Interoperable HTTP Signalling with Matrix

matthew@matrix.org
Quick Intro/Recap:

WebRTC has no standard signalling.

- Users lose control of UX
- Fragmentation
- Vendor lock-in
- The Dark Side

So, how about some standard WebRTC-friendly signalling?
Client–Server Signalling

Do you do HTTP/1? HTTP/2? WebSockets? non–HTTP?
Is it an open standard, proprietary open API, or closed?
What are your identifiers?
How stateful is your connection?
What payload encoding? JSON? Protobuf? CBOR?
What session descriptors? SDP? ORTC?
How extensible do you make it?
Do you do end–to–end crypto? How do keys work?
Do you support group conversations?
Do you handle conversation history?
Do you specify how Push fits in?
Do you support federation?
How mature is the technology?
SIP-over-WebSockets

Do you do HTTP/1? HTTP/2? WebSockets? non-HTTP?
Is it an open standard, proprietary open API, or closed?
What are your identifiers? SIP URIs
How stateful is your connection? Very
What payload encoding? SIP Headers + MIME body
What session descriptors? SDP? ORTC?
How extensible do you make it? Fairly
Do you do end-to-end crypto? No
Do you support group conversations? Via focuses
Do you handle conversation history? Not really
Do you specify how Push fits in? No
Do you support federation? Yes
How mature is the technology? Very
XMPP–FTW

Do you do HTTP/1? HTTP/2? WebSockets? non-HTTP?


Is it an open standard, proprietary open API, or closed?

What are your identifiers? XMPP JIDs

How stateful is your connection? Very

What payload encoding? JSON

What session descriptors? Jingle

How extensible do you make it? Very

Do you do end-to-end crypto? Competing XEPs

Do you support group conversations? Via MUCs

Do you handle conversation history? Competing XEPs

Do you specify how Push fits in? Yes, as of Mar 2015

Do you support federation? Yes

How mature is the technology? Very
Matrix

Do you do **HTTP/1? HTTP/2? WebSockets? non-HTTP?**
Is it an open standard, proprietary open API, or closed?
What are your identifiers? **3rd party IDs & Matrix IDs**
How stateful is your connection? **No state.**
What payload encoding? **JSON**
What session descriptors? **SDP (ORTC in future?)**
How extensible do you make it? **Extensible data**
Do you do end-to-end crypto? **Per-room**
Do you support group conversations? **Yes**
Do you handle conversation history? **Yes**
Do you specify how Push fits in? **Yes**
Do you support federation? **Yes**
How mature is the technology? **Beta**
Server–Server Signalling

Is it similar complexity to client–server or heavier?
Is compulsorily encrypted?
Does it have cryptographically strong IDs?
Does it track reputation or trust?
Does group traffic fan out (multicast)?
Is history decentralised?
Is history tamper-resistant?
SIP-over-WebSockets (S2S)

Is it similar complexity to client-server or heavier? No
Is compulsorily encrypted? No
Does it have cryptographically strong IDs? No
Does it track reputation or trust? No
Does group traffic fan out (multicast)? No
Is history decentralised? No
Is history tamper-resistant? No
XMPP-FTW (S2S)

Is it similar complexity to client-server or heavier? Yes
Is compulsorily encrypted? Since May 2014
Does it have cryptographically strong IDs? With XEPs
Does it track reputation or trust? No
Does group traffic fan out (multicast)? No
Is history decentralised? Only on FMUC XEP
Is history tamper-resistant? Only on FMUC XEP
Matrix Federation

Is it similar complexity to client-server or heavier? Yes
Is compulsorily encrypted? Yes
Does it have cryptographically strong IDs? Yes
Does it track reputation or trust? Perhaps
Does group traffic fan out (multicast)? Real soon now
Is history decentralised? Yes
Is history tamper-resistant? Yes
Open
Decentralised
Persistent
Eventually Consistent
Cryptographically Secure
Messaging Database
with JSON–over–HTTP API.
Federation Demo

https://matrix.org
Architecture (Bridging)
Dangerous Demo!!!
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
}'

{ "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/QGtlZ2FuOm1hdHJpeC5vcmcvNupjfhmFhjxDPquSZGaG1Yj.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"}
  ]
}'
n?access_token=ACCESS_TOKEN"

{ "event_id": "ORzcZn2" }
curl -XPOST -H 'Authorization: X-Matrix origin=matrix.org, key="898be4...", sig="j7JXfIcPFDWl1pd3z..."' -d '{
    "ts": 1413414391521,
    "origin": "matrix.org",
    "destination": "alice.com",
    "prev_ids": ["e1da392e61898be4d2009b9fcce5325"],
    "pdus": [{
        "age": 314,
        "content": {
            "body": "hello world",
            "msgtype": "m.text"
        },
        "context": "!fkILCTRBThtNYgkP:matrix.org",
        "depth": 26,
        "hashes": {
            "sha256": "MqVORjmjauxBBDzSyNz9Yt+15JGw0oxrr3yuPW8NpELs"
        },
        "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/7EZbHFiFg8Xj6HGoSI+j7JXfIcPFDWl1pdJz+JiPMHTDIZRha75oJ71g7UM+CnhNAayHWzUY3Ag"
            }
        },
        "origin_server_ts": 1413414391521,
        "user_id": "@matthew:matrix.org"
    }
}'} https://alice.com:8448/_matrix/federation/v1/send/916d630ea616342b42e98a3be0b74113
Application Services (AS)

- Extensible custom application logic
- They have privileged access to the server (granted by the admin).
- They can subscribe to wide ranges of server traffic (e.g. events which match a range of rooms, or a range of users)
- They can masquerade as 'virtual users'.
- They can lazy-create 'virtual rooms'
- They can receive traffic by push.
Uses for AS API

• Gateways to other comms platforms
• Data manipulation
  – Filtering
  – Translation
  – Indexing
  – Mining
  – Visualisation
  – Orchestration
• Application Logic (e.g. bots, IVR services)
• …
The application service API

Register the AS with the homeserver (this actually is turning into HS config).

curl -XPOST -d \
'\n    {\n        "as_token": "TOKEN",\n        "url": "http://localhost:5000",\n        "namespaces": {\n            "aliases":[{"regex": "#logged_.*", "exclusive": false}]\n        }\n    }' \n"https://alice.com:8448/_matrix/appservice/v1/register"
The application service API

Receive events from the homeserver.

```python
import json, requests  # we will use this later
from flask import Flask, jsonify, request
app = Flask(__name__)

@app.route("/transactions/<transaction>", methods=["PUT"])  
def on_receive_events(transaction):
    events = request.get_json()["events"]
    for event in events:
        print "User: %s Room: %s" % (event["user_id"], event["room_id"])
        print "Event Type: %s" % event["type"]
        print "Content: %s" % event["content"]
    return jsonify({})

if __name__ == "__main__":
    app.run()
```
Current Progress

• Funded May 2014
• Launched alpha Sept 2014
• Entered beta Dec 2014
• May 2014: v1.0 release?!
• Remaining:
  – Performance improvements in reference impls
  – Build more gateways
  – Finalise spec
  – End-to-End Encryption
  – v2 Client-Server API

Won Audience Choice & Best Social Integration awards at WebRTC Expo 2014 and Best Innovation at WebRTC Paris 2014
We need help!!
• We need partners to participate in Matrix.

• We need people to run their own servers and join Matrix.

• 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