#ASLI ENGINEERING

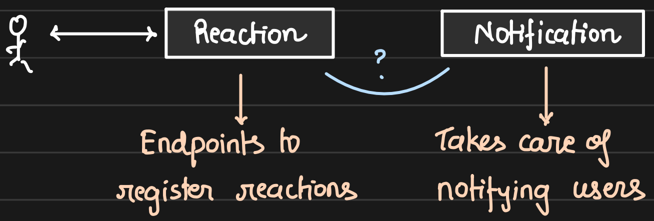
Sync Async Communication in Microservices

SWIPE

BY

ARPIT BHAYANI

Synchonous and Asynchronous Communication


Reaction
Notification
?
Endpoints to register reactions
Takes care of notifying users

Say, we are building a social network and we have to notify user B when user A likes/comments on the post.

How should the Reaction service communicate with the Notification service to notify user B?

What if we just have a monolith?

Sending out the notificatis is just a function call.

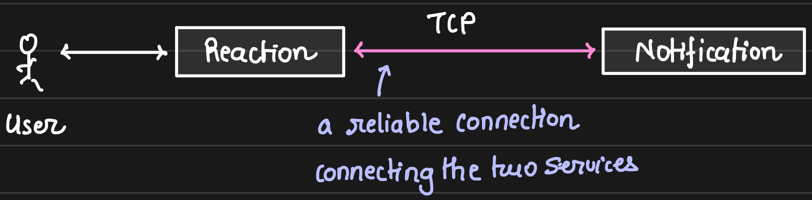
- ↳ superfast
- ↳ always succeeds → invaluable

Things become extra-challenging when we have a Distributed System

- ↳ network failure
- ↳ target service is down
- ↳ target service is over-whelmed
- ↳ target service is not reachable

So, how do we make services talk to each other?

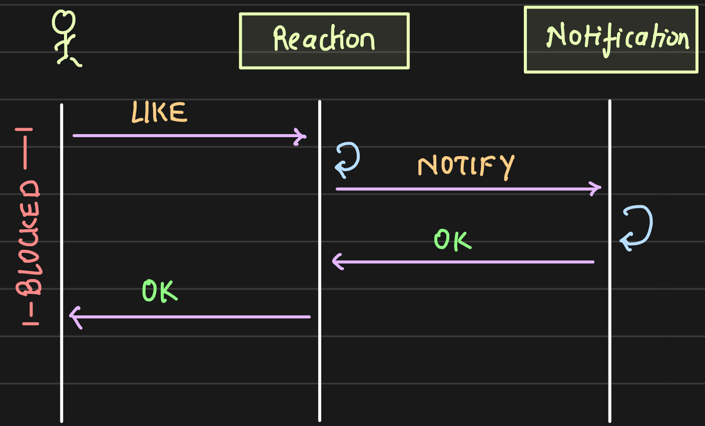
Synchronous Communication



a reliable connection connecting the two services

One service sends the request, and waits for the response from the other service

Request made from one service to another is BLOCKING



How do these services actually communicate?

Most communication paradigms are based on HTTP like REST, GraphQL and gRPC

*We will touch upon each in detail some other time

Advantages of Synchronous Communication

- ↳ communication happens in realtime
- ↳ super simple, intuitive

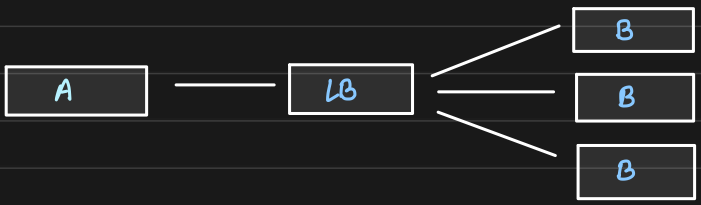
Disadvantages of Synchronous Communication

ms, sec, minutes

- ↳ Caller is blocked until the response is received
  - ↳ would be an issue if it takes too much time

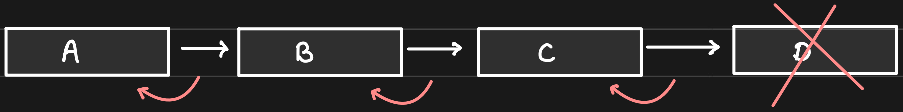
Timeouts

- ↳ servers need to be pro-actively provisioned for peaks.



Servers need to be very readily available to handle the incoming requests

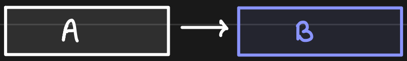
- ↳ Risk of cascading failures



To mitigate this we need circuit breakers

Arpit Bhayani

↳ creates a strong coupling b/w participating services



service A should be kept in the loop for any changes happening in B

versioning
strong contract
backward compatability

When should you use synchronous communication ?

↳ when you cannot move on

You need result before you move forward

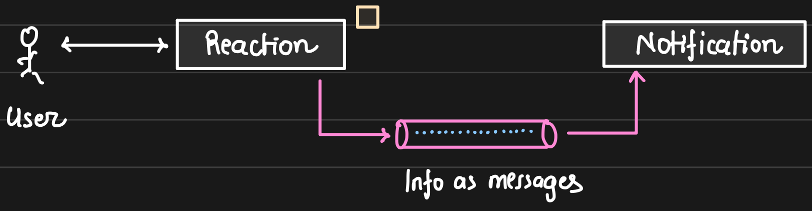
eg: Database Queries , API responses

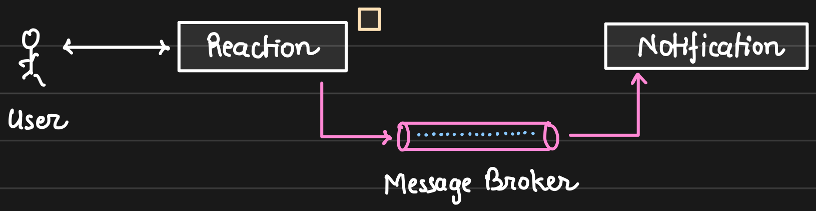
↳ when you want realtime response

eg: chat, checkout

↳ when it will take relatively less time to compute and respond

Asynchronous Communication



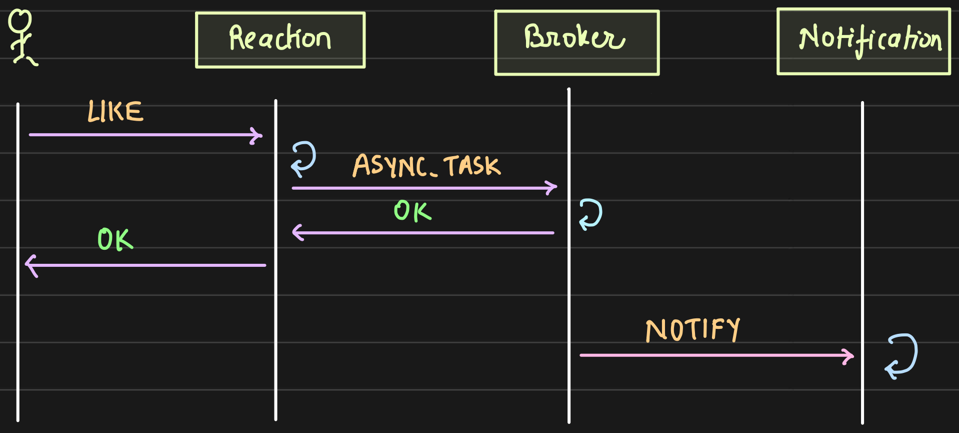


Interaction between services is an exchange of text message via a message broker

The messages are buffered in the broker and the consuming services will consume them when it can.

If the consuming service is down, the messages will be consumed when the service comes back up again. So, no cascading failure

Request made from one service to another is NONBLOCKING

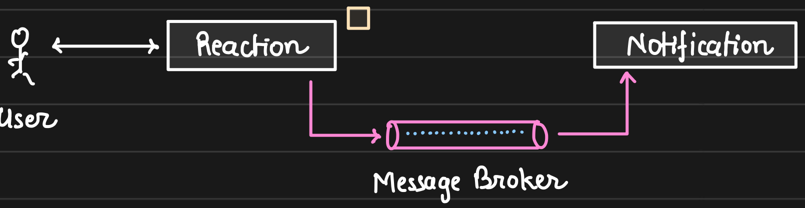


A few Message Brokers are

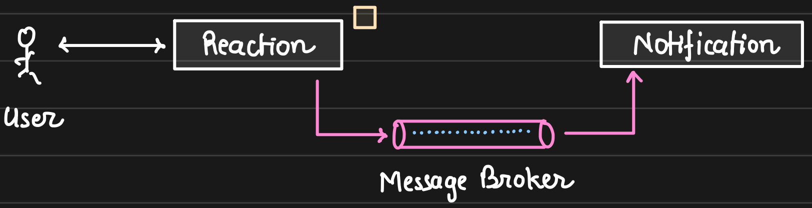
RabbitMQ, SQS, Kafka, Kinesis, Google PubSub

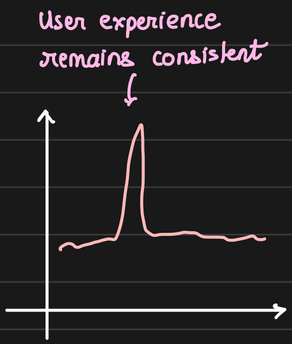
Advantages of Asynchronous Communication

↳ Services don't need to wait for the response



↳ System can handle surge and spikes better





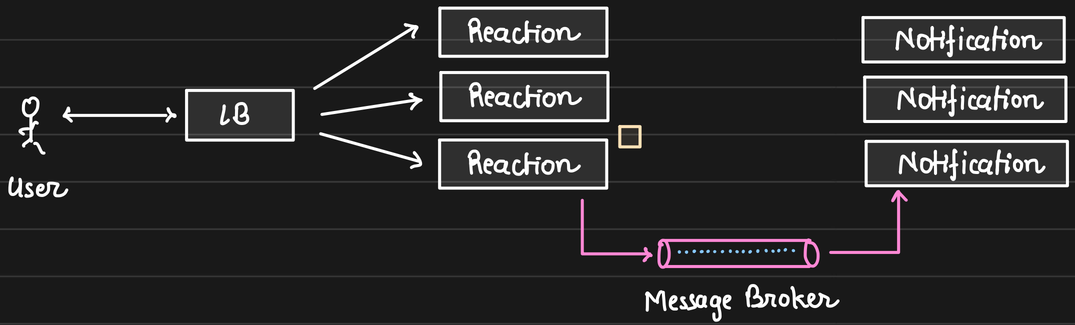
In case of a surge, the messages will pile up in the broker but there will be no difference in user experience

* Notification service would need to scale up to clear the backlog

↳ No need of pro-active scaling

Upon surge the messages will queue up but there will be no difference in UX. You will need to scale to consume the messages eventually, but not realtime

↳ No load balancer required, so no additional network hop



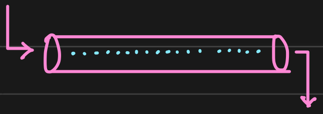
↳ No request drop or data loss

In synchronous communication, if the target service is overwhelmed there will drop in request, but with asynchronous because we have a message buffer there is no request loss

The messages pile up and the target system eventually catches up.

↳ Better control over failures

In case of a failure you can always retry because message is still there in the broker



↳ Services are truly decoupled

Disadvantages of Asynchronous Communication

↳ Eventual Consistency

You cannot have a strongly consistent system with brokers and hence you have to be okay for the messages to be eventually consumed.

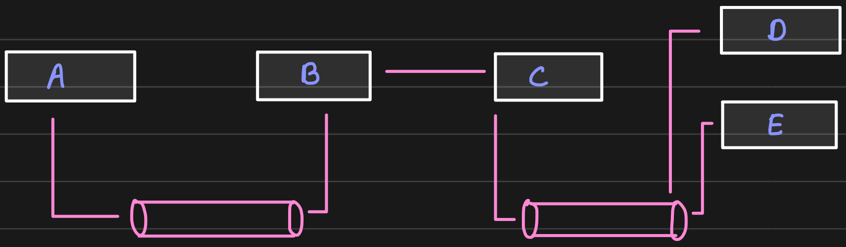
But with brokers our system does scale better

↳ Broker is a SPoF

The message broker is the backbone of the system, hence we need to be super cautious about it.

The broker we use should be horizontally scalable

↳ Harder to track the flow of communication



ARPIT BHAYANI

When should we use Asynchronous communication?

↳ when delay in processing is okay

eg: notification , analytics , reporting

↳ when the job at hand is long-running

eg: provisioning a server , order tracking , DB backups

↳ when multiple services need to 'react' to same event

eg: blog published
- index in search
- notify the followers
- update user analytics

↳ when it is okay for you to allow failures and retries

eg: send notification. if failed retry