Introducing Matrix

matthew@matrix.org
http://www.matrix.org
Matrix is an...

Open
Decentralised
Persistent
Eventually Consistent
Cryptographically Secure
JSON-over-HTTP
Messaging Fabric
Matrix is for:

Group Chat (and 1:1)
WebRTC Signalling
Bridging Comms Silos
Internet of Things Data

...and anything else which needs to
pubsub persistent data to the world.
Why are you re-inventing XMPP!?!?
WE ARE NOT.
Matrix is essentially a distributed EC persistent messaging database with an HTTP API.

History and Group comms are 1\textsuperscript{st} class citizens.
What does it look like?
Demo time!

http://matrix.org/beta
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): maps 3rd party IDs to matrix IDs.

- **Application Services**: Optional; delivers application layer logic on top of Matrix (Gateways, Conferencing, Archiving, Search etc). Can actively intercept messages if required.
What do you get?

• **Open Standard HTTP APIs (WIP)**
  • Client-Server API (v2)
  • Server-Server API
  • Application Service API
  • Identity Server API

• **Open Source Reference Implementations**
  • Synapse Homeserver (Python/Twisted)
  • SDKs for AngularJS, iOS, Android, Perl, Python
  • Clients for AngularJS, iOS, Android

• **Third Party Implementations**
  • Pallium Homeserver (Go)
How does it work?

http://matrix.org/matrix-graph.html
The client-server API

To send a message:

curl -XPOST -d '{"msgtype":"m.text", "body":"hello"}'
"https://alice.com:8448/_matrix/client/api/v1/rooms/
ROOM_ID/send/m.room.message?access_token=ACCESS_TOKEN"

{
    "event_id": "YUwRidLecu"
}
The client-server API

To set up a WebRTC call:

curl -XPOST -d '{
  "version": 0,
  "call_id": "12345",
  "offer": {
    "type": "offer",
    "sdp": "v=0\r\n      o=- 658458 2 IN IP4 127.0.0.1...
  }
}'} https://alice.com:8448/_matrix/client/api/v1/rooms/ROOM_ID/send/m.call.invite?access_token=ACCESS_TOKEN

{ "event_id": "ZruiCZBu" }
The client-server API

To persist some MIDI:

curl -XPOST -d '{
    "note": "71",
    "velocity": 68,
    "state": "on",
    "channel": 1,
    "midi_ts": 374023441

{ "event_id": "ORzcZn2" }
The client-server API

...or to persist some tap gestures for animating an Avatar...

curl -XPOST -d '{
  "thumbnail": "http://matrix.org:8080/_matrix/content/QGtlZ2FuOm1hdHJpeC5vcmcvNupjfhmFhjxDPquSZGaGlYj.aW1hZ2UvcG5n.png",
  "actions": [
    {"x": "0.5521607", "y": "6.224353", "t": "0.9479785"},
    {"x": "0.5511537", "y": "6.220354", "t": "0.9701037"},
    {"x": "0.5510949", "y": "6.214756", "t": "0.9804187"},
    {"x": "0.5499267", "y": "6.213634", "t": "0.9972034"},
    {"x": "0.5492241", "y": "6.210211", "t": "1.013744"},
    {"x": "0.5486694", "y": "6.206304", "t": "1.030284"},
    {"x": "0.5482137", "y": "6.201648", "t": "1.046764"},
    ...
    {"x": "0.9997056", "y": "4.022976", "t": "8.970592"},
    {"x": "0.9995697", "y": "4.043199", "t": "8.987072"}
  ]

{ "event_id": "ORzcZn2" }
The server-server API

curl -XPOST -H 'Authorization: X-Matrix origin=matrix.org, key="898be4...", sig="j7JXfIcPFDWl1pdJz..."' -d '{
"ts": 1413414391521,
"origin": "matrix.org",
"destination": "alice.com",
"prev_ids": ["e1da392e6189be4d2009b9fece5325"],
"pdus": [{
  "age": 314,
  "content": {
    "body": "hello world",
    "msgtype": "m.text"
  },
  "context": "!fkILCTRBTNhftNYgkP:matrix.org",
  "depth": 26,
  "hashes": {
    "sha256": "MqVORjmjauxBDBzSyN2+Yu+KJxw0oxrrJyuPWBNpELs"
  },
  "is_state": false,
  "origin": "matrix.org",
  "pdu_id": "rKQFuZQawa",
  "pdu_type": "m.room.message",
  "prev_pdus": [
    ["PaBNREEuZj", "matrix.org"]
  ],
  "signatures": {
    "matrix.org": {
      "ed25519:auto": "jZXTwAH/7EZbjHFhIFg8Xj6HGoSI+j7JXfIcPFDWl1pdJz+JJPMDTZjTRh475oJ7lg7UM+CnhNAayHwZsUY3Ag"
    }
  }
},
"origin_server_ts": 1413414391521,
"user_id": "@matthew:matrix.org"
}

https://alice.com:8448/_matrix/federation/v1/send/916d630ea616342b42e98a3be0b74113
Current Progress

• Funded May 2014
• First public release in Sept 2014
• Crypto and iOS/Android landed Oct 2014
• Exited alpha Nov 2014
• Dec: 40 federated homeservers; 500 end users.
• Next up:
  – Spec overhaul
  – Finalise Application Server APIs
  – v2 Client-Server API
  – UX polish for the reference clients
  – End-to-End Encryption
We need help!!
• We need people to try running their own servers and join the federation.

• We need feedback on the APIs.

• We need more people to actually use it!
THANK YOU!

matrix: @matthew:matrix.org
mail: matthew@matrix.org
twitter: @matrixdotorg

http://matrix.org
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 8448 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) by Identity Servers, 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
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, 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
Security Design #2

- SPAM is contained by mandating invite handshake before communication
- Invite handshakes are throttled per user
- Homeservers and users may be blacklisted on identity servers
- ID servers authenticating 3PIIDs are obligated to mitigate bulk registration of users via CAPTCHAs or domain-specific techniques (e.g. 2FA SMS for MSISDNs)
Application Services: Spec & API

- Still in development; some early prototypes
- "Passive AS-API" Builds on the client-server API
  - Service registers a URL for inbound events to be PUT to
  - Allows a service to register for traffic on behalf of a namespace of virtual users and virtual rooms
  - Adds "superuser" permissions to subscribe to arbitrary filters of events on the homeserver, and inject arbitrary events
  - Modeled loosely after IRC Services
- Also: Active AS API for intercepting inbound events on a HS, and Storage API for exposing existing conversation DBs to Matrix via a HS.
AS Example: Matrix/SMS Gateway

- matrix.org runs a homeserver.
- Matrix/SMS gw AS is registered to the homeserver, masquerading for the 'sms.matrix.org' domain.
- @447968722968:sms.matrix.org routes to the homeserver from anywhere in Matrix, which passes events for *:sms.matrix.org through to the AS
- Matrix/SMS Gateway then relays via SMS aggregators to send SMS to +447968722968
- The reverse path is symmetrical, with the Matrix/SMS AS injecting events into the HS on behalf of @447968722968:sms.matrix.org
• Similarly, AS can implement a SIP gateway, posing as a range of virtual matrix users.

• Events such as 'm.call.invite' and 'm.call.candidates' are PUT to the AS by the HS

• AS converts directly into SIP signalling (reINVITEing to advertise new ICE candidates)

• Media flows out-of-band to Matrix as typical WebRTC SRTP.

• We've already written a basic Matrix/Verto gateway (using client-service API – see matrix.org/blog)
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