Enhancing DIDComm messaging for mobile environments

By: Ariel Gentile 2060.io
Date: Aug 12, 2022 Version: 1.0

Introduction

By defining its simple yet powerful message routing protocol[1] and the ones built on top of it (such as mediation coordination [2] and message pick up[3]), DIDComm has enabled secure messaging for a wide range of agents, including those that are not online all the time. The resulting system looks like the traditional POP3 e-mail, that allows clients and servers from different vendors to connect and exchange messages.

While this scheme certainly is capable of working in mobile devices, there are some aspects that should be addressed to enhance both user experience, efficiency and interoperability between agents from different vendors. A couple of them will be introduced here to illustrate the work item proposal for RTWoT.

Notification coordination

In mobile world, push notifications are a common usage pattern that is becoming more and more advanced over the years, so it's crucial for mobile agents to be able to receive notifications when there are new messages while they are in background. Finding a transport-agnostic way of managing notifications in agent messaging is challenging for different reasons:

- Notifications rely on the OS the device is running, and require an OS vendor-centralized backend such as FCM for Android and APN for iOS to be in the middle
- Notification behaviour and constraints in terms of throughput and size differ between these systems
- Some of these mechanisms require the notification sender to have access to a certificate or key coming from the app developer: they expect the notification sender to be the mobile app developer, and not any associated third-party. This means that, in practice, there will be some dependency on the mobile app publisher when attempting to send notifications

There are good ongoing efforts within the Aries community to standardize notification coordination mechanism for different native notification systems [4], altough they have not yet resolved entirely these issues and are mostly applicable

[1]Routing Protocol 2.0
[2]Mediation Coordination Protocol
[3]Message Pickup Protocol
[4]Push Notifications APN Protocol, Push Notifications FCM Protocol PR and Push Notifications Expo Protocol PR

to specific cases where mediator and app are in the domain of the same company. More fine-grained notification configuration options could be discussed now that SSI community has some hands-on experience on the first deployments of mobile wallets.

And while the ‘natural’ notification sender would be the mediator, there could be some room for use cases where the message sender wouldn’t want to use any mediator to exchange messages with their recipients: they might internally queue the messages and notify the recipients, so they can pick them up as soon as they are online.

Delegated outbound message queues

In battery-backed devices, every background process and exchanged network byte counts when saving power, and this applies not only to the notifications but also to network connections. If a mobile agent needs to manage individual transport sessions for each of its outbound messages, it could become tricky to maintain when dealing with connectivity issues, as some messages could not probably be delivered and must be retried. In these cases, it could be convenient to rely on a trusted outbound messaging queue where a mobile device can delegate all outbound connections that otherwise would need to manage by itself.

This concern not only apply to DIDComm messaging but also VDR access, as usually it’s costly to have a direct access to ledgers in mobile devices and therefore a trusted proxy is used to operate with it, apart from generic solutions for DID resolution like discussed in Universal DID Resolver.

References