Matrix – прагматический подход к децентрализации

Распределённая система хранения и обмена сообщениями в реальном времени

Matrix – прагматический подход к децентрализации

- Возраст проекта – 1 год (июль 2014)
- Команда разработчиков – ~10 человек
- Состояние – late beta:
  - Спецификация (WIP; достаточна для реализации клиентской части)
  - Эталонная реализация серверной части: Synapse (python2/twisted)
  - Веб-клиент, плагин к WeeChat, клиенты для мобильных платформ, SDK…

Что представляет собой сеть Matrix

- распределённая **база данных** цепочек сообщений
  - хронологически упорядоченная
  - криптографически защищённая (Merkle tree)
  - в конечном счёте согласованная (eventually consistent)
- **совокупность узлов** со свободным участием в сети
- протокол на основе **JSON-over-HTTP(S) API**
  - расширяемый новыми типами событий (сообщений)
  - интеграция с WebRTC как пример такого расширения

идеализм ⟷ **прагматизм**

полный P2P (GNUnet, Tox, …)

есть концепция сервера, но нет центрального (*федерация*)

Matrix и XMPP (Jabber)

X**MPP – Message Passing Protocol.**

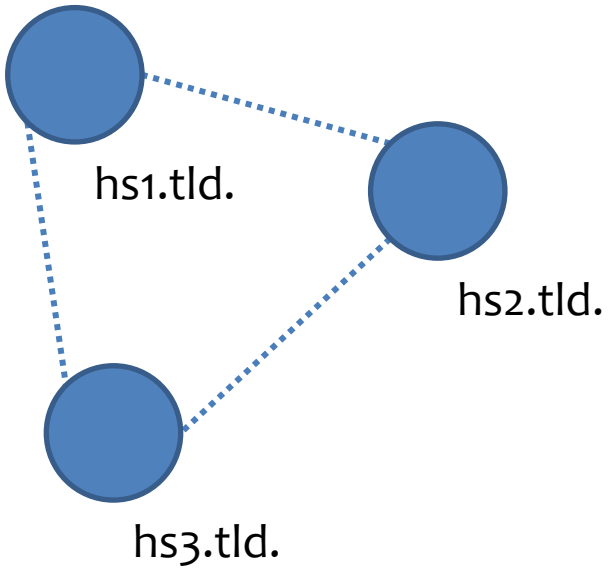
Идейные различия:

- передача сообщений ↔ синхронизация состояния
  - «вечное» хранение истории сообщений
  - распределённые по нескольким серверам конференции
  - принципиальная устойчивость к проблемам со связью
  - полная поддержка нескольких устройств на аккаунте (опять же, исходя из сущности протокола)

Технические различия:

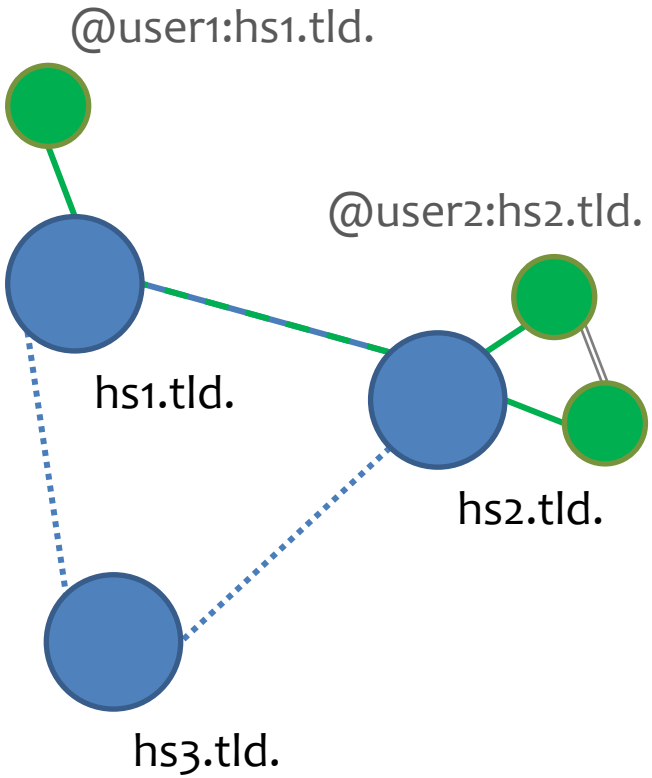
- JSON эффективнее использует место, чем XML
- A/V стандартизировано с самого начала (WebRTC)
- На стороне клиента нет сложной логики *(меньше возможностей для мутаций протокола)*

Архитектура Matrix



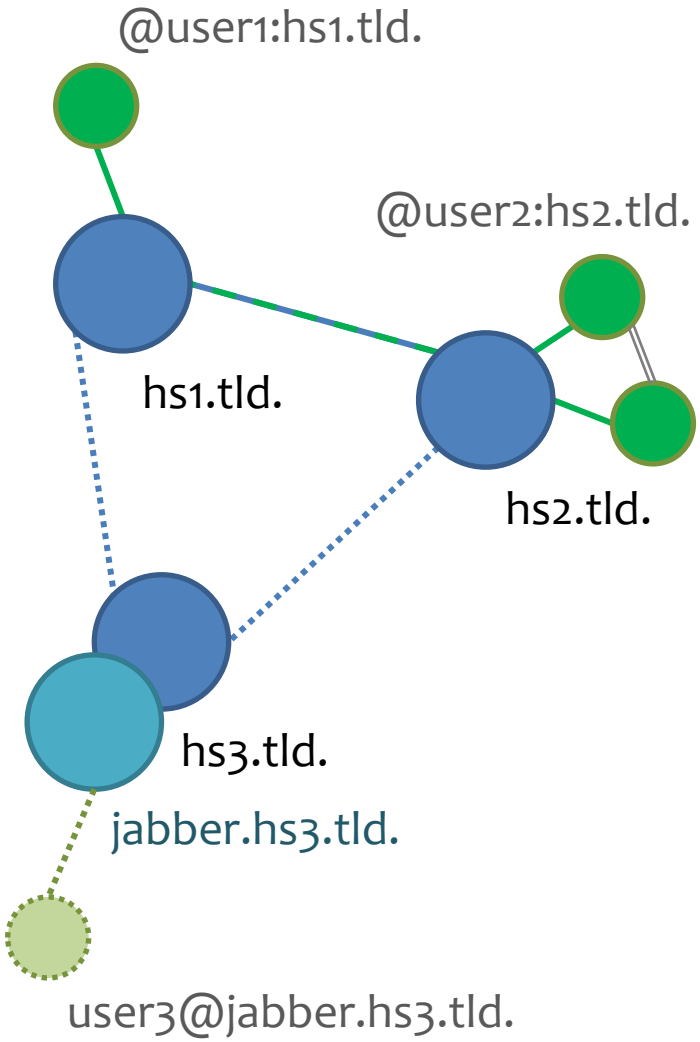
1. **Homeserver** – полноправный узел сети, участвующий (или не участвующий) в федерации
   - хранение профиля пользователя
   - хранение и синхронизация **комнат** (независимых цепочек событий)
     - Merkle tree в хронологическом порядке
     - между серверами передаются вершины и рёбра графа, а не отдельные события
   - состояние комнат распределено по всем участвующим серверам (и только по ним)
     - сейчас – полностью реплицируется
   - связь – HTTPS

Архитектура Matrix



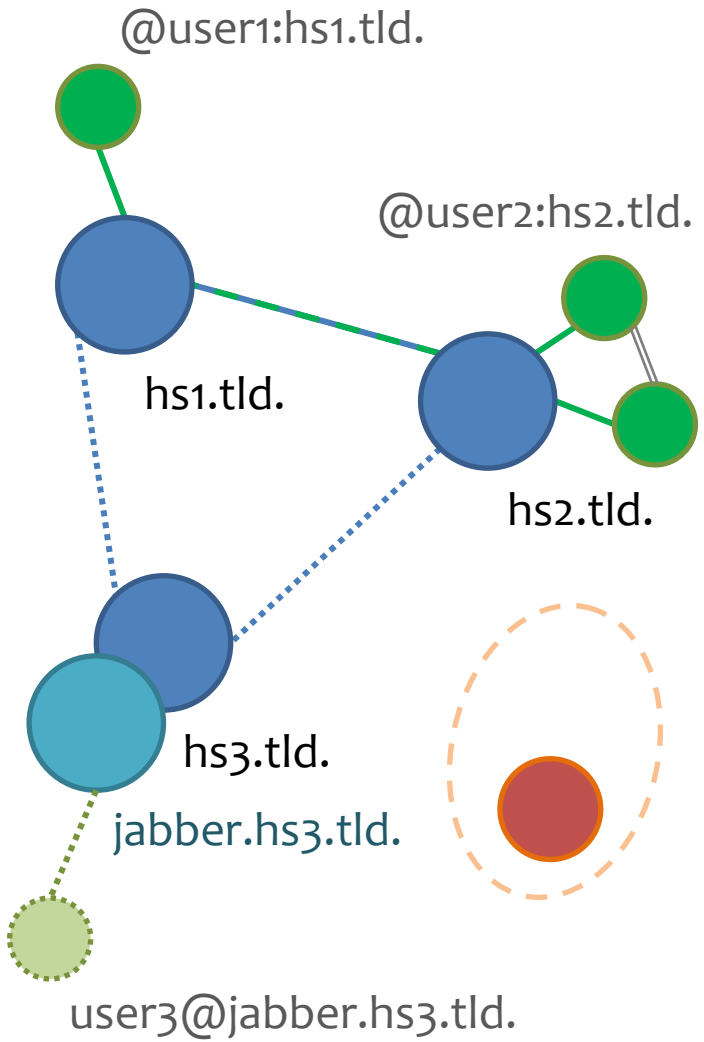
1. **Homeserver** – полноправный узел сети, участвующий (или не участвующий) в федерации
2. **Клиенты** – занимаются только тем, что получают от «своего» сервера поток событий (и отправляют свои)
   - могут быть сколь угодно простыми
   - модель publish/subscribe
   - REST-подобное API ⇒ состояние клиента хранится на клиенте (и больше ничего)
   - опять же, HTTP(S)

Архитектура Matrix



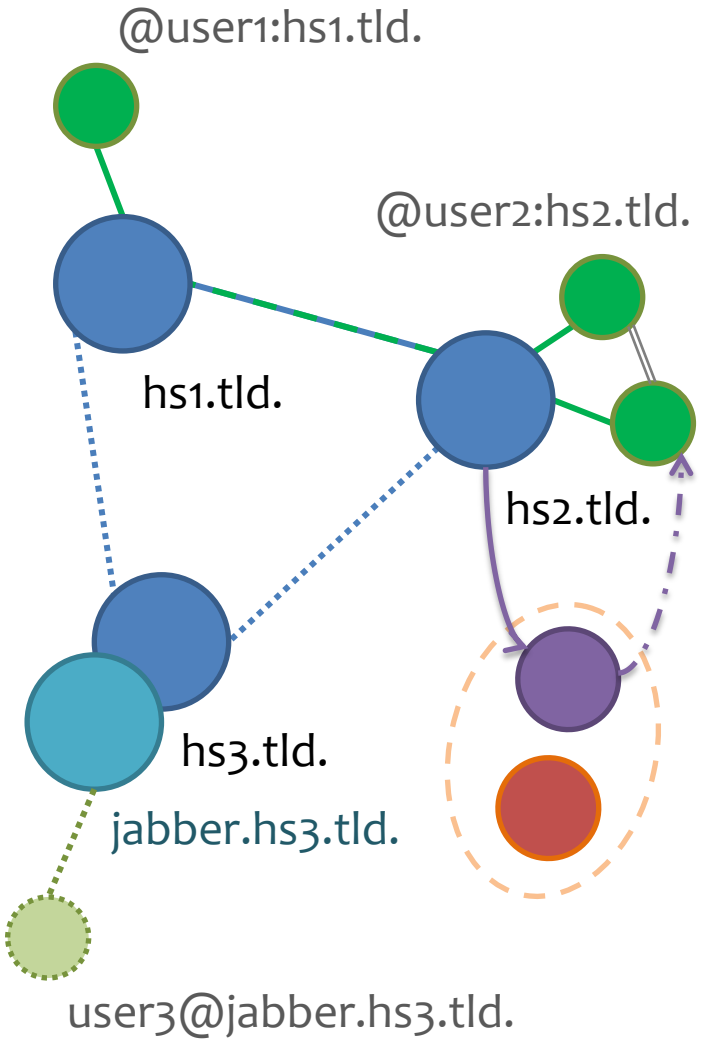
1. **Homeserver** – полноправный узел сети, участвующий (или не участвующий) в федерации
2. **Клиенты** – занимаются только тем, что получают от «своего» сервера поток событий (и отправляют свои)
3. **Application Server** – своего рода произвольное расширение HS
   - например, шлюз в другую сеть (s2s в терминах XMPP)
   - может создавать пользователей, перехватывать и генерировать события

Архитектура Matrix



1. **Homeserver** – полноправный узел сети, участвующий (или не участвующий) в федерации
2. **Клиенты** – занимаются только тем, что получают от «своего» сервера поток событий (и отправляют свои)
3. **Application Server** – своего рода произвольное расширение HS
4. **Identity Server** – опциональная сущность, (централизованно) отображающая сторонние идентификаторы на Matrix ID
   - Matrix ставит перед собой цель стать универсальным высокоуровневым транспортом

Архитектура Matrix



1. **Homeserver** – полноправный узел сети, участвующий (или не участвующий) в федерации
2. **Клиенты** – занимаются только тем, что получают от «своего» сервера поток событий (и отправляют свои)
3. **Application Server** – своего рода произвольное расширение HS
4. **Identity Server** – опциональная сущность, (централизованно) отображающая сторонние идентификаторы на Matrix ID
5. **Push Gateway** – шлюз для пересылки push-оповещений на устройства

Криптография в Matrix

- **Соединения «сервер-сервер»**
  - *TLS и подписывание на уровне протокола*
  - запросы подписываются ключом сервера (ed25519)
  - ответы аутентифицируются на уровне TLS
  - события подписываются ключом создавшего их сервера
  - ключи и TLS-сертификаты проверяются по принципу “trust on first use” + perspective servers
- **Соединения «клиент-сервер»**
  - *только TLS*
- **End-to-end шифрование (между участниками)**
  - **в разработке; будет в ближайших релизах**
  - *Axolotl – усовершенствованный вариант OTR*

Спасибо за внимание.

Спасибо за внимание.

References:

1. Спецификация, FAQ, демонстрационный сервер https://matrix.org
2. Perspectives https://perspectivessecurity.files.wordpress.com/2011/07/perspectives_usenix08.pdf
3. Axolotl https://github.com/trevp/axolotl/wiki