[matrix]

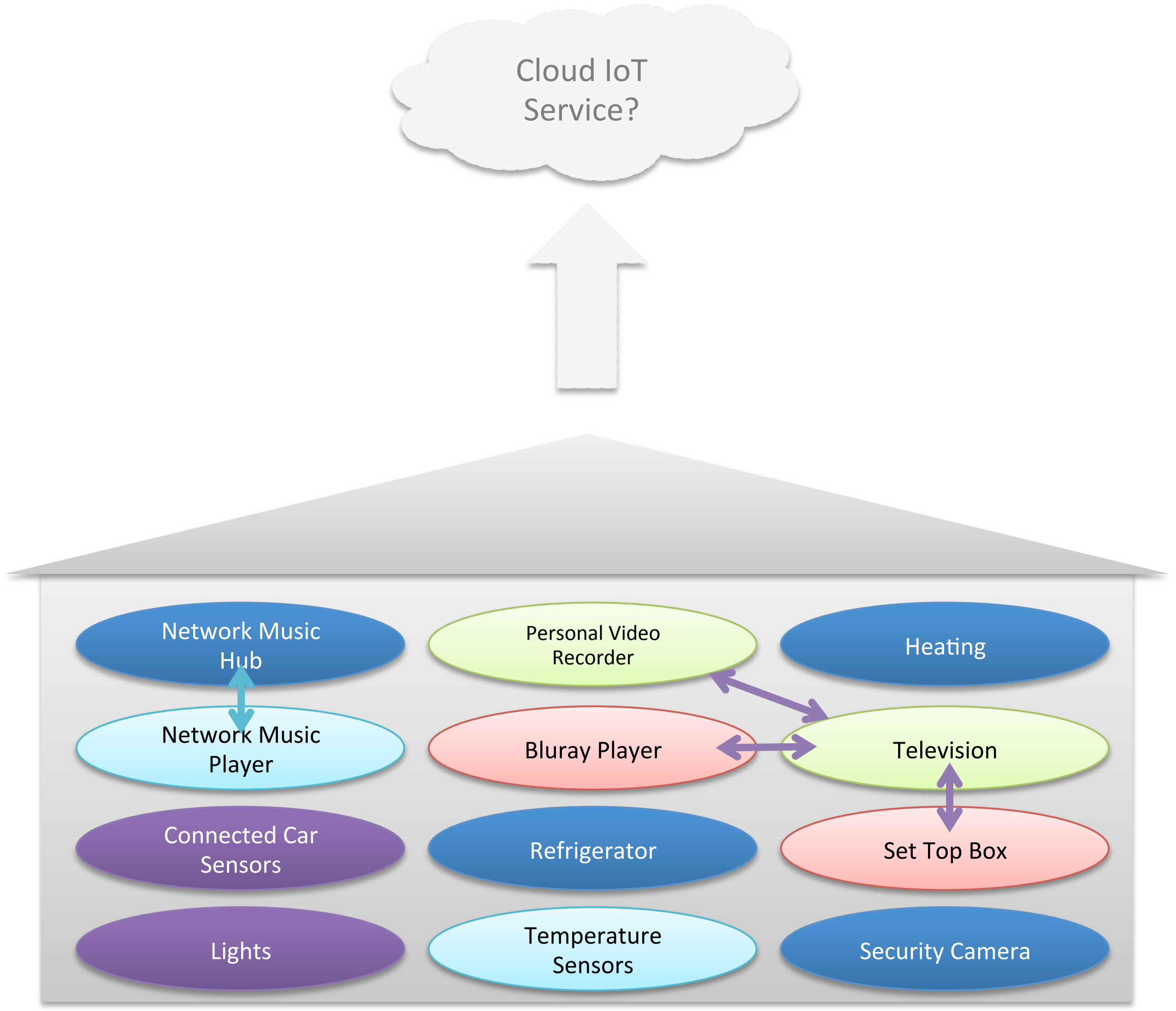
IoT through Matrix

matthew@matrix.org

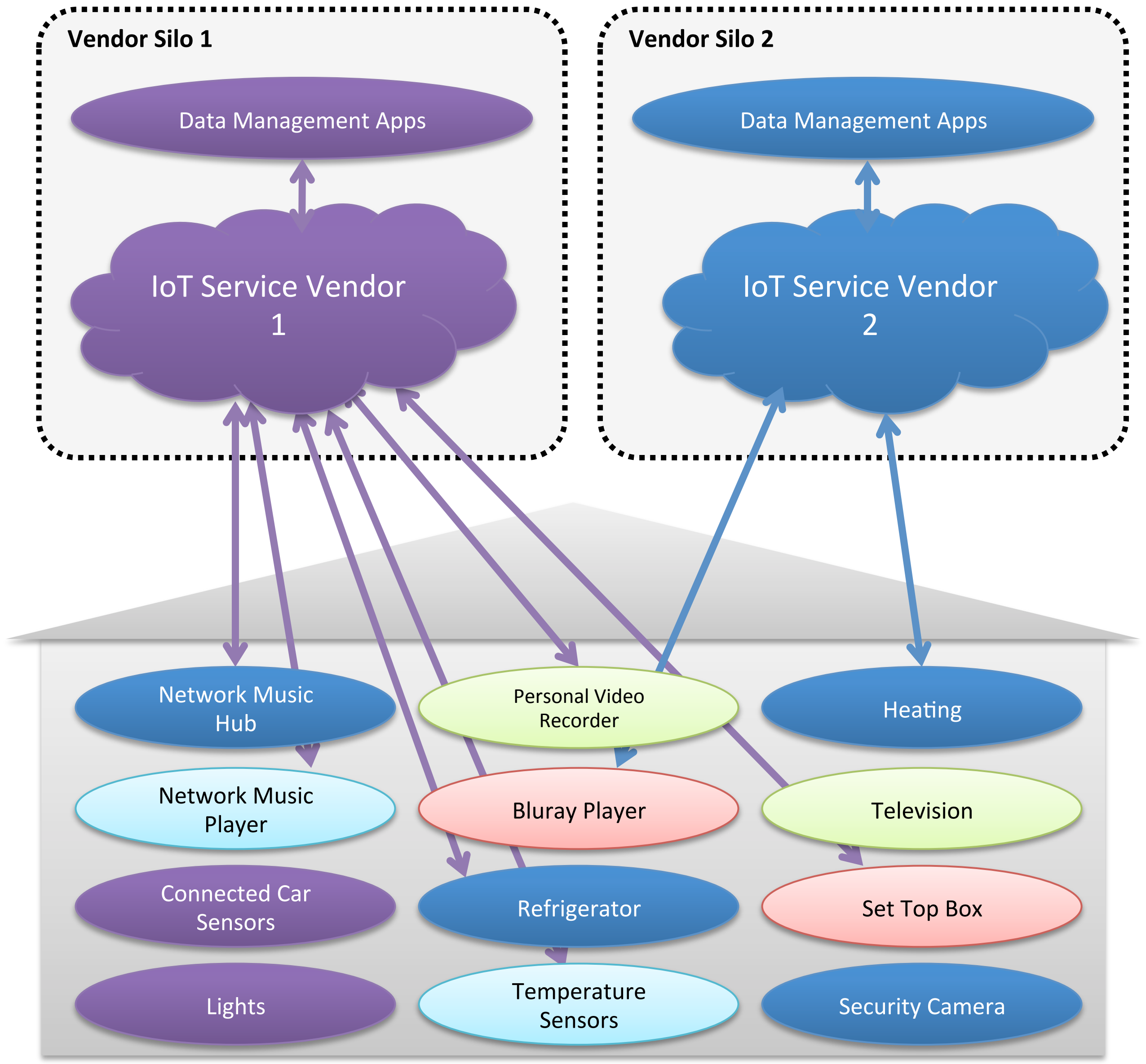
What's the problem with IoT?

The past:

[matrix]



The present:


Vendor Silo 1
Data Management Apps
IoT Service Vendor 1
Vendor Silo 2
Data Management Apps
IoT Service Vendor 2
Network Music Hub
Personal Video Recorder
Heating
Network Music Player
Bluray Player
Television
Connected Car Sensors
Refrigerator
Set Top Box
Lights
Temperature Sensors
Security Camera

The Problem:

- Data stored in vendor silos
(even for open protocols like XMPP, MQTT, COAP…)
- Device/data management apps locked to vendor silos…
- Devices locked to specific vendor services…

=> Vendor Lock In and Fragmentation.

Introducing Matrix

[matrix]

Introducing Matrix

- New Open Source project (launched Sept 2014)
- Setting up as not-for-profit org (matrix.org)
- Publishing pragmatic simple HTTP API standard for persistent decentralised messaging.
- Defines client-server, server-server and application-service APIs
- Provides Apache-Licensed reference implementations of the server (Python/Twisted) and clients (web, iOS, Android, Python, Perl...)

Open
Decentralised
Persistent
Eventually Consistent
Cryptographically Secure
Messaging Database
with JSON-over-HTTP API.

Key Characteristics

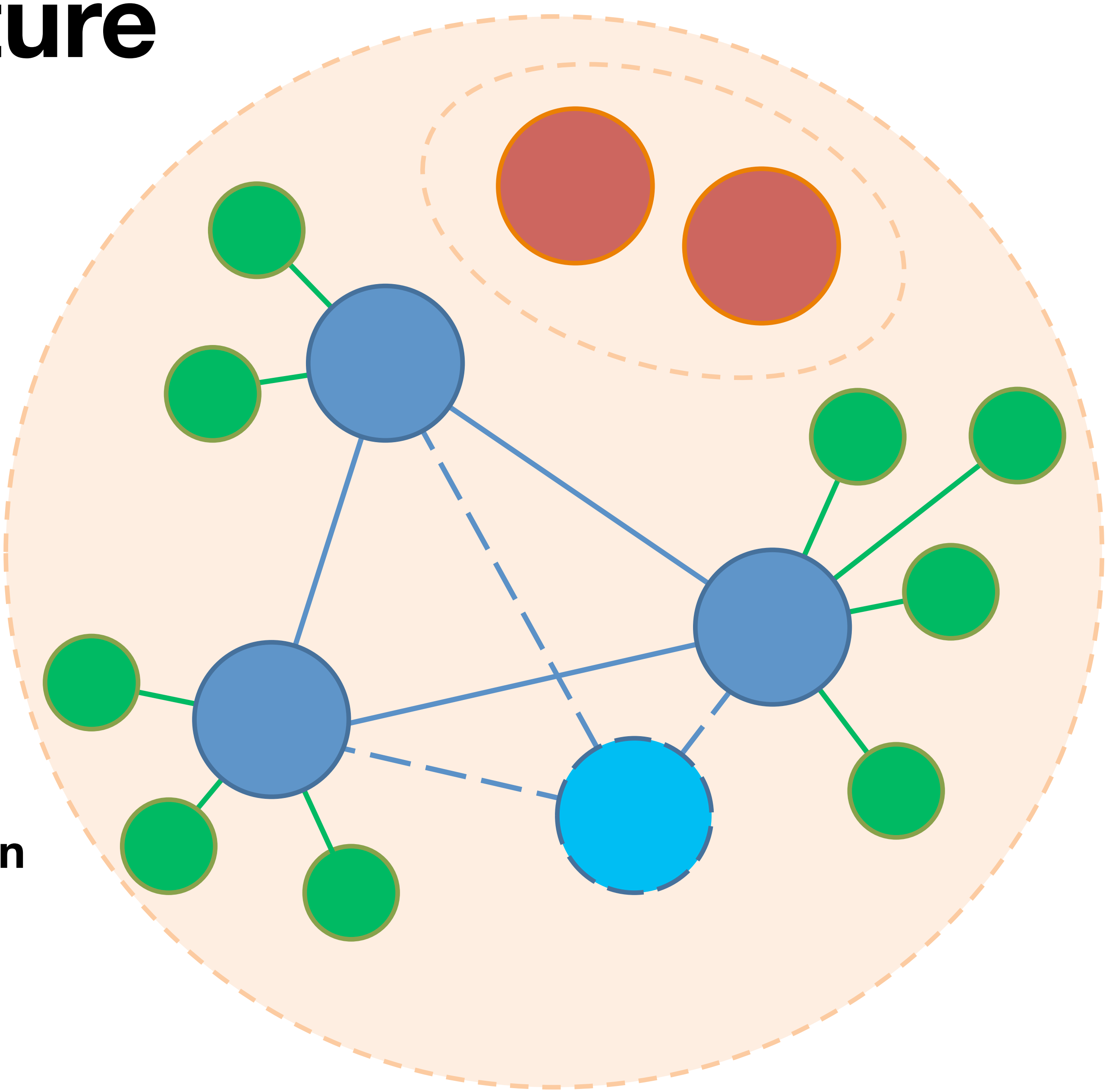
- Entirely open:
  - open standard; open source;
    open project; open federation.
- Message History as first-class citizen
- Group communication as first-class citizen
  - Fully distributed room state (cryptographically signed)
    - no SPOFs or SPOCs.
- Strong cryptographic identity to prevent spoofing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

Demo time!

http://matrix.org/beta

Architecture

●	Clients
●	Home Servers
●	Identity Servers
●	Application Services



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.

Federation Demo

http://matrix.org/matrix-graph.html

[matrix]

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

[matrix]

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 3PIDs are obligated to mitigate bulk registration of users via CAPTCHAs or domain-specific techniques (e.g. 2FA SMS for MSISDNs)

Application Services (AS)

- …are Bots on steroids (with a hint of IRC services)
- 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 worlds
- Data manipulation
  - Filtering
  - Translation
  - Indexing
  - Mining
- Application Logic (e.g. bots, IVR services)
- …

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"
}
```

[matrix]

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” }
```

[matrix]

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” }
```

[matrix]

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"}
    ]
}' "https://alice.com:8448/_matrix/client/api/v1/rooms/ROOM_ID/send/
org.matrix.demos.unity.stickmen?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
```

Federating IoT Data

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.

Exposing Matrix via CoAP is trivial:

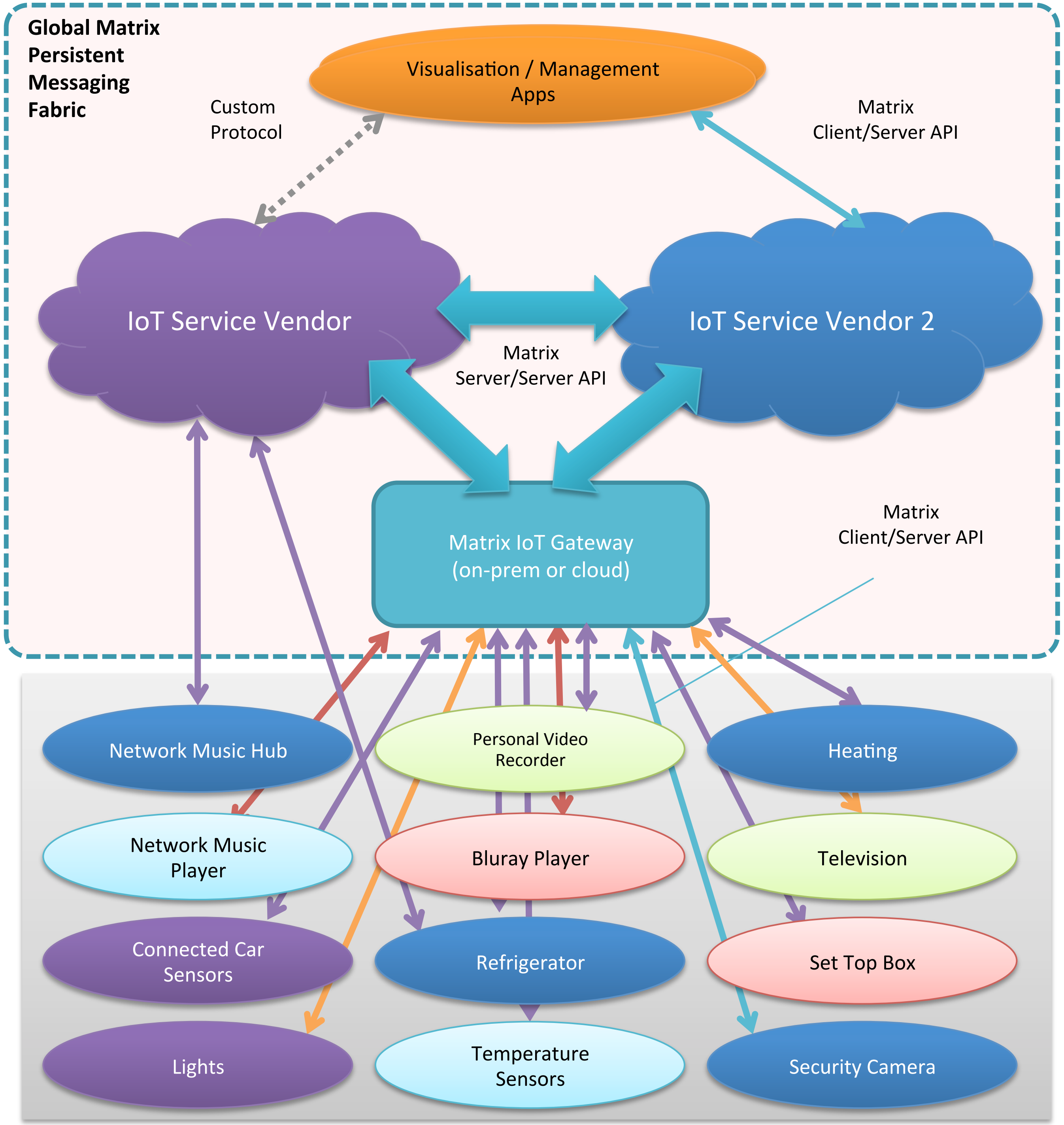
```
echo '{"msgtype":"m.text", "body":"hello"}' |
perl -MCBOR::XS -MJSON -pe '$_=encode_cbor decode_json' |
coap-client -m post \
coaps://alice.com/_m/c/a/v1/r/ROOM_ID/s/m.room.message?
a=ACCESS_TOKEN
```

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.


Global Matrix Persistent Messaging Fabric
Custom Protocol
Visualisation / Management Apps
Matrix Client/Server API
IoT Service Vendor
IoT Service Vendor 2
Matrix Server/Server API
Matrix IoT Gateway (on-prem or cloud)
Matrix Client/Server API
Network Music Hub
Personal Video Recorder
Heating
Network Music Player
Bluray Player
Television
Connected Car Sensors
Refrigerator
Set Top Box
Lights
Temperature Sensors
Security Camera

Current Progress

- Funded May 2014
- First public release in Sept 2014
- Crypto and iOS/Android landed Oct 2014
- Exited alpha Nov 2014
- Jan 2014: 60 federated homeservers; 700 end users.
- Next up:
  - Release v2 Client-Server APIs
  - Release Application Server APIs
  - Build gateways!
  - End-to-End Encryption
  - Sort out Federated ID & Key Distribution

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!
  - Come talk on #matrix:matrix.org (http://matrix.org/beta)
  - Follow us @matrixdotorg

[matrix]

http://matrix.org

THANK YOU!

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