[matrix]

The missing signalling layer for WebRTC?

matthew@matrix.org

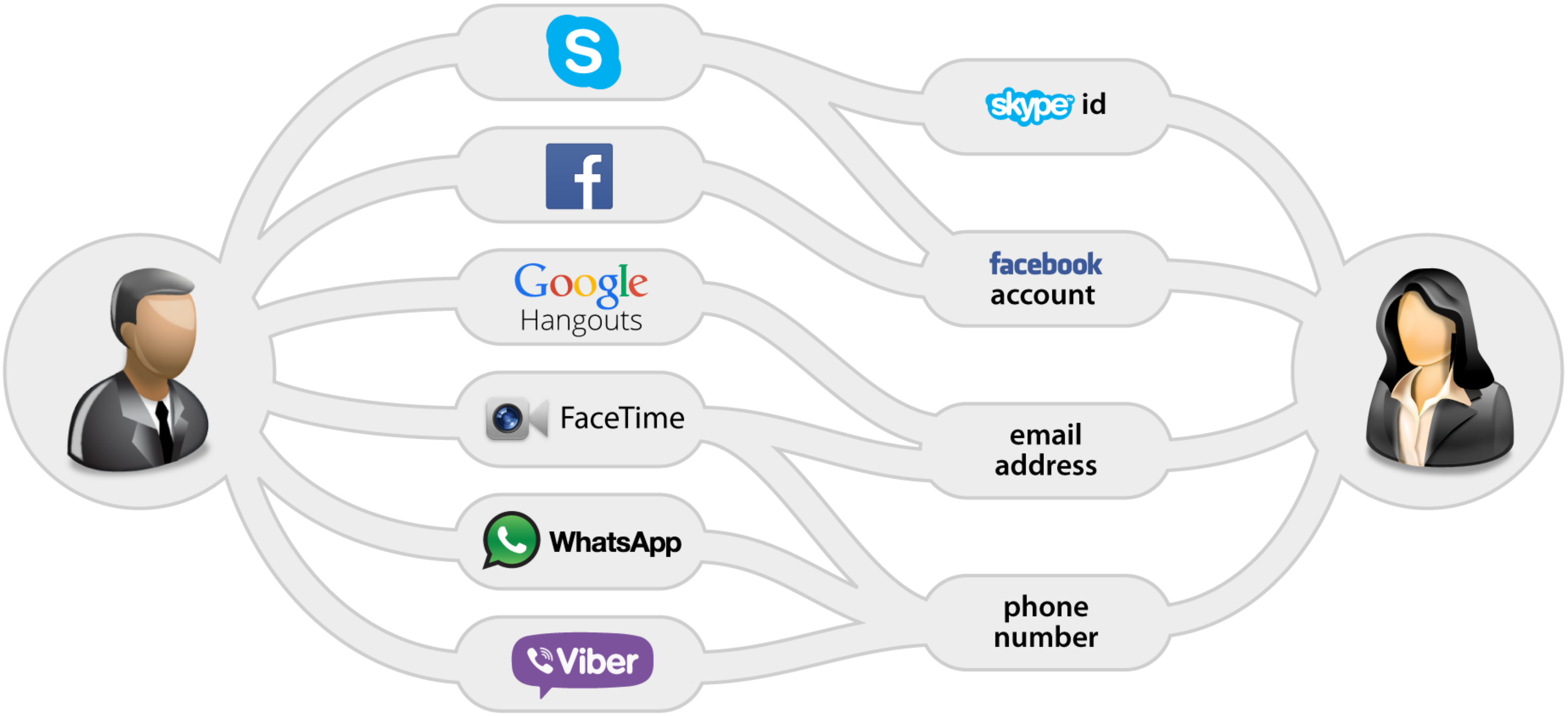
WebRTC deliberately specifies no specific signaling protocol.

➔ It makes interoperability and federation hard.

➔ It creates silos.

As a user:

matrix



Google Hangouts
FaceTime
WhatsApp
Viber
skype id
facebook account
email address
phone number

I want to use my preferred apps and services to communicate

[matrix]

Not be forced into specific services chosen by my contacts.

If email gives me that flexibility, why not VoIP and IM?

Current signaling protocol options include:

- SIP
- XMPP
- WebRTC Data Channel (e.g. Open Peer)
- Assorted HTTP APIs

SIP:

- Heavyweight
- Complicated specification
- Complicated stack
- Buys little over HTTP

XMPP/Jingle:

- Streamed XML is debatable
- Relatively complicated spec
- Jingle has relatively little uptake
- Custom stack

HTTP APIs:

- Simple
- But fragmented
- And often proprietary
- Or closed (Firebase, Pusher, PubNub...)

Introducing Matrix

Introducing Matrix

- New Open Source project (launched Sept 2014)

[matrix]

Introducing Matrix

- New Open Source project (launched Sept 2014)
- Setting up as non-profit org (matrix.org)

[matrix]

Introducing Matrix

- New Open Source project (launched Sept 2014)
- Setting up as non-profit org (matrix.org)
- Publishing pragmatic simple HTTP API standard for federated VoIP (WebRTC), IM and generic messaging.

Introducing Matrix

- New Open Source project (launched Sept 2014)
- Setting up as non-profit org (matrix.org)
- Publishing pragmatic simple HTTP API standard for federated VoIP (WebRTC), IM and generic messaging.
- Defines client-server and server-server APIs (and, shortly, server<->application-server APIs).

Introducing Matrix

- New Open Source project (launched Sept 2014)
- Setting up as non-profit org (matrix.org)
- Publishing pragmatic simple HTTP API standard for federated VoIP (WebRTC), IM and generic messaging.
- Defines client-server and server-server APIs (and, shortly, server<->application-server APIs).
- Provides Apache-Licensed reference implementations of the server and clients (web, iOS, Android, Python, Perl...)

Who is Matrix?

Matthew Hodgson

- Technical Leader of matrix.org
- Set up and runs the Unified Communications line of business within Amdocs (formerly MX Telecom)
- 11 years of experience building IP telephony solutions and leading units

Amandine Le Pape

- Business Leader of matrix.org
- Set up and co-runs the Unified Communications line of business within Amdocs as a Product Manager
- 10 years of experience in mobile services and telecommunications

The Dev Team

- A dozen of experienced developers specialized in VoIP and IM mobile app development
- Most of them originally from the Amdocs Unified Communications team (flagship deployment: blah.com)

Matrix comes from realising that VoIP and IM fragmentation is holding back the whole industry - we didn't want to be part of the problem, but try to solve it.

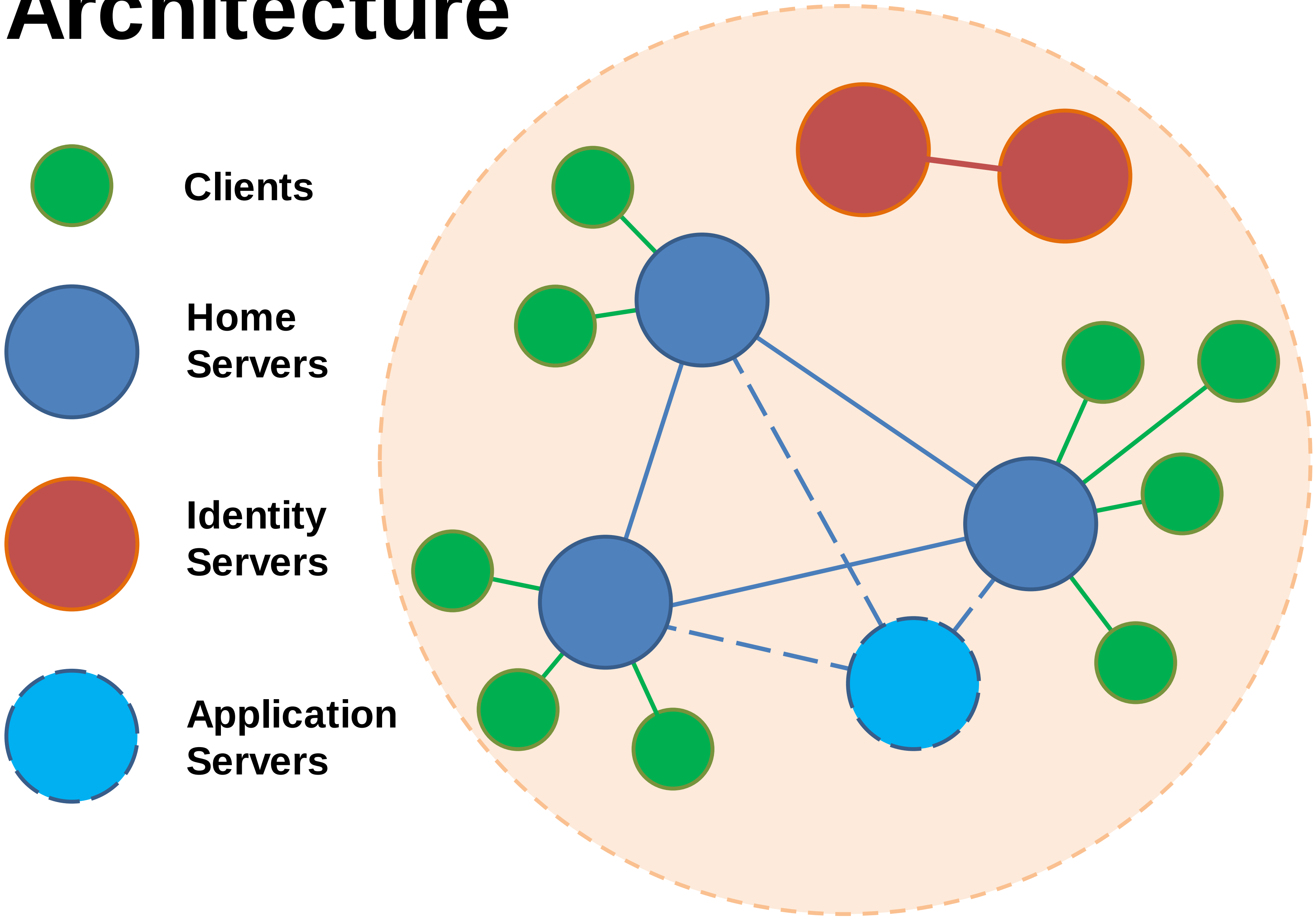
Key Characteristics

- Entirely open:
  - open standard; open source; open project.
- Message History as first-class citizen
- Group communication as first-class citizen
  - Fully distributed room state (cryptographically signed) - no SPOFs or SPOCs.
- Strong cryptographic identity to prevent spoofing
- Identity agnostic
- End-to-end encryption (RSN)

Demo time!

matrix

Architecture



Federation Demo

[matrix]

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\no=- 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\
}'
"https://alice.com:8448/_matrix/client/api/v1/rooms/ROOM_
ID/send/org.matrix.midi?access_token=ACCESS_TOKEN"

{ "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.aW1hZ2U
vcG5n.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"}
    ]
}'
"https://alice.com:8448/_matrix/client/api/v1/rooms/ROOM_ID/send/org.matrix.demos.unity.stickme
n?access_token=ACCESS_TOKEN"

{ "event_id": “ORzcZn2” }
```

[matrix]

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": ["e1da392e61898be4d2009b9fecce5325"],
    "pdus": [{
        "age": 314,
        "content": {
            "body": "hello world",
            "msgtype": "m.text"
        },
        "context": "!fkILCTRBTHhftNYgkP:matrix.org",
        "depth": 26,
        "hashes": {
            "sha256": "MqVORjmjauxBDBzSyN2+Yu+KJxw0oxrrJyuPW8NpELs"
        },
        "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+JJPMHTDIZRha75oJ7lg7UM+CnhNAayHWZsUY3Ag"
            }
        },
        "origin_server_ts": 1413414391521,
        "user_id": "@matthew:matrix.org"
    }]
}’ https://alice.com:8448/_matrix/federation/v1/send/916d630ea616342b42e98a3be0b74113
```

What about IoT?

CoAP:

- REST over UDP (sort of)
- Everything's a server!
  (and a client)
- Maps onto HTTP APIs.

MQTT:

- PubSub over TCP (sort of)
- Everything can pub & sub!
  (via a broker).
- Maps onto message passing.

Both are very different.

But neither provide:

- **Global federated messaging**
- **Message History**
- **Message Signing**
- **E2E Encryption**

Matrix to the rescue?

Exposing Matrix via CoAP is trivial:

```
echo '{"msgtype":"m.text", "body":"hello"}' |
perl -MCBOR::XS -MJSON -pe '$_=encode_cbor decode_json' |
coap-client -m post \
coaps://alice.com/_m/c/a/v1/r/ROOM_ID/s/m.room.message?a=
ACCESS_TOKEN
```

is the same as…

```
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"
```

Any CoAP device can persist data into Matrix, and act on data pushed from Matrix.

A Matrix-aware MQTT Broker could similarly store history to Matrix, and expose Matrix history and pubsub to MQTT clients.

Current Progress

- Began May 2014
- First public release in Sept 2014
- Crypto and iOS/Android landed Oct 2014
- Next up:
  - Complete the spec
  - Complete federation implementation
  - Declare reference server production ready
  - UX polish for the reference clients
  - Define Application Server APIs
  - End-to-End Encryption
  - IoT implementations!

[matrix]

Get involved!

- Run a server
  ➔ host your own data or be a trusted provider for your customers

- Build something (anything!) on top

- Build interoperability gateways
  ➔ add a whole new ecosystem to your community

Check out http://matrix.org!

Follow us at @matrixdotorg!

[matrix]

http://matrix.org

THANK YOU!

matrix: @matthew:matrix.org

mail: matthew@matrix.org

twitter: @matrixdotorg

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