ANR-PMR Health PPML workshop

Towards a hospital-friendly communication module

Marc DAMIE

Wednesday 27th October 2021

1. Context and motivations

2. Communication design

3. Technical solutions

4. Few recommendations

Who am I?

- Computer engineer (graduated from UTC, Compiègne) specialized in Data Mining/Engineering but with professional experience in cybersec.
- This presentation exposes few of the results of the internship I did at Inria (Decentralized Secure Privacy-Preserving Surveying for Mobile Devices)
- Currently: *PhD student* between Inria Lille (Magnet) & University of Twente (Services, Cybersecurity and Safety group)

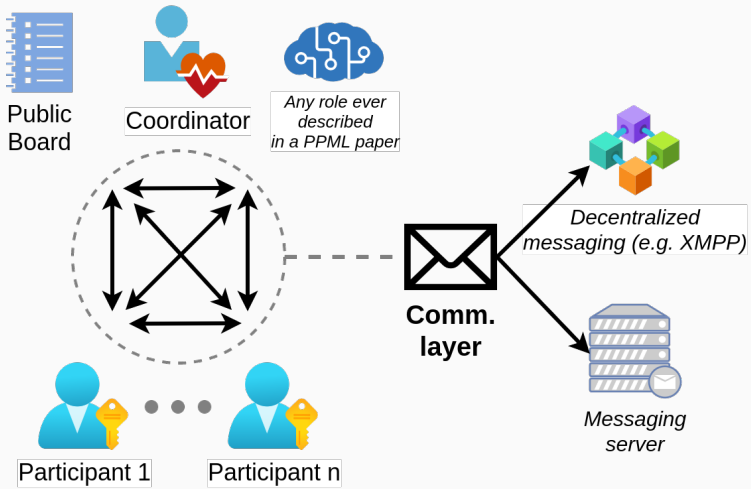
Inria

Tailed library

- Tailed: Trustworthy AI Library for Environments which are Decentralized
- *Goal*: develop a cross-platform framework to deploy federated learning with user trust: learning contracts, verifiability, auditability, etc.
- Communication module studied in the context of Tailed
- $\Rightarrow$ we present our conclusions but *no strong claims*

Inria

Quick vocabulary overview: roles in PPML protocols



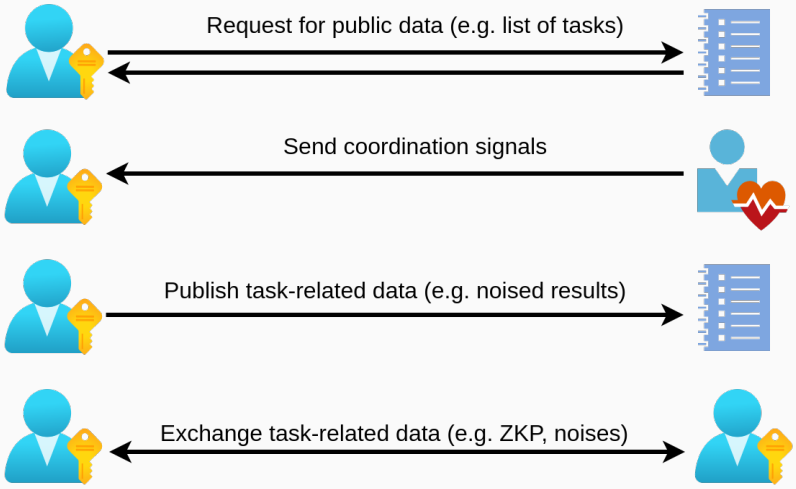
NB: an agent can have several roles
e.g. public board + messaging server

Motivations

- *General goal*: make the communication technically possible between all the agents involved in a PPML protocol [not studied in the ML community]
- *Particular focus*: have a *hospital-friendly* communication system
- ⇒ *constraints*: possibly restrictive firewalls, minimize the permissions needed, avoid deploying something in their DMZ, minimize the setup/configuration cost, not only Linux servers, etc.
- *Perspectives*: build an OS-independent solution. Windows for hospitals but also Android and iOS to work on projects involving mobile users

Inria

Communication requirements



1. Context and motivations

2. Communication design

3. Technical solutions

4. Few recommendations

Communication centralization

- Everything cannot be addressed *easily* via decentralization: identity authority, public bulletin (e.g. to publish learning tasks), etc.
- Distributed ledgers, XMPP, etc. interesting but … messaging server = less costly and *easier deployment* (i.e. doesn't require n independent nodes)
- Centralization $\not\Rightarrow$ Trusted server (public keys, gossip, etc.)
- Communication centralization $\not\Rightarrow$ Learning centralization

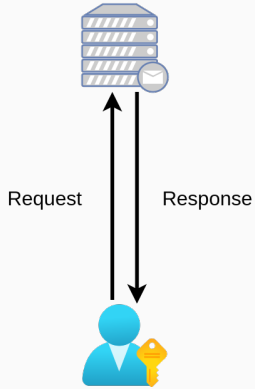
Inria

P2P limitations

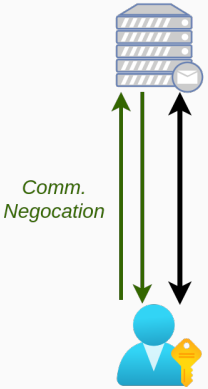
- 1. *Cost*: requires some communications to set up a P2P channel ⇒ not worth it to set up a channel for only one message
- 2. *Trust model*: P2P protocols such as WebRTC require a central server to set up the communication ⇒ if the server is not trusted, requires public key infrastructure (PKI)
- P2P advantage: secure/private against honest-but-curious adversary
- Both P2P and centralized communications require a PKI to be secure against malicious server

Inria

Communication paradigm: request-response vs. bidirectional



Request-response paradigm
(e.g. HTTP)



Bidirectional paradigm
(e.g. WebSockets)

Well-suited for
PPML requirements

1. Context and motivations

2. Communication design

3. Technical solutions

4. Few recommendations

Overview of communication protocols

- We will *present and compare* the following protocols: HTTP polling, Server-sent events, sockets, WebSockets and HTTP2

- Focus on popular and widely implemented *client-server* protocols

Inria

Comparison of the communication protocols: the veterans

- *HTTP polling* | **Pros**: compliant with any kind of (client) security policies | **Cons**: not very optimized
- *HTTP long polling* | **Pros**: compliant with any kind of (client) security policies | **Cons**: not very optimized and quite ”dirty”
- *Server-sent events* | **Pros**: compliant with old systems | **Cons**: outdated AND only allows server to client messages ($\Rightarrow$ should be combined with HTTP)
- *Socket communication* | **Pros**: tailor-made solution | **Cons**: very complex to implement (+ security risks!)

Inria

Comparison of the communication protocols: the newcomers

- *WebSockets* | **Pros**: real-time bidirectional communication, available in most of the major web frameworks | **Cons**: no major WebSockets-specific framework has survived over time ⇒ no guidelines to use it properly

- *HTTP2 (+ gRPC)* | **Pros**: allows request-response as well as bidirectional communication (≈ HTTP1 + WS), gRPC is a rich framework | **Cons**: younger (⇒ less supported for now)

- **Summary**: for convenience purpose, HTTP2 (or WebSockets) seems to be a natural choice. For maximum compliance, HTTP polling is interesting.

Inria

Introduction to asynchronous programming

- Motivation: *optimize waiting times* by creating a sort of concurrency in the execution while avoiding the cost of massive multithreading.
- Useful in *network* (especially bidirec. comm.) and file system interactions.
- *Async-await* paradigm: has a synchronous-like structure with simply ”await” keyword before blocking statements. For example: *await server.connect()*
- Async-await available in: C#, Python, JavaScript, Nim, Rust, C++20, etc.
- Even it looks like sync code, it requires *designing differently* the programs because of concurrency: events won’t always happen in an expected order

Inria

1. Context and motivations

2. Communication design

3. Technical solutions

4. Few recommendations

How to choose a library?

- *Quality*: maturity, reputation, SSL support, Autobahn test result for WebSockets lib, etc.
- *Convenience*: how quick and easy will be the coding? Is there a framework? (protocol implementation $\neq$ framework)
- *Performances*: TechEmpower Framework Benchmarks. Perf.-convenience trade-off $\Rightarrow$ 1M req/sec is cool but not useful to everyone
- Number of protocols supported.
- Only choose a protocol if it has a good library in your language $\Rightarrow$ good theoretical properties are not enough!

Inria

Security considerations

- *SSL/TLS*: the fundamental *must-have* (reminder: prevents eavesdropping)
- *Firewall policies*: some hospitals could filter protocols such as WebSockets. Minimum risk is obviously HTTP. HTTP2/gRPC could be preferred by some hospitals compared to WebSockets.
- *Message encryption*: SSL is not sufficient $\Rightarrow$ find an adequate PKI to encrypt agent-to-agent messages

Inria

Concluding thoughts

- Prefer using only *one communication server*: more optimized and an easier (and more realistic) deployment especially for scenarios involving hospitals
- Prefer supporting *several comm. protocols*: WebSockets, gRPC, HTTP polling $\Rightarrow$ compliant with ”any” firewall.
- *Under development*: OS-independent communication module creating an abstraction of the protocol APIs $\Rightarrow$ user can focus her attention on the ML
- Studied only the communication level, but there are other *open questions*: cryptography, identity, scalability, etc. $\Rightarrow$ will be explored during my PhD

Inria

Questions?