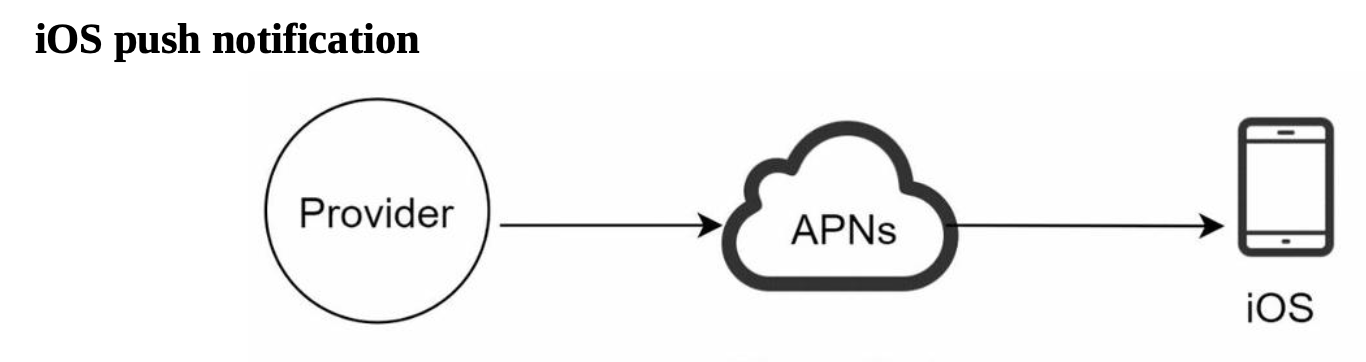
**NOTIFICATION SYSTEM**

A notification system serves to deliver messages or alerts to users based on specific events or criteria. The goal is to inform, alert, or engage users effectively without overwhelming them or the system itself.

Basically there are 3 types of notification formats:

* Mobile push notification
* SMS message
* Email

For Androids it’s called **Android Push Notifications** and For iOS it is called **iOS push notification**.



* **Provider**: A provider builds and sends notification requests to Apple Push Notification Service (APNS). To construct a push notification, the provider provides the following data:

• **Device token:** This is a unique identifier used for sending push notifications.

• **Payload**: This is a JSON dictionary that contains a notification’s payload.



* **APNS**: This is a remote service provided by Apple to propagate push notifications to iOS devices.
* **iOS Device**: It is the end client, which receives push notifications.

**Android push notification**

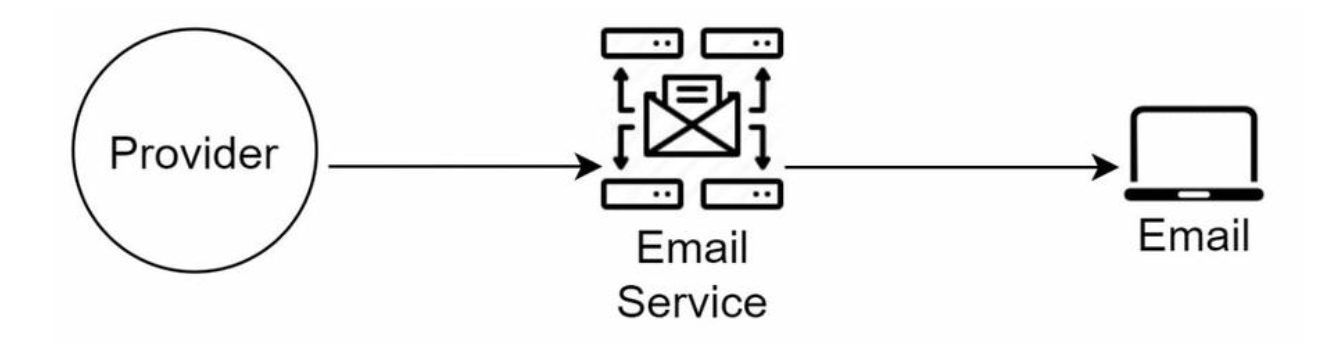
Android adopts a similar notification flow. Instead of using APNs, Firebase Cloud Messaging (**FCM**) is commonly used to send push notifications to android devices.

**SMS message**

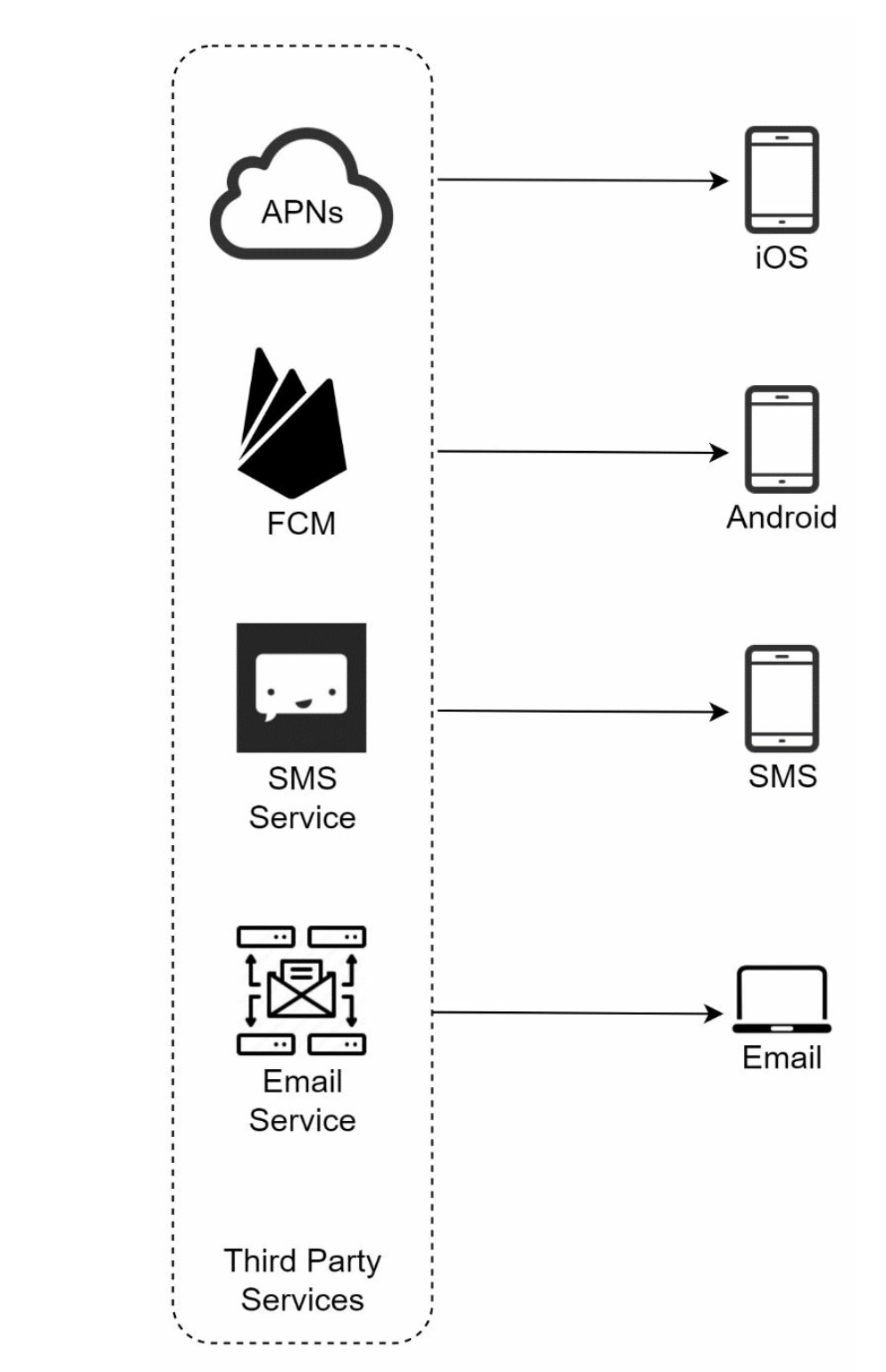
For SMS messages, third party SMS services like **Twilio**, **Nexmo**, and many others are commonly used. Most of them are commercial services.

**Email**

Although companies can set up their own email servers, many of them opt for commercial email services. **Sendgrid** and **Mailchimp** are among the most popular email services, which offer a better delivery rate and data analytics.

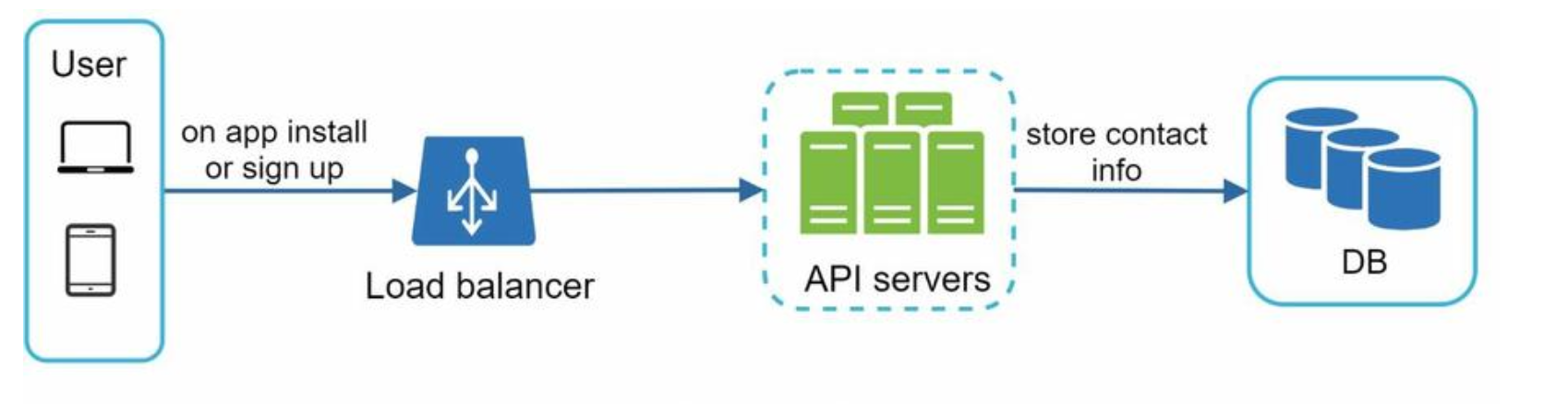


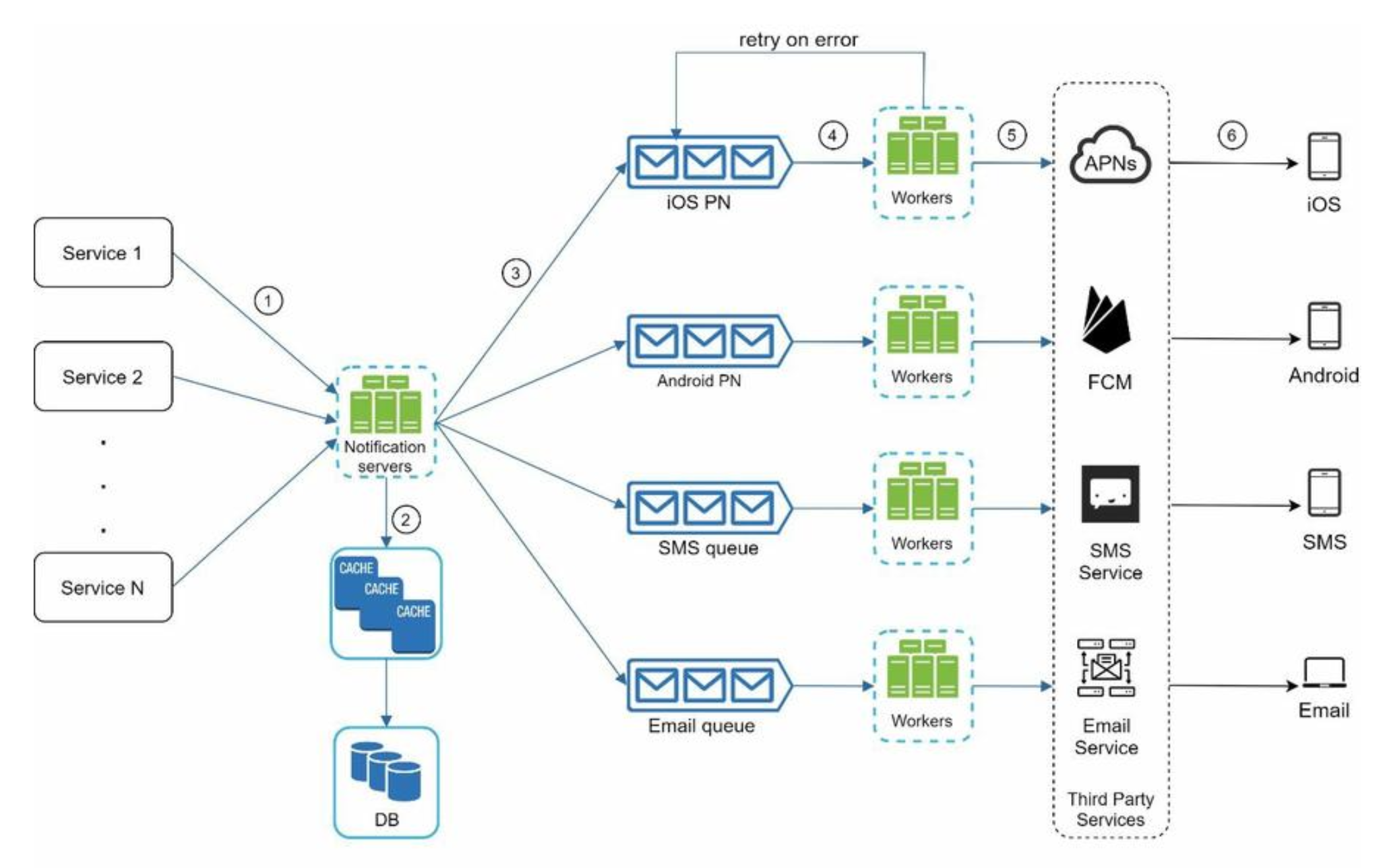
**Design after including all the third-party services:**



**Contact info gathering flow**

To send notifications, we need to gather mobile device tokens, phone numbers, or email addresses. When a user installs our app or signs up for the first time, API servers collect user contact info and store it in the database.



**High-level design**   
  


**Service 1 to N**: They represent different services that send notifications via APIs provided by notification servers.

**Notification servers**: They provide the following functionalities:  
• Provide APIs for services to send notifications. Those APIs are only accessible internally

or by verified clients to prevent spams.  
• Carry out basic validations to verify emails, phone numbers, etc.  
• Query the database or cache to fetch data needed to render a notification. • Put notification data to message queues for parallel processing.

Here is an example of the API to send an email:

*POST https://api.example.com/v/sms/send*

Request body



**Cache**: User info, device info, notification templates are cached. **DB**: It stores data about user, notification, settings, etc.

**Message queues**: They remove dependencies between components. Message queues serve as buffers when high volumes of notifications are to be sent out. Each notification type is assigned with a distinct message queue so an outage in one third-party service will not affect other notification types.

**Workers**: Workers are a list of servers that pull notification events from message queues and send them to the corresponding third-party services.

**How every component works together to send a notification:**

1. A service calls APIs provided by notification servers to send notifications.

2. Notification servers fetch metadata such as user info, device token, and notification setting from the cache or database.

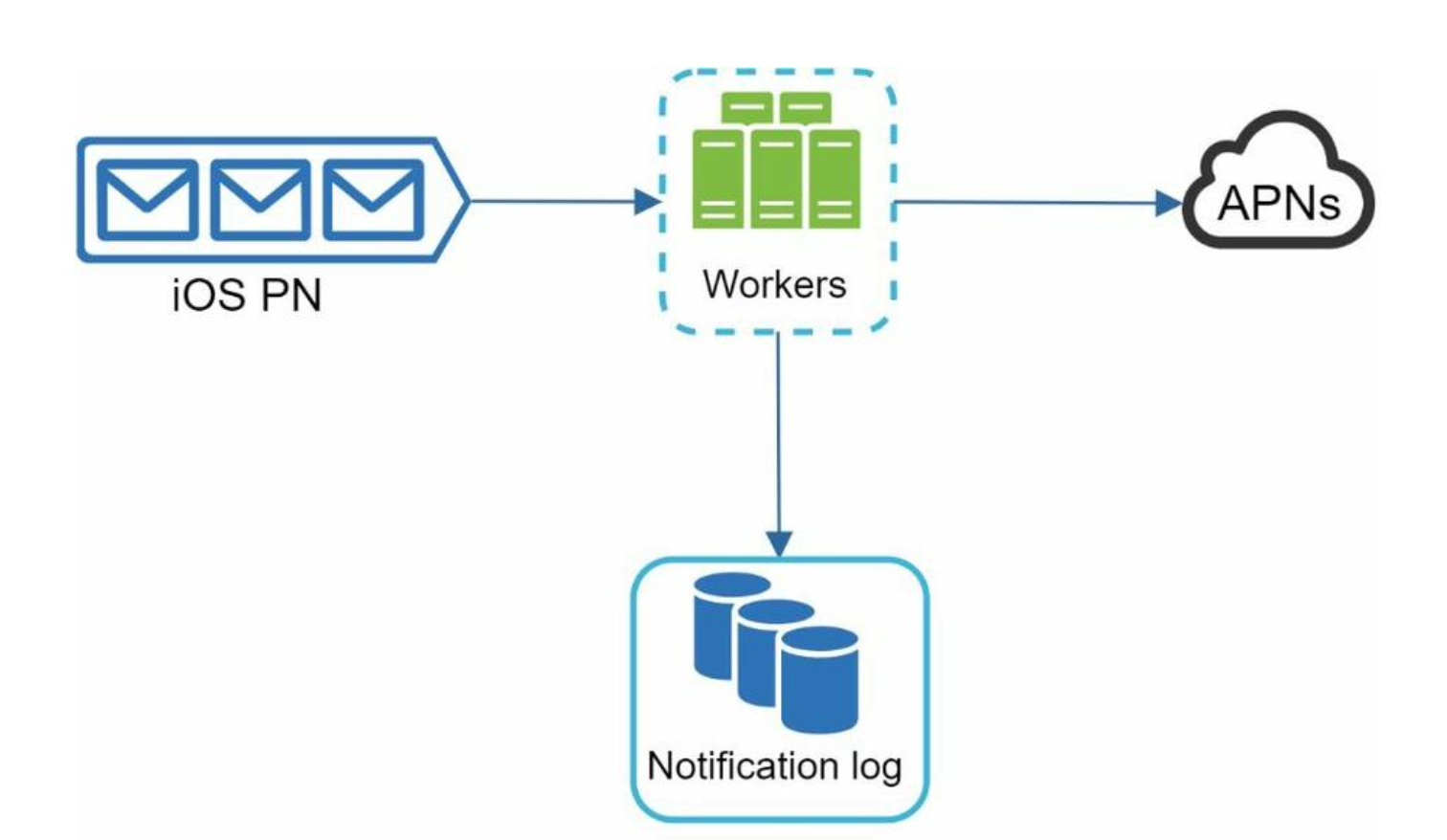
3. A notification event is sent to the corresponding queue for processing. For instance, an iOS push notification event is sent to the iOS PN queue.

4. Workers pull notification events from message queues. 5. Workers send notifications to third party services.  
6. Third-party services send notifications to user devices.   
  
  
  
**Reliability**

We must answer a few important reliability questions when designing a notification system in distributed environments.

**How to prevent data loss?**

One of the most important requirements in a notification system is that it cannot lose data. Notifications can usually be delayed or re-ordered, but never lost. To satisfy this requirement, the notification system persists notification data in a database and implements a retry mechanism. The notification log database is included for data persistence

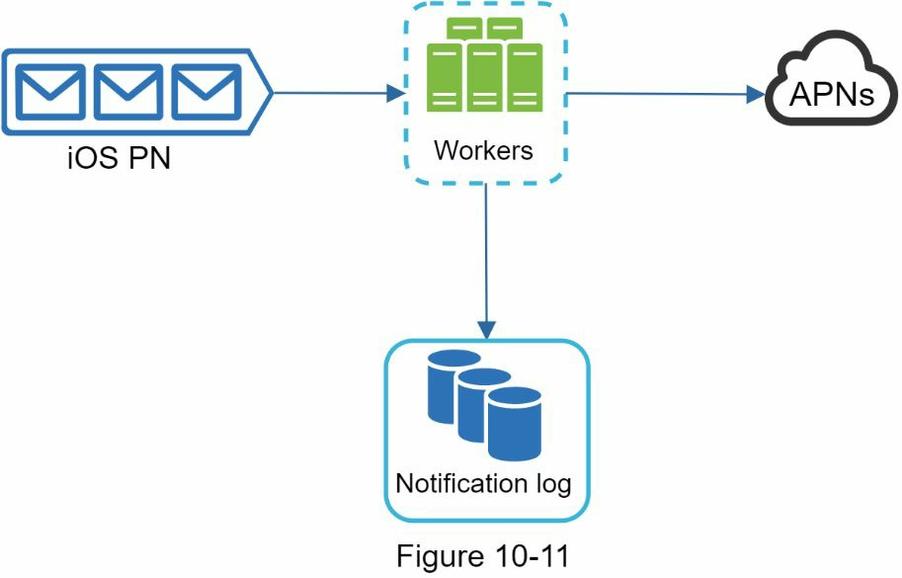


**Will recipients receive a notification exactly once?**

The short answer is no. Although notification is delivered exactly once most of the time, the distributed nature could result in duplicate notifications. To reduce the duplication occurrence, we introduce a dedupe mechanism and handle each failure case carefully. Here is a simple dedupe logic:

When a notification event first arrives, we check if it is seen before by checking the event ID. If it is seen before, it is discarded. Otherwise, we will send out the notification.

**Notification template**



A large notification system sends out millions of notifications per day, and many of these notifications follow a similar format. Notification templates are introduced to avoid building every notification from scratch. A notification template is a preformatted notification to create your unique notification by customizing parameters, styling, tracking links, etc.

**Notification setting**

Users generally receive way too many notifications daily and they can easily feel overwhelmed. Thus, many websites and apps give users fine-grained control over notification settings. This information is stored in the notification setting table

**Rate limiting**

To avoid overwhelming users with too many notifications, we can limit the number of notifications a user can receive. This is important because receivers could turn off notifications completely if we send too often.

**Retry mechanism**

When a third-party service fails to send a notification, the notification will be added to the message queue for retrying. If the problem persists, an alert will be sent out to developers.

**Security in push notifications**

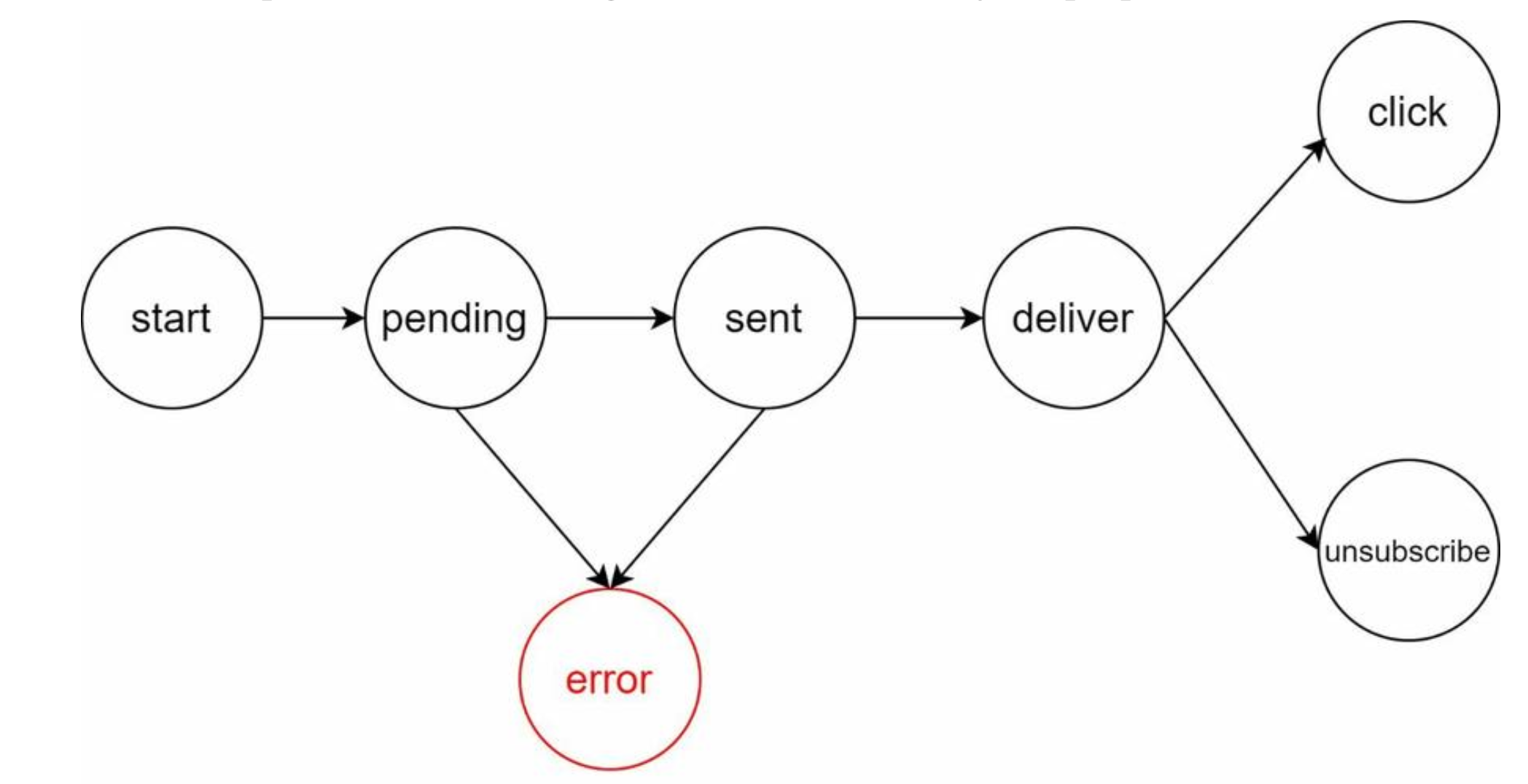
For iOS or Android apps, appKey and appSecret are used to secure push notification APIs . Only authenticated or verified clients are allowed to send push notifications using our APIs.

**Monitor queued notifications**

A key metric to monitor is the total number of queued notifications. If the number is large, the notification events are not processed fast enough by workers. To avoid delay in the notification delivery, more workers are needed.

**Events tracking**

Notification metrics, such as open rate, click rate, and engagement are important in understanding customer behaviors. Analytics service implements events tracking. Integration between the notification system and the analytics service is usually required.



**Updated design**

