

## Notification System

Questions:

Q. What are the type of notifications?

PUSH notification (Mobile push)

SMS notification

EMail Notification

In-App notification

Real time System : Immediately send notification to user.

vs.

Soft Real time system :

We try to send Notification as soon as possible.

--> there can be exceptions.

--> system is down

--> server is busy.

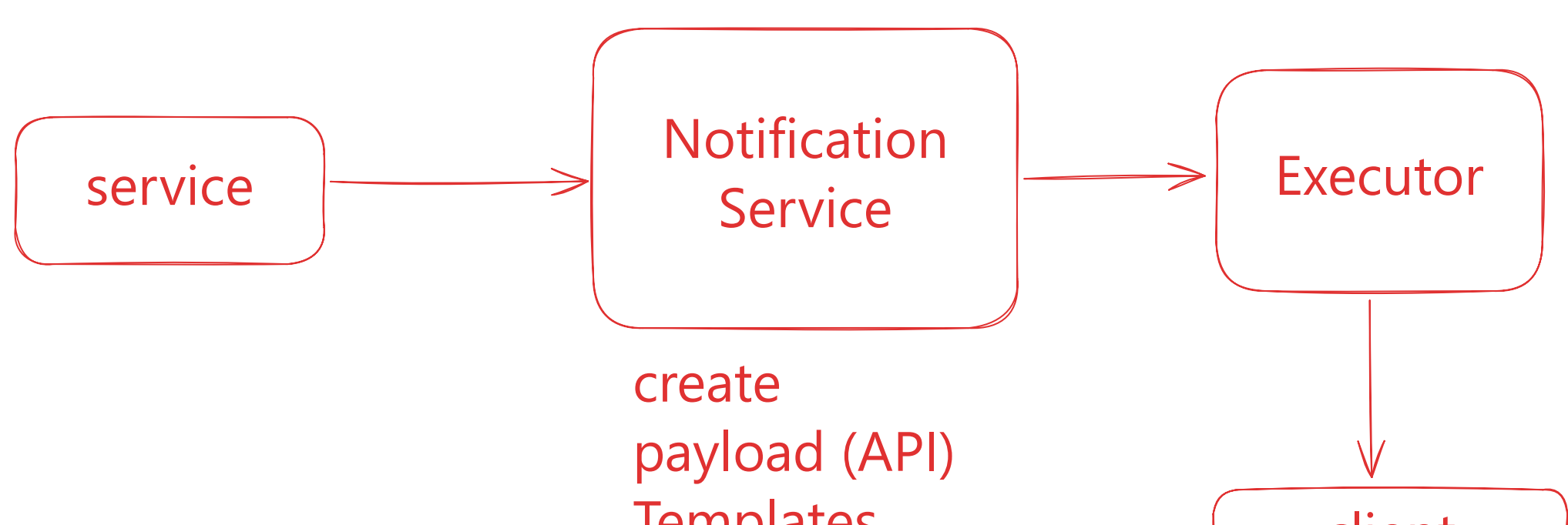
Q. Real time/ Soft real system?

Soft real time.

Q. Device type?

IOS, Android, Web(windows, macos)

Core Entities:



Back of the Envelope calculations

QPS: (Read / write)

Storage Unit:

Bandwidth Calculations:

Assumptions:

Daily Active users : 1 Million

how many notifications are to be sent :

1 user --> 10 notifications.

Total notifications = 1 million \* 10 = 10 million/day

Storage Unit:

1 notification = 200 bytes (content, timestamp, userId, notificationId etc)

total = 10 million \* 200 byte  
=  $10 * 2^{20} * 0.2 * 10^3$   
=  $10 * 2^{20} * 0.2 * 2^{10}$   
=  $2^{30} * 2$   
= ~2GB/ day

Notification store for 1 month = 2GB \* 30 days  
= ~60GB

Bandwidth Calculations:



Bandwidth :

The max amount of data  
that can be transmitted through  
the network in a given time.

How busy our server will be?

Calculations:

Notification payload : 1KB (upload/download)  
headers, Body, request param etc.

DAU = 1 million

Total notifications/day = 10 Million

1 KB \* 10 Million  
=  $1 * 2^{10} * 10 * 2^{20}$   
10GB/day

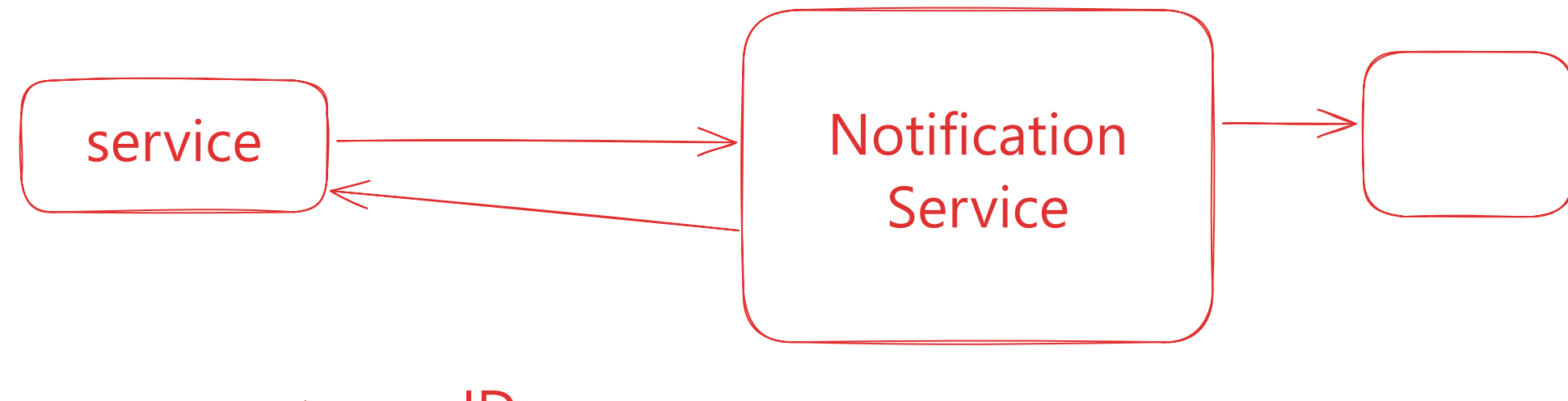
Bandwidth = 10 million \* 1 KB / 86,400 seconds  
=  $10 * 2^{20} * 1 * 2^{10} / 86.4 * 2^{10}$   
=  $0.115 * 2^{20}$   
=  $115 * 10^{-3} * 2^{20}$   
=  $115 * 2^{-10} * 2^{20}$   
= ~ 115KB/sec

Server :

server scaling

Server RAM increase.

API Interactions:



user --> userID.

SQL

User :

userId, Name, mobile...

1. To retrieve user information

GET : /fetchuser/{{userId}}

Response:

200 ok

```
{
  "userId" : 101,
  "userName" "Aditya",
  "notificationPermissions" : {
    "PUSH" : true,
    "Email " : true,
    "SMS" : false
  }
}
```

2. Send Notification

POST : /sendNotification/{{userId}}

Body :

```
{
  "from" : null,
  "to" : "202",
  "subject" : "",
  "content" : "",
  "type" : "push/email/sms"
}
```

Response:

200 ok

```
{
  "status" : sent
  "notificationId" : "301"
}
```

3. To fetch notification history. (optional)

GET : /fetchAllNotifications

Queryparam : filters :

date : from, to

Type : email/sms etc.

Response:

200 ok

```
[
  {
    "status" : "sent",
    "content" : ""
  },
  {
    "status" : "sent",
    "content" : ""
  }
]
```

Core Entities

BEC

Api flow

Basic Design:

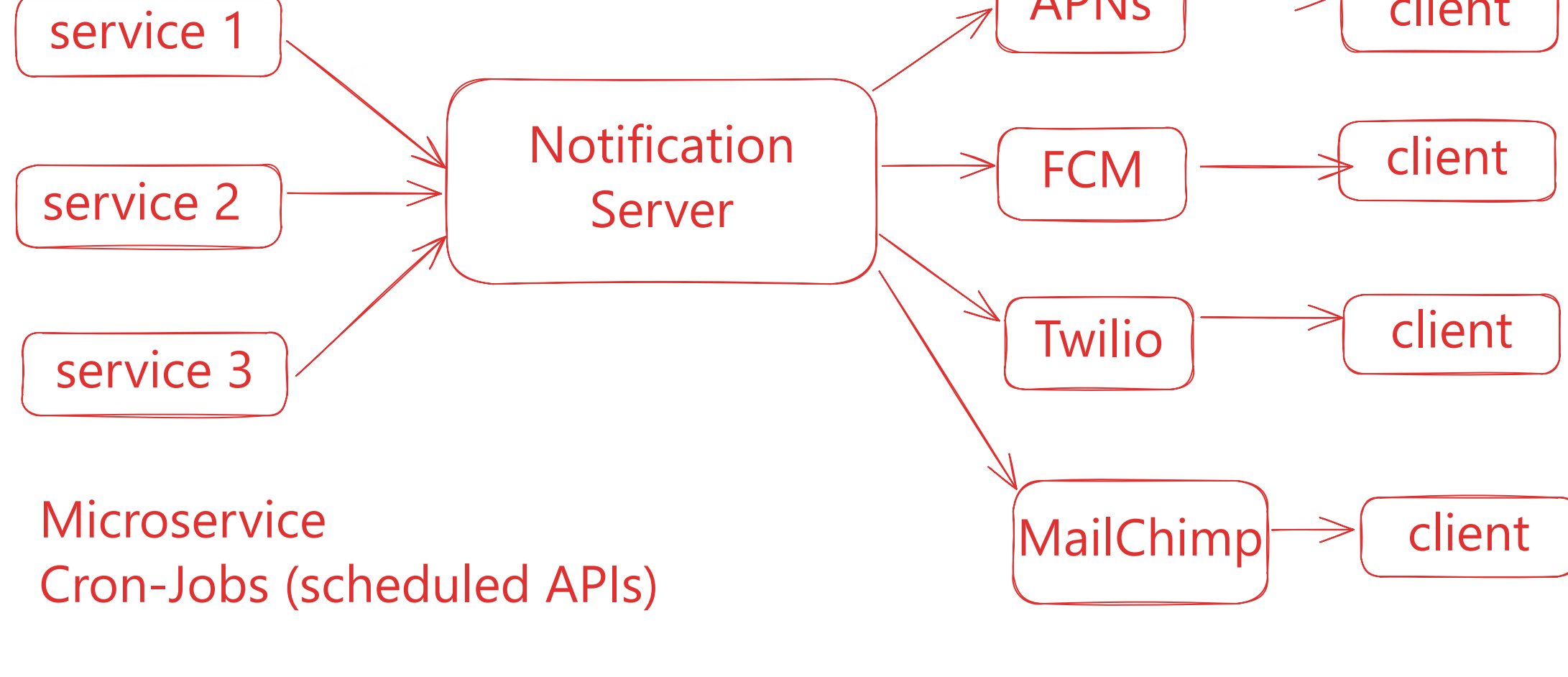
Push Notifications --> Mobile

IOS (Apple) --> APNs (Apple push Notifications)

Android --> FCM (Firebase Cloud Messaging)

SMS Notification --> Twilio, nexmo etc

Email --> MailChimp, Sendgrid etc



Microservice

Cron-Jobs (scheduled APIs)

Flow:

1. Any service (microservice, cronjobs) trigger Notifications and send it to our notification server.

2. The notification server has APIs that builds up the payload, for 3rd party services(APIs) to send the requested notifications.

3. Third party service actually send the notification to the client.

Problems(issues) with initial design:

1. SPOF (Single Point of Failure)

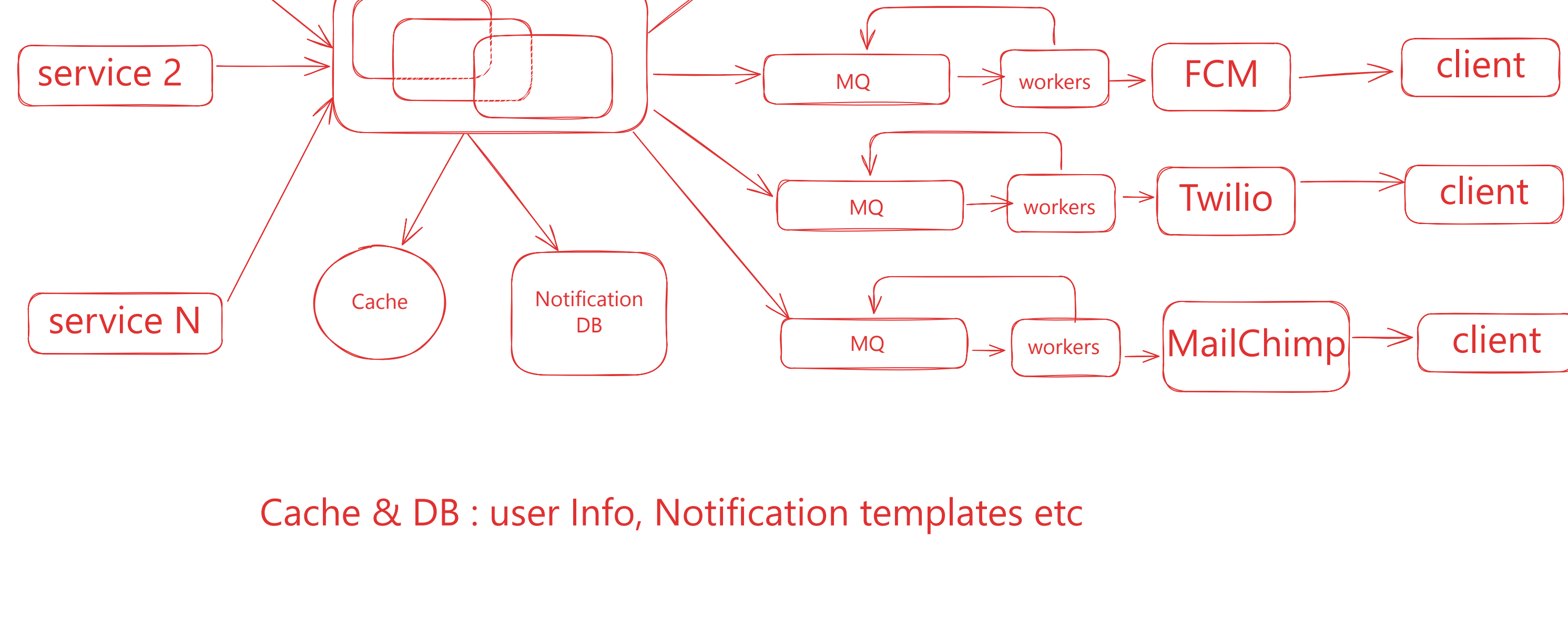
I need to scale the Notification server.

2. hard to scale :

3. Retry mechanism missing.

4. Tight coupling (synchronous)

Shift to async.



Cache & DB : user Info, Notification templates etc

MQ : Removed dependency bw 3rd party APIs and our notification server. Service is now Async

Improvements:

1. Notification Templates

2. Notification settings on user level :

user can turn on/off a particular type of notification

3. Rate Limiting

4. API authentication

5. Monitoring Logic

6. Notification Log.

