Web Push
Background and Overview

Dan DRUTA

IETF 91

A few use cases

- Notification delivered when browser is not running

  Alice uses her browser on her laptop to chat with her mom overseas. She goes online and marks her status available. Then she closes her browser window to work on a presentation due next week. Her mom wants to tell her the great news about her cousin's new baby and calls her. Alice receives a notification that her mom wants to talk and answers.

- WebRTC call

  Bob uses Ringo's STAR WebRTC service on his mobile device. Bob calls Alice at work and leaves a voice mail for her to call him back. Alice comes back from her meeting and calls Bob. Bob receives an incoming notification. He can clearly identify the caller as Alice and answers.

Why Web Push

Most of the web based solutions today work under the premises that parties involved are online and connected to the same web application at the same time. With the addition of real-time capabilities to the web, there has to be a way for servers to send Web App specific notifications efficently and reliably.

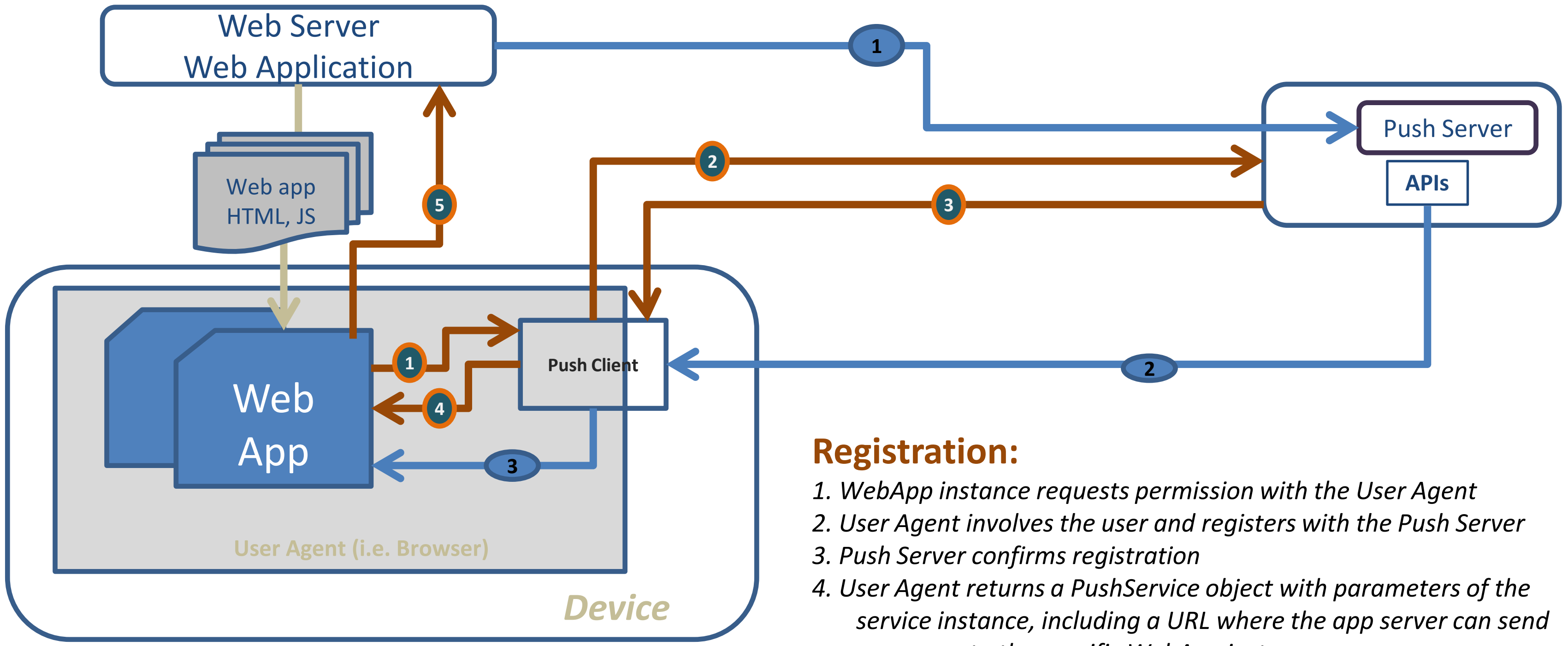
- Push enabled solutions should have the ability to:
  - Provide unsolicited notifications once the user registered for them
  - Make notifications available to the user irrespective if the app is running, in focus or not.
  - Provide notifications irrespective of the browser, device type or network environment
- In addition to that, Web Push Enabled solutions should:
  - Reliably deliver notifications
  - Minimize latency
  - Efficiently use network recourses
  - Optimize signaling to maximize battery life on the client device

A brief history

- Presented the need for push notifications in support of WebRTC: http://lists.w3.org/Archives/Public/public-webrtc/2012Jan/att-0083/WebRTC_PushNotifications.pdf
- The need was acknowledged and it was recognized that push notifications have a broader applicability so the group requested W3C WebApps to take on this work as the API must be able to be used for other types of solutions
- WebApps has been working on this. The scope is limited to the API surface and leaves the delivery protocols out: http://www.w3.org/TR/push-api/

This is why we're here!

How it works



Registration:

1. WebApp instance requests permission with the User Agent
2. User Agent involves the user and registers with the Push Server
3. Push Server confirms registration
4. User Agent returns a PushService object with parameters of the service instance, including a URL where the app server can send messages to the specific WebApp instance.
5. WebApp notifies the App server of the Push Service URL

Fulfillment (message delivery):

1. Webapp server requests delivery of a message via the Push service URL.
2. Push Server delivers the message to the User Agent
3. The User Agent delivers the Push message to the specific WebApp associated with the message.