Introducing OpenPush

A Free, Decentralized Push Messaging Framework for Android

Marcus Hoffmann - bubu@bubu1.eu

About Me

Hi, I'm Marcus!

Free software developer from Berlin.

Working on a free Android ecosystem.

F-Droid core contributor and maintainer.

Push Notifications

- One of the key missing pieces for apps on **F-Droid**
- Server sending **unsolicited** content to an app on your phone
- Used for ~~SPAM~~ marketing purposes
- But also essential for any kind of **instant messaging** or **VoIP** client

How?

Q: How do you send unsolicited content?

How?

Q: How do you send unsolicited content?

A: It's not *really* unsolicited. The client keeps a connection open.

How?

Q: How do you send unsolicited content?

A: It's not *really* unsolicited. The client keeps a connection open.

Problem: Keeping a connection open requires maintenance (**energy**).

How?

Q: How do you send unsolicited content?

A: It's not *really* unsolicited. The client keeps a connection open.

Problem: Keeping a connection open requires maintenance (**energy**).

Keeping an open connection for every app is not a great idea.

Status Quo

- Currently there's **Firebase Cloud Messaging** (FCM), formerly **Google Cloud Messaging** (GCM)
- Firebase is a **proprietary platform** owned by Google
- Requires **Google Play Services** being present on the device
- AND requires each app to include a **proprietary library** to receive push notifications.

Status Quo

- Currently there's **Firebase Cloud Messaging** (FCM), formerly **Google Cloud Messaging** (GCM)
- Firebase is a **proprietary platform** owned by Google
- Requires **Google Play Services** being present on the device
- AND requires each app to include a **proprietary library** to receive push notifications.
    - All your messages are sent to to google
    - Your FOSS apps **aren't FOSS** anymore
    - Riot, Conversations, Firefox, Jami, RocketChat, Nextcloud, ...
    - No **decentralization** possible

Decentralized Systems and FCM

- Sending messages through FCM requires a developer key
- This key is tied to an APK

Decentralized Systems and FCM

- Sending messages through FCM requires a developer key
- This key is tied to an APK

All push messages from self-hosted services need to go through *another* centralized instance which holds the FCM key.

Decentralized Systems and FCM

- Sending messages through FCM requires a developer key
- This key is tied to an APK

All push messages from self-hosted services need to go through *another* centralized instance which holds the FCM key.

Or you need to distribute your own APKs as a platform host.

Alternatives?

- Everyone builds their more or less reliable custom solution
- Some protocols can do it reasonably well in-band (like XMPP)
- But Android makes it increasingly hard to run background tasks
- Battery life is still horrible

OpenPush Goals

1. **Free Software**: Including server, client and client library parts
2. **Decentralized**: You can run your own instance.
3. **User is in Control**
    - Smartphone user chooses pusherver instance.
    - User decides which apps are allowed to subscribe to push notifications.
4. **No developer key required**
    - Used by Google for **accounting** and possibly abuse mitigation.
    - Provide a service to users not app developers, so we don't need accounting there
    - Find other methods for abuse mitigation

Architecture

- **App**
  - Wants to receive push notifications.
  - Talks to a **webservice**.

Architecture

- **App**
  - Wants to receive push notifications.
  - Talks to a **webservice**.
- **Webservice**
  - Receives a **pushtoken** from an **app**.
  - Sends pushes to specific instances of an **app** via a **pushserver**.

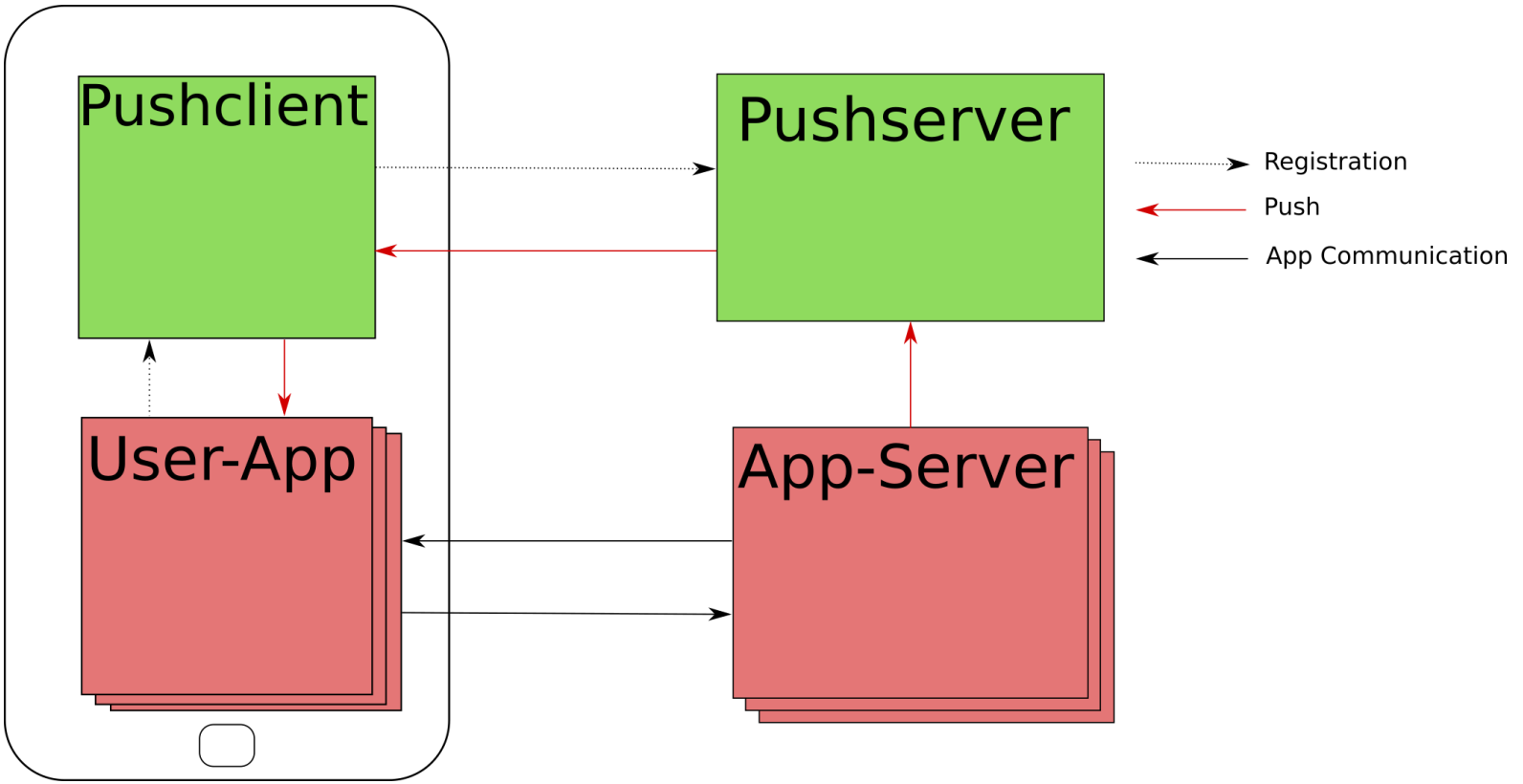
Architecture

- **App**
  - Wants to receive push notifications.
  - Talks to a **webservice**.
- **Webservice**
  - Receives a **pushtoken** from an **app**.
  - Sends pushes to specific instances of an **app** via a **pushserver**.
- **Pushserver**
  - Provides an API for a **pushclient** to register for push notifications
  - Hands out a **pushtoken** for every **app** registration
  - Provides the API for receiving push messages from a **webservice**

Architecture

- **App**
  - Wants to receive push notifications.
  - Talks to a **webservice**.
- **Webservice**
  - Receives a **pushtoken** from an **app**.
  - Sends pushes to specific instances of an **app** via a **pushserver**.
- **Pushserver**
  - Provides an API for a **pushclient** to register for push notifications
  - Hands out a **pushtoken** for every **app** registration
  - Provides the API for receiving push messages from a **webservice**
- **Pushclient**:
  - Handles **app** registration
  - Keeps an open connection to a **pushserver**. Distributes push messages to **apps**.

Architecture



Current Status

- **Pushserver OpenAPI Spec ===> ~Done ✔**

Current Status

- **Pushserver OpenAPI Spec ===> ~Done ✔**
- **Server implementation in Python ===> Done ✔**

Current Status

- **Pushserver OpenAPI Spec ===> ~Done ✔**
- **Server implementation in Python ===> Done ✔**
- **Android Client implementation ===> WiP**

Current Status

- **Pushserver OpenAPI Spec ===> ~Done ✔**
- **Server implementation in Python ===> Done ✔**
- **Android Client implementation ===> WiP**
- **Android Client library ===> WiP**

Current Status

- **Pushserver OpenAPI Spec ===> ~Done ✔**
- **Server implementation in Python ===> Done ✔**
- **Android Client implementation ===> WiP**
- **Android Client library ===> WiP**
- **Integration into other Systems ===> ToDo!**

Implementation details

Pushclient <--> Pushserver

Currently using **Server-Sent Events**.

Deliberately **transparent** to users of OpenPush, so we can experiment with different protocols.

Implementation details

Pushclient <--> Pushserver

Currently using **Server-Sent Events**.

Deliberately **transparent** to users of OpenPush, so we can experiment with different protocols.

Pushclient

Currently a **standalone Android app**.

Possible integration with **MicroG** in the future.

Connection kept alive by using a **foreground service** (or being a system app).

Client library communicates via **Android IPC** mechanisms (bound service and broadcast intents).

Outlook

- **Integration** into server and client apps (Nextcloud, Matrix, RocketChat)

Outlook

- **Integration** into server and client apps (Nextcloud, Matrix, RocketChat)
- Adding **E2EE**
  - From Webservice to a users phone.
  - Essential if you want to to host public pushervers

Outlook

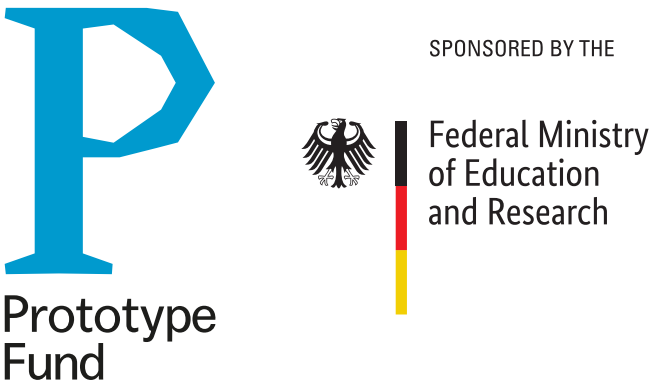
- **Integration** into server and client apps (Nextcloud, Matrix, RocketChat)
- Adding **E2EE**
  - From Webservice to a users phone.
  - Essential if you want to to host public pushervers
- Experimenting with **different transports**

Outlook

- **Integration** into server and client apps (Nextcloud, Matrix, RocketChat)
- Adding **E2EE**
  - From Webservice to a users phone.
  - Essential if you want to to host public pushervers
- Experimenting with **different transports**
- Having existing systems provide the **OpenPush APIs**
  - I.e. Pusherver API provided by an **XMPP/Matrix** server
  - Apps register for push notifications with your existing **XMPP/Matrix** app

Acknowledgements

The original development of the OpenPush project was sponsored by the German Federal Ministry of Education and Research through the Prototype Fund program.



Find more info at https://bubu1.eu/openpush