Push Notification

Arnav Gupta

December 10, 2024

Contents

1 Motivation

In client-server architecture, client makes request and server sends responses, so client must start all interactions.

Helpful for server to start an interaction when some interaction is caused by another user's action or some timed event.

Possible to notify users by sending emails, but these are limited. Can send links o apps, but cannot directly interact with your app.

Want to be able to send request to a client app.

Client cannot implement a REST API since it is not easily reachable since:

- IP addresses change or can be private

- device/network might have a firewall
- client does not control DNS server
- may be powered off or out of radio contact
- app might not be running

2 Push Notifications

Push notifications: hacks to send messages to clients

Client's OS makes 1 connection for all apps and services send notifications through 3rd party.

2.1 Smartphones

Location registration used for mobile push notifications. Apple and Google run a **push notification service** (PNS) for apps on their OS.

Whenever phone gets new IP address, OS opens long-lived connection to the PNS. PNS stores user id and IP address.

Apps cannot contact a user's device directly, so they send user notifications to the PNS, which relays the message to the user's current P address.

OS can show notification even if app is not running.

On iOS, for privacy, different apps have different user IDs (device tokens).

To deal with NAT, OS sends keep-alive messages, where just 1 port is needed for all apps.

2.1.1 Device Registration

Every time devices connect to the network, the OS creates a long-lived connection to the PNS:

1. app registers for notifications
2. PNS returns a unique push ID
3. app sends push ID to its backend service, which stores user's push ID in a DB for later use
4. wait for the backend to have a notification for the user

5. after the backend gets the push ID for that user from its DB, it sends a notification request to the PNS with push ID
6. PNS uses push ID to identify the long-lives connection to the client and relay the notification

2.2 Web Browsers

Designed to pull data from servers, but modern application desire updates to be pushed from service.

Client can make repeated requests for new data (polling), but poor solution that requires tradeoff between latency and network overhead.

Websockets preferred modern solution, and long-polling before websockets.

2.2.1 Websockets

Websocket: long-lived, bi-directional network connection, similar to TCP socket but available in JS code in browser

JS app creates websocket connection to server. Client can then send API requests through websocket and responses come back through websocket.

Connection remains open.

Server can send messages at any time, independent of client requests.

2.2.2 Long-Polling

Client sends HTTP request. Server waits, then sends a response only when new data is ready. If no data available within 60 seconds, send an empty response. Client then makes another long-polling request (infinite loop).

Client instantiates request and server controls when response is sent.

Sever always has 1 outstanding request from client available to send data.

Periodic requests every 60 seconds are wasteful since periodic gap in service.

3 Challenges

Challenge is to find one long-lived connection to the client.

Network socket (connection) is tied to a single IP address.

Notifications originating from anywhere in the large, distributed system must be somehow directed to the 1 appropriate notification server instance that the client is connected to.

3.1 Notification Service

To solve the problem, notifications are often a separate microservice.

Clients connect themselves by either:

- opening a websocket
- making an API request providing a push ID usable on APN or GCM

In both cases, user's location is stored in a DB.

Other microservices send notifications through API calls. Implementation looks up connection location and relays message.

3.2 API Gateway

Clients have long-lived websocket connections to gateway.

Requests are handled by Serverless Functions (lambdas) and when connection is established, save connection ID.

Later, use connection ID to push data to clients.