Scenario analysis

RTC call with push message

Shijun Sun

(shijuns@microsoft.com)

Purpose

- Make sure we understand the steps and options in the basic E2E flow of Real-Time Communications (RTC) with push message, i.e. push an “incoming call” notification to the user.
- Identify bottlenecks and open issues.

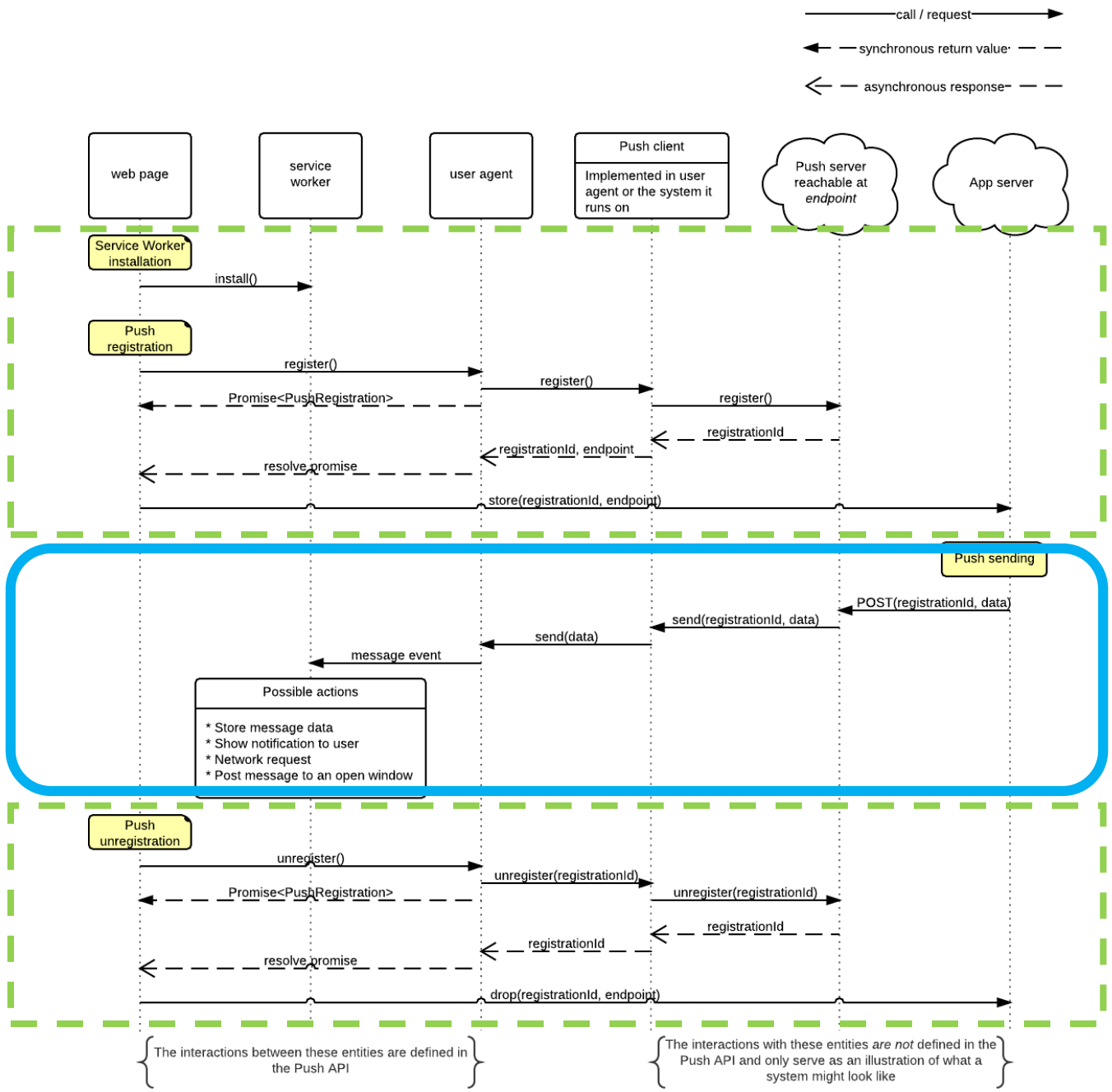
Priorities

- Low latency
- High reliability
- High power efficiency

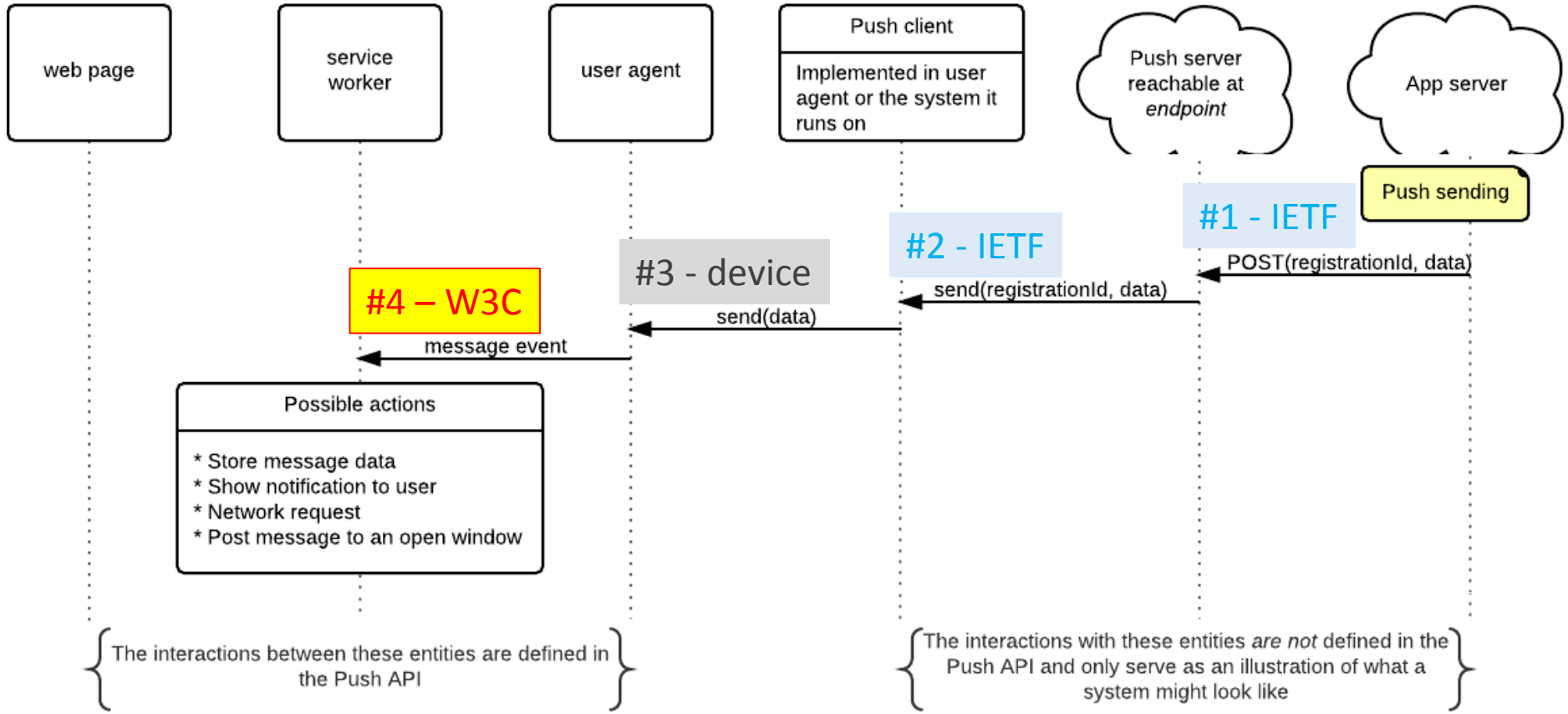
Register – no low latency requirement

Push – this is what we want to talk about

Unregister – no low latency requirement



The Scenario: RTC call with push message



#4. UA dispatch the message

- Current option
  a. UA wakes up the service worker (SW) and dispatch the message to the SW
  b. SW processes the message
  c. SW displays a notification, and optionally renders a ringtone
  d. Upon user click, SW launches the webapp
- Questions
  1. **Latency** – How much latency between “UA receives the message” and “SW displays a notification” to user?
  2. **Reliability** – Can SW stay alive until user click? (issue #85)
  3. **Power Efficiency** – Are long-living SWs a problem to the device? (issue #84)

#4-a. Alternative option

- Proposal
  a. The push client wakes up a lightweight independent background process and pass the message to it
     - The background process can be developed by browser implementers
     - This process only handles push messages
     - The push client automatically wakes up this process when needed
  b. The background process parses the message property to identify predefined action(s)
     - The process only executes a small set of predefined actions: display notification, play ringtone
     - PushRegistraiton can be registered with the predefined action(s)
  c. If predefined actions are identified, the background process displays a notification, and optionally renders a preloaded ringtone
  d. Upon user click, the background process launches the UA which in turn forwards the push message to either a webapp or a SW
- Benefits
  1. **Latency** – minimal latency since message doesn't require the UA to be instantiated before notification happens
  2. **Reliability** – the background process is always available or can always be woke up by the push client
  3. **Power Efficiency** – the background process only runs when push notifications for the browser are received by the push client. It stops running after the push notification is processed. UA only launches upon user click.

References

1. Use cases: http://www.w3.org/2008/webapps/wiki/Push_API
2. Issue #84: Push API should be allowed without dependency on service worker
3. Issue #85: Push API should support SW event.waitUntil