $\Rightarrow$ 10^{4} MBPS required

$\Rightarrow$ ~100 processing servers & 10^{3} dispatch servers needed

APIs needed → 1) registerPublisher(pubAdress), → pubID
2) register Subscriber(sub Adress, pubID) → sub ID
3) pushMessage(pubID, List subIDs, String message content, priority)

Services → Registration service, Message storage service, Message delivery service
Delivery status service, Reporting & Analytics service

Data storage → RDBMS NoSQL

Mesg → Mesg ID, PubID, Content
Priority, Delivery Status, Seq No.

Publisher → PubID, PubAdress, etc

Subscriber → Sub ID, Pub ID, Sub Adress

Mesg ID → {Sub ID1, Sub ID2 ...}

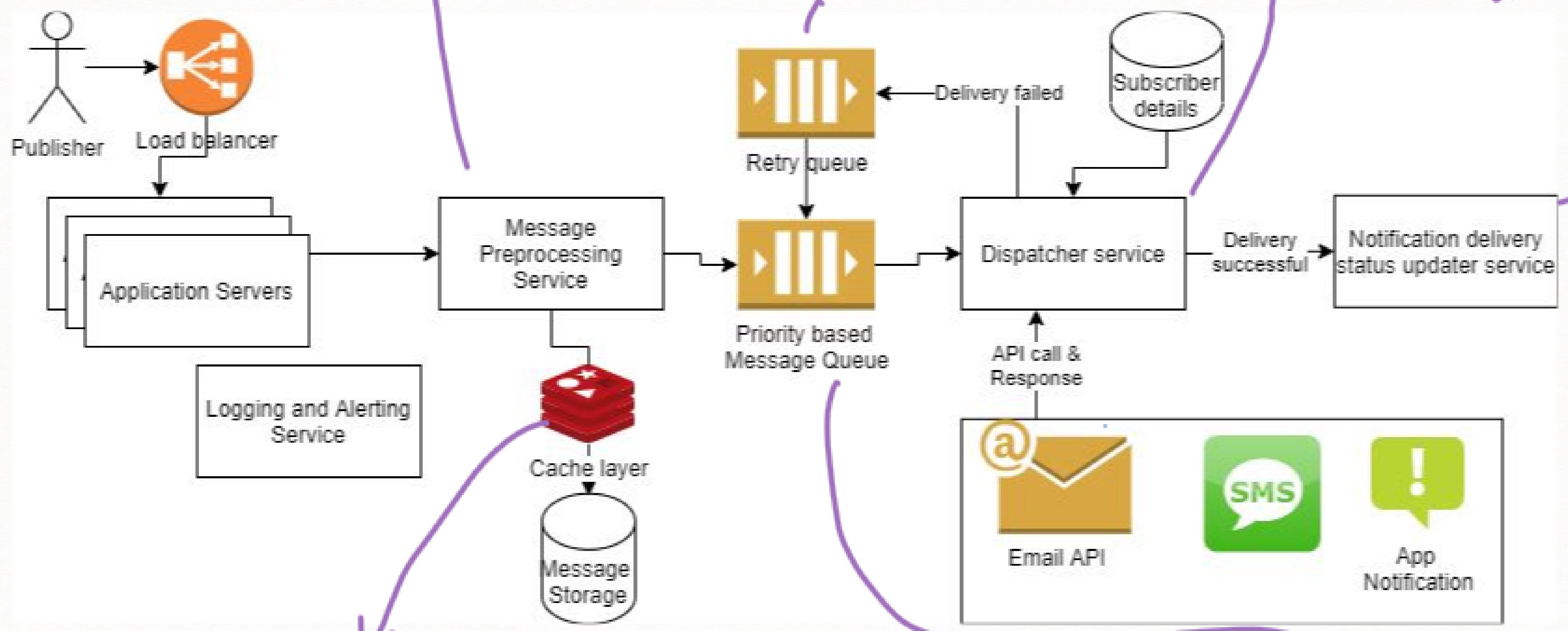
NoSQL for mesg receiver details since that is huge volume hence need easy partitionability

HLD

Preprocessor responsible to order the incoming msgs by timestamp/req No generated from client

Needed for atleast once delivery.

Dispatcher service responsible to load subscriber address and call APIs to send notifications



Background service to give reports to publisher

Cache used to temporarily store msgs to avoid back pressure from MQ
Msgs stored in DB asynchronously if required by analytics services

MQ needed to decouple Sender & receiver services. Dispatcher may be overloaded in peak traffic