Introducing

matrix

Rethinking IP Communication
matrix:

A federated open-source VoIP and IM ecosystem
In practice....

• Pragmatic and lightweight open spec

• Open source reference client and server implementations

• Not-for-profit neutral custodian
What matrix changes:

- Anyone can build and host their own IP-comm service
- Users can choose who they trust with their data
- Users can use their favorite service to reach anyone as all services federate
## Why is now the right time?

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000–2005</td>
<td>SIP Open standard for VoIP; Industry builds proprietary closed ecosystems instead; For example Skype, Viber, WhatsApp… Now stuck in this state of fragmentation; Limited take-off</td>
</tr>
<tr>
<td>2005–2006</td>
<td>RCS initiative; GSMA tries to launch a standard for interoperability between SPs; But RCS is flawed and has very questionable success</td>
</tr>
<tr>
<td>As of the last ~1 year</td>
<td>the tech is finally available; WebRTC appeared; VoIP development is finally mainstream. The industry is ready.</td>
</tr>
</tbody>
</table>
So, what is it?

• Open spec

• Open source reference servers
  – matrix Home Server *(synapse*, written in Python/Twisted)
  – matrix Identity Server *(sydent*, written in Python/Twisted)

• Open source reference clients
  – Command Line Client (Python/Twisted)
  – Web Client *(AngularJS*)

• Third party matrix compliant servers, clients, gateways (in progress), app platforms (in progress), SDKs (in progress)...
What can it do?

• Federated rich instant messaging (1:1, public & private chat rooms, group chat, file sharing)
• Federated Presence, profiles, avatars
• End-to-end encryption (if desired; in progress)
• Federated VoIP calls and conferences
• Full multi-screen synchronisation of all state
Guiding Principles

• Be web-friendly, and trivial for web developers to use.

• Baseline transport is JSON + RESTful APIs (HTTP + SPDY)
  – more efficient transports are allowed as extensions

• All functionality is federated - no single points of control

• Group communication is the 1st class citizen
  – 1:1 is just a degenerate case of group
Architecture

- Clients
- Home Servers
- Identity Servers
- Policy Servers
Functional Responsibility

- **Clients**: Talks simple HTTP APIs to homeservers to push and pull messages and metadata. May be as thin or thick a client as desired.

- **Homeservers**: Stores all the data for a user - the history of the rooms in which they participate; their public profile data.

- **Identity Servers**: Trusted clique of servers (think DNS root servers): global repository of public keys for clients and servers. Also maps 3rd party IDs to matrix IDs.

- **Policy Server**: Optional; Acts as the focal point for all messages in a room which requires a single point of control (e.g. moderation) (in progress)
Federation Design #1

• No single point of control for chat rooms.

• Any homeserver can publish a reference to a chat room (although typically the address is the homeserver of the user who created the room).

• Room addresses look like:

  #matrix:matrix.org

  (pronounced hash-matrix-on-matrix-dot-org)

• The IP of the matrix.org homeserver is discovered through DNS (SRV _matrix record if available, otherwise looks for port 443 of the A record).
Federation Design #2

- When a user joins a room, his HS queries the HS specified in the room name to find a list of participating homeservers via a simple GET.

- Messages form a directed acyclic graph (DAG) of chronologicity, each crypto-signed by the origin HS.

- The user’s HS pulls in messages via GETs from participating HSs by attempting to walk the DAG.

- Each HS caches as much history as its users (or admin) desires.

- When sending a message, the HS PUTs to participating homeservers (currently full mesh, but fan-out semantics using cyclical hashing in development).
Identity Design

- We don't want to be yet another identity system (e.g. JIDs)

- So we aggregate existing 3rd party IDs (3PID) and map them to matrix IDs (MXIDs), whose use in public is strictly optional.

- And so login and user discovery is typically done entirely with 3rd party IDs.

- ID servers validate 3rd party IDs (e.g. email, MSISDN*, Facebook*, G+*) and map them to MXIDs. MXIDs look like:

  @matthew:matrix.org

* In progress
Security Design #1

- Server-server traffic is mandatorily TLS from the outset
- Can use official CA certs, but automagically self-sign and submit certs to **Matrix** ID servers as a free but secure alternative
- Server-client traffic mandates transport layer encryption other than for tinkering
- Clients that support PKI publish their public keys to ID servers, and may encrypt and sign their messages* for E2E security.
- "Well behaved" clients should participate in key escrow servers to allow private key submission for law enforcement.
- End-to-end encryption* for group chat is supported through a per-room encryption key which is shared 1:1 between participating members

* In progress
Security Design #2

• SPAM is contained by mandating invite handshake before communication
• Invite handshakes are throttled per user
• Homeservers may be blacklisted on identity servers
• ID servers authenticating 3PIDs are obligated to mitigate bulk registration of users via CAPTCHAs or domain-specific techniques (e.g. 2FA SMS for MSISDNs)
What about VoIP?

• Simple JSON-based offer-answer over the **matrix** messaging channel
• Basic support in the reference web client
• See the **specs** for details
Where are we at now?

- Matrix is still very much evolving. Main pending features:
  - E2E security
  - Policy servers
  - Mobile clients
  - Improved security

- Goal: mirror WHATWG's Living Standard approach except right now Matrix is more in the process of being born than actually being living!
Why not XMPP?

- We used to use XMPP (ejabberd, OpenFire, Spectrum, psyced, Psi, Pidgin, ASmack, Spark, XMPP.Framework)
- We built an alternative because:
  - Single server per MUC is single point of control
  - Synchronised history is a very 2nd class citizen
  - Stanzas aren't framed or reliably delivered
  - XMPP stacks are not easy to implement in a web environment
  - Jingle is complicated and exotic
  - XML is needlessly verbose and unwieldy
  - The baseline feature-set is too minimal
  - JIDs haven't taken off like Email or MSISDNs
  - Not designed for mobile use cases (e.g. push; low bw)
  - Well documented spam and identity/security issues
  - ejabberd
Why not psyc?

• psyc is an interesting early instance of better-than-XMPP federated chat
• psyc v1 has limitations:
  – Minimal spec
  – Few implementations
  – Security issues
  – Not web-friendly
• psyc v2 has become part of GNUnet, providing end-to-end secure group chat on top of the censorship-resistant GNUnet overlay network.
  – Dependent on the complexities and usability challenges of the GNUnet ecosystem
  – Not web-friendly
  – But does provide anti-censorship guaranties that Matrix doesn’t
Get involved!

- Support a promising new ecosystem
- Be a Thought Leader and play an active role
Support a promising new ecosystem

- Validate and sanity-check the spec
- Help guide how matrix grows
Be a Thought Leader and play an active role

- Expose **matrix** APIs and act as a gateway to any existing community
  ➔ add a whole **new ecosystem to your community**

- Build your own customized **matrix** clients
  ➔ participate in **matrix**'s diversity and growth while **keeping your identity and maintaining your level of service**

- Host your own **matrix** homeservers
  ➔ be a **trusted provider for your customers** like you are today

- Provide self-hosted **matrix** compliant platforms
  ➔ allow your customers to host their own platform to **give them control over their data**
THANK YOU!

http://matrix.org

Contact us at:
matthew@matrix.org
amandine@matrix.org