

StreamProcesses2

MessageQueue:

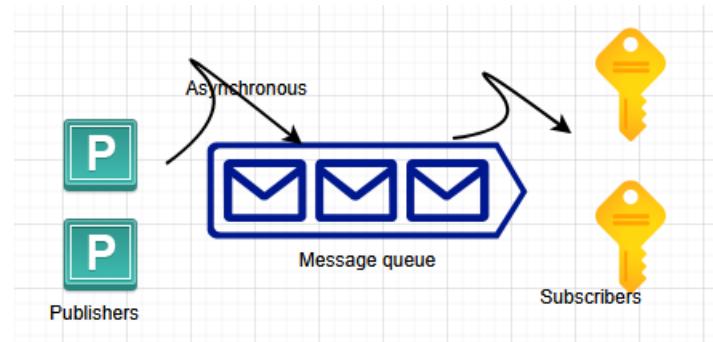
Sync and Async communication: Sync is processing requests one by one without changing the order. In Async, we can fire a request then forget about it.

Pub/Sub Case is an important topic in Message Queue:

Pub – Publisher (sends messages)

Sub – Subscriber (receives messages and work accordingly)

Between Publisher and Subscriber we have a Message Queue



Various Subscribers can pull requests from Message Queue.

Message Queue:

It should hold messages

It should not go to multiple Subscribers

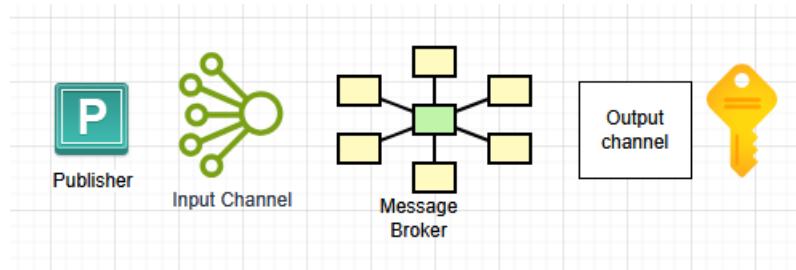
We don't want our users to get multiple payment requests so that's maintain the state

State of the message could be assigned to subscriber, processed or unprocessed

To see if they are failing – to manage failures also through DLQ (Dead Letter Queue)

It should check the Subscriber status also

It also does the Load balancing of Subscribers to process the messages



Publisher share requests with Input channel. Input channel sends requests to Message Broker application. From that Output channel gets the responses from Message broker. From that Subscriber picks the request to process.

Input channel:

- where the events are published

Output channel:

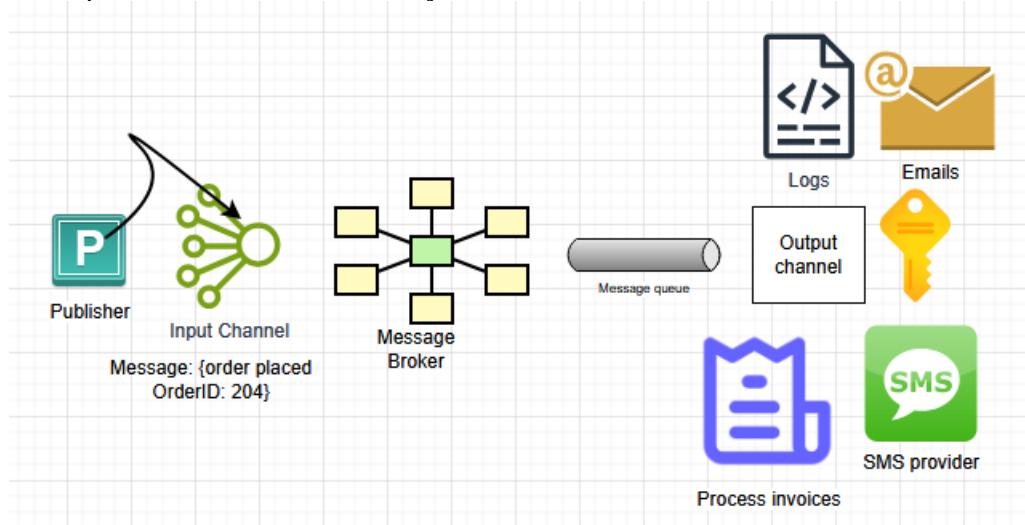
- Provide tasks or data or operation to the Subscribers
- There can be multiple output channels

Message Broker: Message broker is the application for handling message queue. it is responsible for getting messages from Publisher and to place messages in the queue. It could change the data (modify data, it adds data such that it is beneficial for the Subscriber).

- It could add Topic. A Topic is like category of requests. Subscribers choose their messages according to their category.
- It categorizes the messages
- Subscriber will pick a particular topic

Example Placing of order

When an order is placed in an e-commerce system.



First Publisher shares with Input channel, then it goes to Message broker, then we can have multiple subscribers like Sms, Email, Logs or Process invoices services. Message broker will make sure SMS service is only taking SMS data not other data. For Add to cart operation, we don't need SMS service or Email service.

AddToCart ==> ATC topic ==> Log. Message broker will send to only Log service not any other subscribers.

When you click SubscribeNow button ==> SN topic ==> Only Email service is required. Maybe, Logs are required. We don't need any other subscribers like Invoice or SMS service etc. SMS will not fetch any requests from SN topic or ATC topic. In Pub/Sub model, Message broker cannot send a message or request to the Publisher, which is a drawback of the model. Publisher is trying to process Async operations in Pub Sub model. So Publisher don't want to wait for anything / any reply from Message broker. If SMS service fails, Message broker will know that information. Responsible person is the developer or architect, which they design SMS service they must ensure what kind of data it can process. When we say Message broker can change data, that means it can add Timestamp to the data, assigns topic, add routing information (Source & Destination info), add priority.

Factors of Pub / Sub:

- Message order (Messages from Publisher can be sent to Message queue in a particular order but messages can be processed in an entirely different order). Order is not managed. Work around is PriorityQueue if we want to maintain an order. We could attach Priority to every message then PriorityQueue can take care of order. Mostly, if we want order, then we stick to Sync approaches.
- Message consumption: Message could be consumed in any order. If you want order, we need to use PriorityQueues.
- Poison message: Messages that are not handled by Subscribers. Messages that Subscriber is not able to process. Say if date is 26/12/2025. If Subscriber thinks like MM/DD/YYYY then 26 is an invalid month. They even fail after multiple retries. The poison messages are consuming our resources. How do we check? Check through Logs
- Duplicate message: One message should be taken care by one Subscriber. One dedicated subscriber for each topic.

Use cases of Pub/Sub model:

1. Async workload: wherever we can deliberately push our task to Message queue and forget about it. We can use it in Fire and Forget kind of system. If order is placed, we can send notification, email, sms, talk to banking etc in parallel in Pub Sub model. If we want to do it in a Sync then we don't use Pub Sub.
2. Decoupling: We are telling the user we are going to use Pub Sub in Analytics. Our system will work freely for Analytics task without waiting. Pub Sub model can be used in Analytics
3. Load balancing: Message queue works as a Load balancer. If load increases, Message queues will hold the request instead of bombarding consumers. Consumers can pick whenever they are free. Message queue will also keep a check on Subscribers whether they are up or not. Only if they are up, they can pull the requests from Message queue. If load is increased, one solution is increase the number of Subscribers so they can process bit faster. If the Subscribers are limited, then Message queue will hold those messages.
4. Deferred system: Do it later kind of system. It can put it in Backlog. Scheduled processes (cron kind of application where they generate invoices after every 2 weeks).

When not to use Pub/Sub model:

- Don't have a lot of users [100-500]. The user count is not high enough we need a Message broker here
- Low number of requests (we will use only if requests > 1000 / 10000 requests per second)
- Real-time application (avoid Pub/Sub model with multiple Publishers or Subscribers)
- When we need acknowledgement (we don't use Pub/Sub)

