Stream processes:

<u>Messaging queue:</u>

Messaging queues are life-savers for our communication channel, especially if we are having a lot of requests.

Synchronous and Asynchronous communication:

In Synchronous requests,

- we will handle requests one by one.
- Any Rest call
- TCP is an example
- Both parties wait for confirmation
- Single thread only
- Request and response with the same session
- Our server could be blocked because they are waiting for other responses
- it will impact our performance due to longer wait period

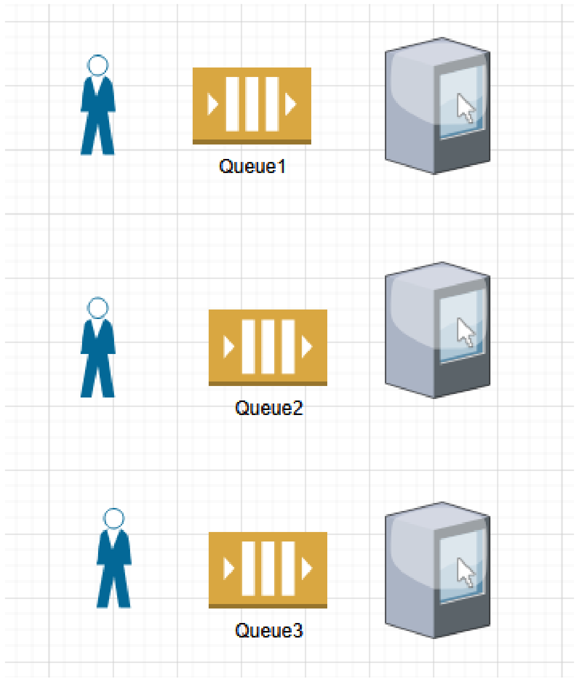
Examples where Sync communication is a must:
- Security
- Chatting
- Ticket booking
- Bank transactions
- Authentication or Authorization
- Payment

Asynchronous requests

- dump our request and move on to the next task, example: in a restaurant, we order food then we do other things. Then we forget about our food.
- Fire and forget type of operations
- Unblock us
- We have to do Event/Failure handling

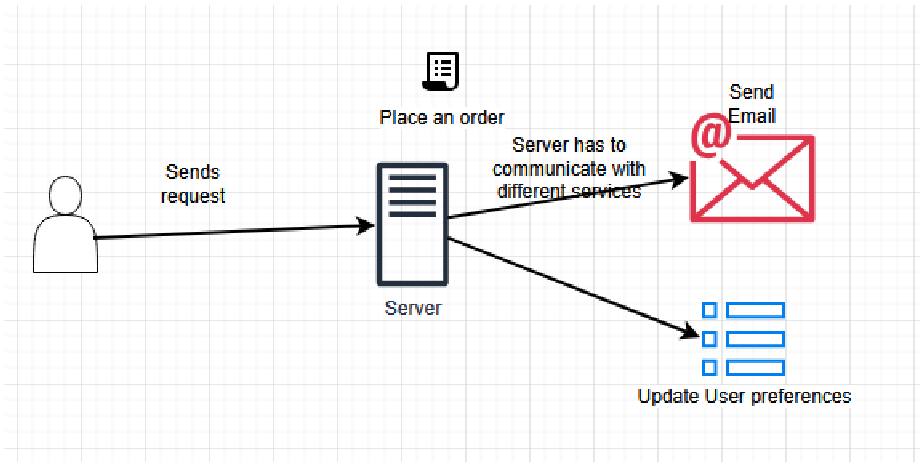
Examples where we could implement Async communication:
- Live broadcasting
- Emails
- SMS / any notifications

Say 100 people are going to a bank and we have only 3 counters serving customers.
Different services:
- Print passbook
- Create check

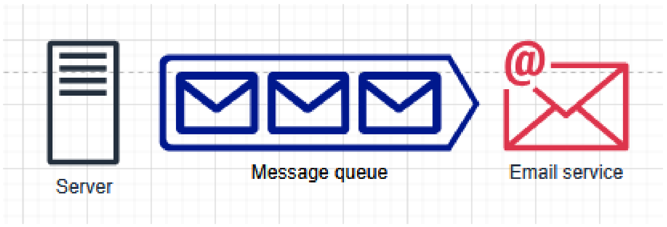


There are 3 queues and 3 counters. Sometimes, you don’t have to be in queues if you get a token number. Lets say we have token system. Token 1 will go to Counter 1, Token 2 to Counter 2 and Token 3 to Counter 3 and so forth. Queues are helping to organize the system in a better way. If we have all 100 customers directly hit the Counters, there will be problems and Counters might crash. Similarly, instead of sending all requests to servers directly, we can organize queue type structure and in that queue every request will be answered in some time.
Lets say we have a friend who is in an Emergency situation and needs to make a call immediately. At that time, he needs to call directly, he cant put a WhatsApp message and wait forever or have Async communications. In Sync communication, the person that’s putting a call cant perform any other tasks unless this call task is completed. He cant do anything else unless he is free from the task he is performing. So in Synchronous cases, Queue is generally not preferred. Say ServiceA is calling ServiceB in Synchronous manner, in that case, ServiceA is waiting for the response before moving-on to something else. We cant put a queue in between and have ServiceA wait forever. In that kind of case, we prefer not to have a queue. Say payment is processed and order is placed then we got to send an email notification. For sending email notification, queue works.

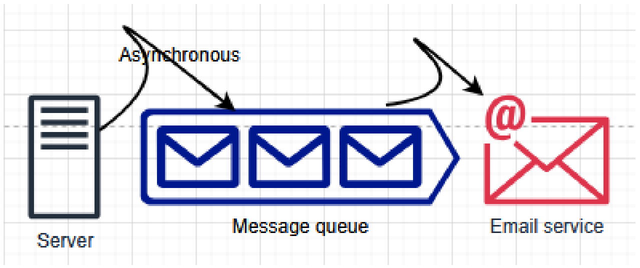
What are Message queue?
- The queues which contains some message.
- Some applications that can create this type of queues: Kafka, RabbitMQ, AWS SQS, Redis stream, Google Pub/Sub, Microsoft Azure service Bus



The server needs to communicate with different services like Send email and Update user preferences based on the purchase order. Servers don't have to notify user for these services. In a big system like Amazon, Walmart, say massive number of orders are created per second, 1. we cant simply expect our Email service to work in a Synchronous way, 2. we have a case we could use our Message queue.



Message queues can pick up requests from Server and pass them on to Email service in FIFO. How can we incorporate Message queue in our system? By incorporating above-mentioned commercial applications like SQS.

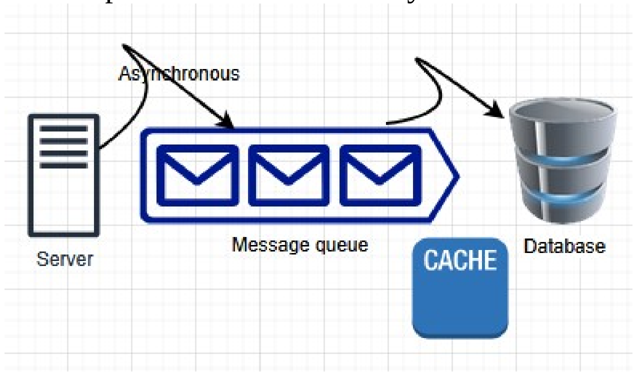


Server will hand-over requests to Message queue (asynchronously) and then Message queue can take care of the requests. Now Server is free to do other things.

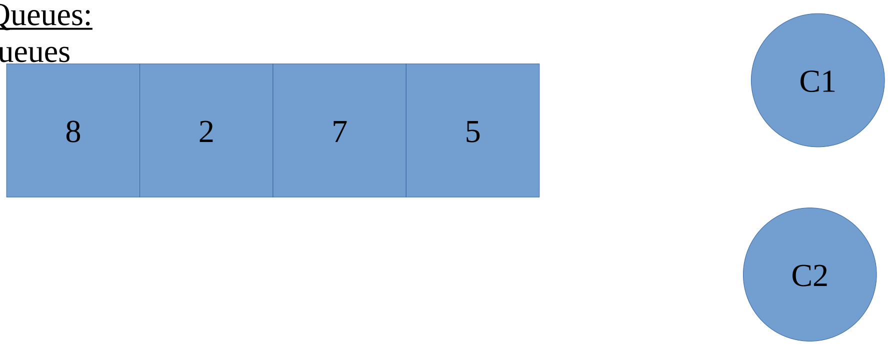
Characteristics of Message queue:
- Holding messages
- Responsibility of Message queue (since Server is out) is to follow-up with different services and to make sure requests are done.
- Responsibility of communicating with different services. While communicating with different services, some of the requests might fail. Message queue should be able to segregate those failed requests from successful ones. They should be capable of handling failed requests also.

- They help other 3rd party services like Paypal / Stripe by regulating requests and traffic load.

Until the Services talk to Message queue about the requests, Message queue will hold on to those messages and wont tell them unless they are asked first. The Servers that are giving requests to Message queue, they can dump as many requests they want on to Message queue.
Message queues are important components of a Database system with cache also

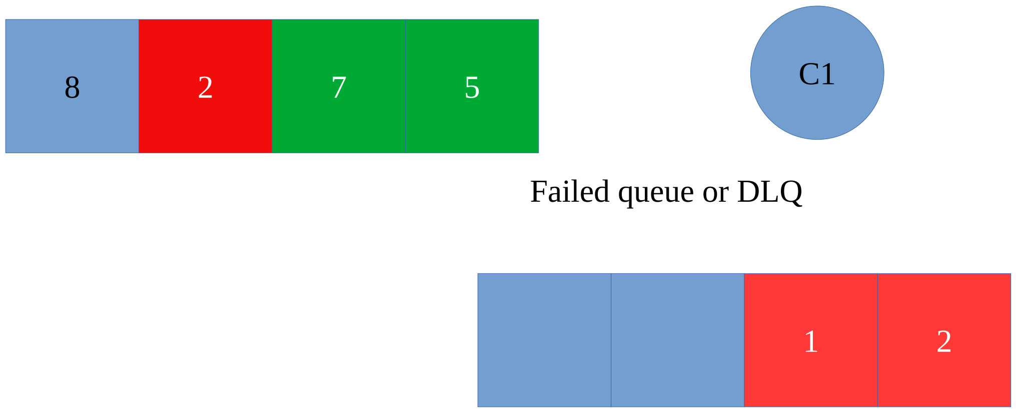


<u>Types of Queues:</u>
Ordered queues



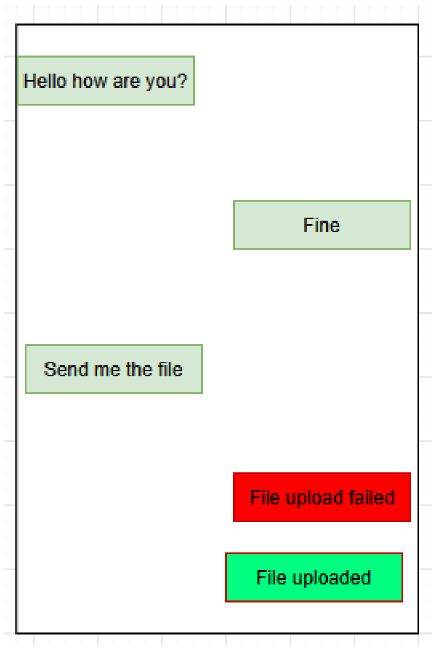
Lets say we have Customers 8,2,7,5 in the Message queue being processed by Counters C1 and C2
C1 pulls the first request 5. Once C1 gives the confirmation that Request 1 is done, it will discard 1 from the queue. The one that's pulling the requests from the Message queue is the Consumer and the one that's filling Message queue with requests is the Producers. After Request 5 is processed, Request 7 will be pulled by Consumer
- Consumer will pull the requests
- Consumer works on the requests
- Consumer notifies the MessageQueue, success or failed
In Ordered queue, our Consumer has to go in the sequence only like Requests 5, 7, 2, 8. They cannot jump from 5 to 8 directly. FIFO algorithm is used here.
Worst cases: Lets say C1 was down for sometime while working on the Request 7. In that case, they cant proceed to Request 2 because it processes in order only. They are blocked. We cant handle failures because they are strictly ordered and we cant break the order.

Unordered queues
We don’t need to follow any sequential order, we can skip the order also. Like processing emails. Say Request 7 breaks down, we don’t care, we will continue processing the next requests in unordered queue. In this case, we don’t have to follow order. In case of failure, the consumer can pull the next request. What happens when a Request fails? We can have a retry queue / failed queue / DLQ. Dead Later Queue.
- Managing mail service



Service or Counter 1 successfully processed Requests 5,7 but couldn’t process Request 2. So it puts all the failed requests into DLQ. MessageQueue can decide whether they need a retry or they can be fulfilled after some time. If Requests cannot be completed, MessageQueue needs to send that message back to the Producer that some requests couldn’t be processed.

Example: WhatsApp



Lets say File upload failed. When there is a failure it goes into DLQ. Maybe they retry after sometime.