Interoperability & Matrix

FOSDEM 2024

Travis Ralston
Director of Standards Development - matrix.org
@travis:t2l.io | travisr@matrix.org

But wait, there's more

matrix

- Matthew's mainstage talk this morning covers our DMA adventure in detail.
- DMA requires large messaging providers ("gatekeepers") to open up their systems for interoperability.
- Encryption **must** be maintained at the same level between providers.
- Messengers have three options:
  1. Become multi-headed, like Beeper Mini.
  2. Create a client-side bridge app to proxy between services.
  3. Speak a common protocol.
- We've spent the last year working on **Option 3**.
- Oh, and the DMA starts coming into force March 7th, **2024**.

Projects

- “More Instant Messaging Interoperability” (MIMI) working group at the IETF is aiming to specify a standard for modern interoperable communication.
  - Matrix is a frequent and direct contributor to these efforts.
  - I-D.ralston-mimi-protocol receiving updates to better cover recent feedback.
- Linearized Matrix originally created as a simplified version of Matrix for use within MIMI as an existing protocol.
  - Fully compatible with the existing DAG-based Matrix network.
  - Uses a linked list instead of a DAG internally.
  - Ultimately rejected because it stored history, and providers don’t think that’s required.
- Matrix itself, as a fully-featured and existing open standard for interoperable communications, including messaging.

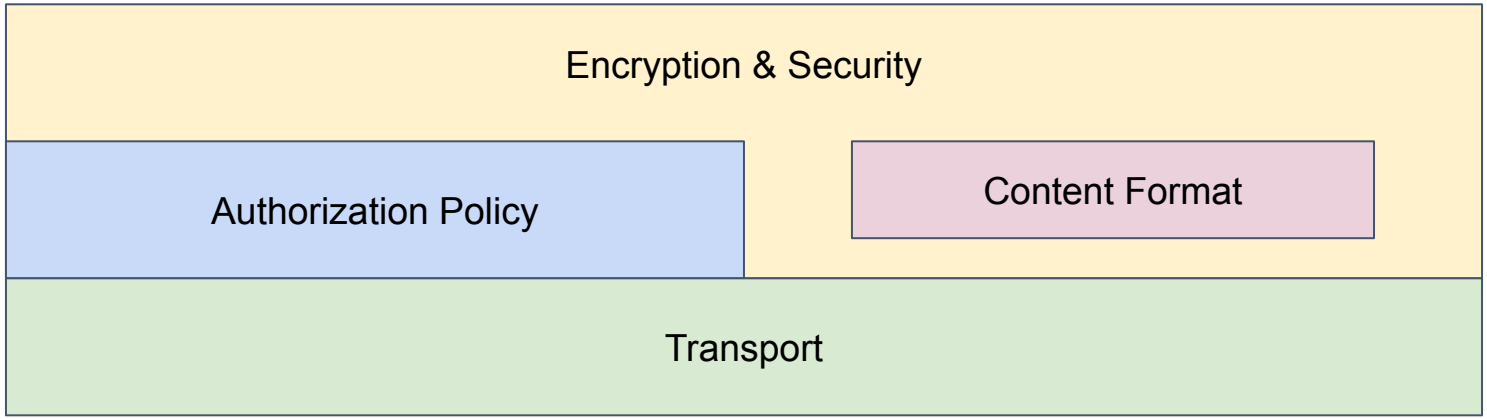
Parts of interop

Technical problem domains

matrix

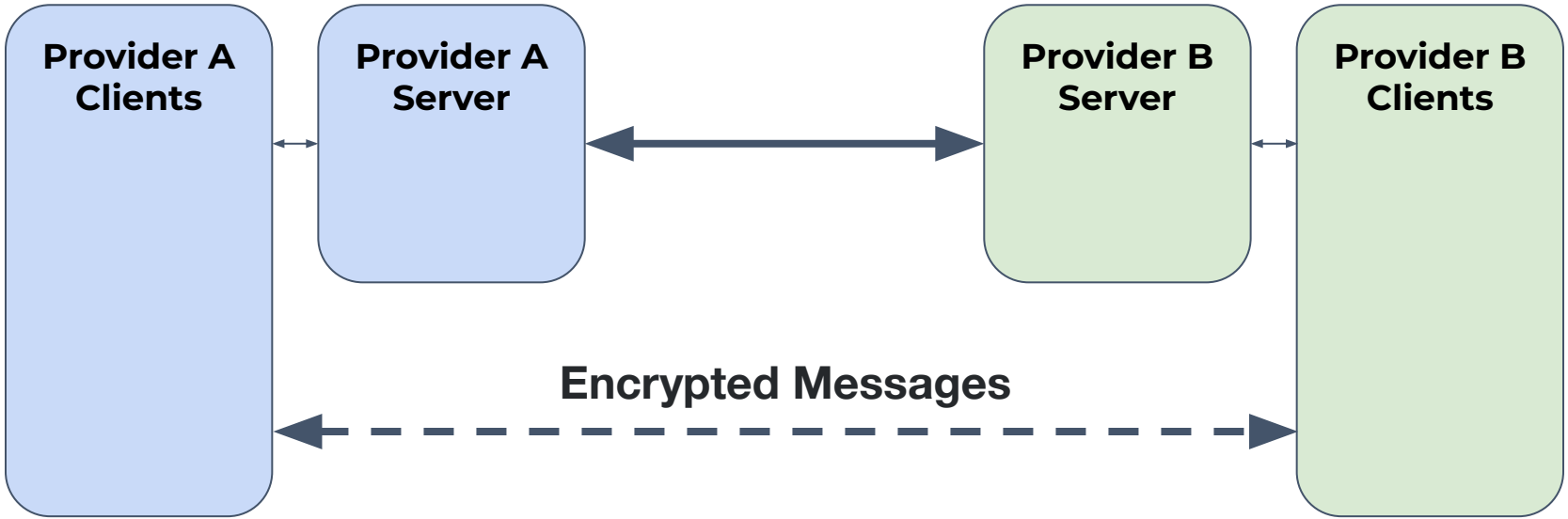
DMA-style protocol interoperability requires 4 major pieces:

1. **Encryption** - how are we securing messages?
2. **Content format** - What does a message *actually* look like?
3. **Authorization policy** - who is allowed to do things?
4. **Transport** - surely we need to ship the messages somewhere.



Room model

- Combination of encryption, authorization policy, and transport.
- Defines notion of membership/participation.
- Fanout considerations are made here.



Briefly: Transport

- MIMI and LM have placeholder transport definitions, but both use some form of TLS-secured HTTP.
- The upcoming I-D.ralston-mimi-protocol draft will have a better HTTP layer, but not final still.
- MIMI prefers mTLS for authentication; [Linearized] Matrix uses the existing signing key infrastructure already in use.
- Providers generally prefer binary instead of JSON.
- Scalability is a major consideration.
- => **Transport is universally some binary-over-HTTPS mechanism.**
- TBD what Matrix's binary event format would look like. Considering protobuf and CBOR currently.
    - Binary events would start at a federation level before impacting clients. Clients can still expect JSON initially.

Briefly: Authorization Policy

matrix

- MIMI does not define an authorization policy (yet).
- Role-based access control (RBAC) is extremely popular/important.
  - MSC4056 (Decentralized RBAC) / MSC2812 (Roles as State Events)
- Consistency and extensibility are important to ensure providers are not arbitrarily blocking messages.
- [Linearized] Matrix uses the existing Authorization Rules to accept events.
  - https://spec.matrix.org/v1.9/rooms/v11/#authorization-rules
- => **Matrix's authorization model already exists and works.**
- TBD what MIMI ends up with.

Encryption

matrix

- Most messaging providers use libsignal or something Double Ratchet flavoured.
- Matrix implemented libsignal-like Double Ratchet back in 2015 as **Olm**.
- Olm isn't interoperable with libsignal … but that's fixed with vodozemac's X3DH support and other similar deltas - "interolm".
- **Megolm** is used for group chats, simplifying the key distribution to a series of Olm sessions and a common group key.
- Double Ratchet relies on surrounding infrastructure to determine *who* to encrypt to, and doesn't scale well to large groups.
- Messaging providers, and Matrix, want to switch to MLS *eventually*.
  - https://arewemlsyet.com
- MLS has a built-in idea of membership, but has no auth policy by default.

MLS

matrix

- Specified by the IETF as RFC 9420.
- Non-cryptographer's crash course here: https://travisr.notion.site/MLS-Crash-Course-1d5d03ca629948c1aaf661d1c2036681
- Has a concept of client membership using a binary tree.
- Each client receives key material for messages it has visibility on.
- Faster than Olm/Megolm in most cases.
- Clients add and remove each other at will, if left ungoverned.
- Supports extensions to add arbitrary complexity.
- Adopted and mandated by MIMI as *the* encryption layer.
- Decentralized environments will need DMLS or similar.

Membership

matrix

- *Users* join rooms, but *clients* encrypt messages.
- MLS and Double Ratchet deal with *clients* (primarily).
- When a user joins a room, all of their clients join as well.
- => We need to synchronize membership at two levels.
- We consider users to have a **participation state**, and clients to have **membership**. Each has their own **list** of entities.
- Changes to participation must be atomic, otherwise users join the crypto state illegally.

MLS + Participation

matrix

- MIMI is proposing a set of new MLS extensions for persisting application state inside the MLS group.
  - https://bifurcation.github.io/ietf-mimi-protocol/draft-ralston-mimi-protocol.html#name-mls-application-state-synch
- Clients propose changes to application state with AppSync MLS proposals.
- Servers can see application state changes.
- Clients apply changes with precise order and behaviour.
- **In Matrix terms, AppSync is state events stored inside MLS.**

Double Ratchet + Participation

matrix

- Participation and membership are stored external to the encryption.
- In Matrix, these are m.room.member state events and device lists.
- In MIMI, these would be AppSync-shaped diffs against a static blob shared between servers (or similar).
- Adding confirmation/security around changes is difficult, but not impossible.
  - See MSC4080 & MSC3917
- Whichever protocol, ramping from Double Ratchet to MLS is a natural evolution of the application.

Content format

matrix

- What clients end up encrypting/decrypting when sending to each other.
- Needs to be well specified, otherwise clients don't know what to do.
- Extensibility is required to support the infinite combinations of messaging features.
- Server can't verify schema because it's encrypted - clients need to do their own parsing and error handling.
- Should require minimal bytes and processing power to encode/decode.
- MIMI is working on their own TLS-encoded multipart MIME format
  - https://datatracker.ietf.org/doc/draft-ietf-mimi-content/
- Matrix already has events with a loose schema.
  - … but what if we made that schema way more *extensible*?

Extensible events (MSC1767)

- Uses **content blocks** to persist information inside an event.
- Core blocks, like text and files, are defined by the Matrix Specification.
- Other blocks are added as-needed to represent the datum.
- Clients which know the event type look for the blocks they need to render that datum.
- Clients which *don't* know the event type look for a collection of blocks which match an event type schema they do know, then render that.
- Events typically contain a text block so they are renderable in the worst case.
- Richness is lost the further a client falls back, but this is better than the user being left out of the conversation.

Extensible events

```
{
   // irrelevant fields not shown
   "type": "m.message",
   "content": {
      "m.text": [
         { "body": "<i>Hello world</i>", "mimetype": "text/html" },
         { "body": "Hello world" }
      ]
   }
}
```

Extensible events

matrix

```
{
   // irrelevant fields not shown - see MSC3381 for actual polls schema
   "type": "org.matrix.poll_start",
   "content": {
      "m.text": [{ "body": "What should we have for lunch? 1. Pizza 2. Poutine" }]
      "org.matrix.poll": {
         "question": "What should we have for lunch?",
         "options": [ "Pizza", "Poutine" ],
      }
   }
}
```

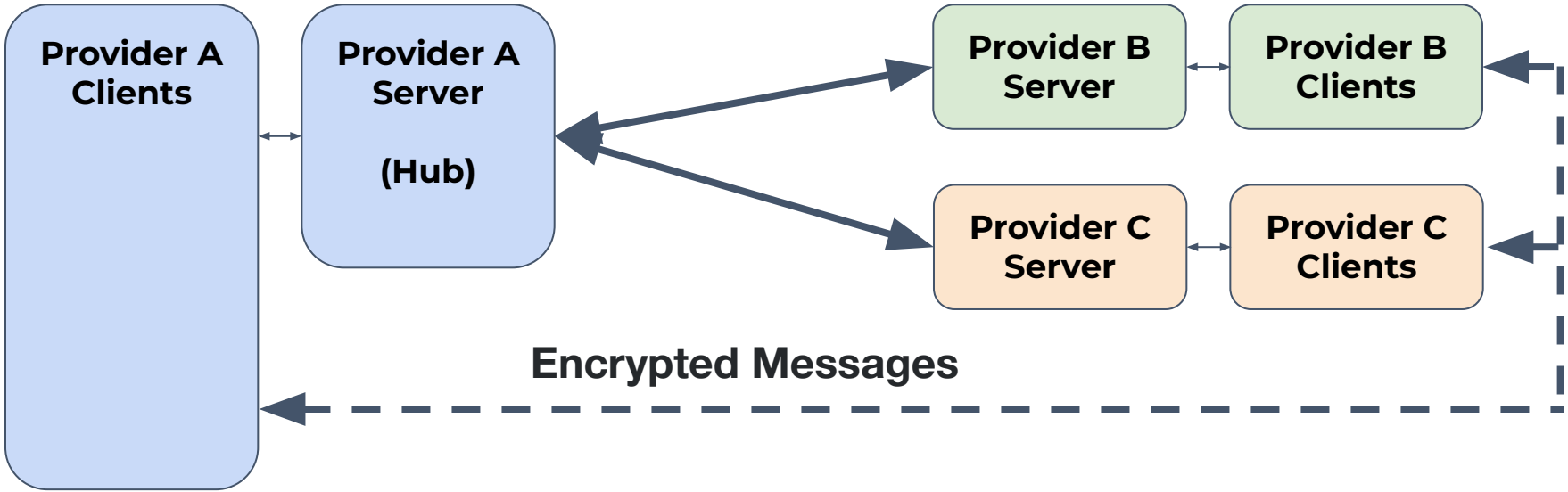
Extensible events

- Currently JSON, could be protobuf or other binary format in future.
- More events get rendered by more clients.
- Create custom event types more easily.
- Use the content blocks which make sense for your event.
- Rapidly iterate on MSCs and new features, though at a cost of potentially reduced interactivability in unsupported clients.
- TBD what an event types/content blocks registry might look like.

Room models

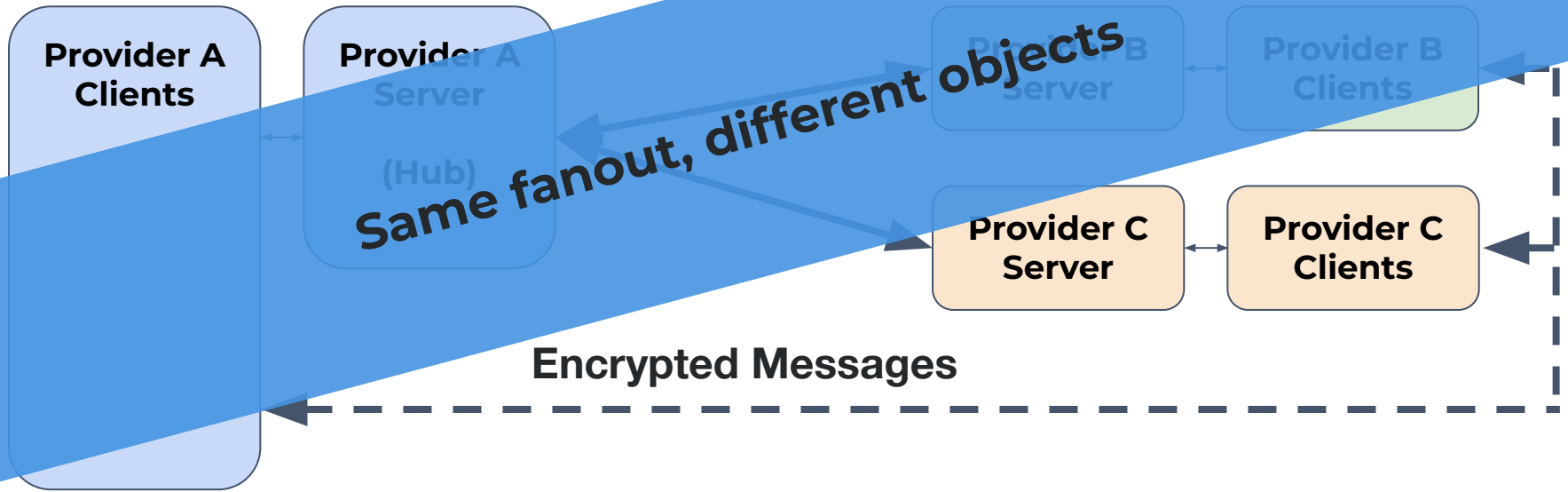
MIMI: Room model

- Uses a ‘hub and spoke’ fanout.
- Hub server enforces policy and distributes messages.
- Follower servers communicate through hub server whenever possible.



Linearized Matrix: Room model

- Uses a ‘hub and spoke’ fanout.
- Hub server enforces policy and distributes messages.
- Follower servers communicate through hub server whenever possible.



Linearized Matrix

- Uses regular Matrix events and room state.
- Uses a stripped down version of the server-to-server API.
- Uses the same authorization rules.
- Uses a linked list instead of a DAG for history (MIMI doesn't have history).
- Can use MLS or Double Ratchet (or something else) as needed.
- Same extensibility capabilities from Matrix.
- **Supports having DAG-capable servers in the same room.**

Decentralization and DAGs

matrix

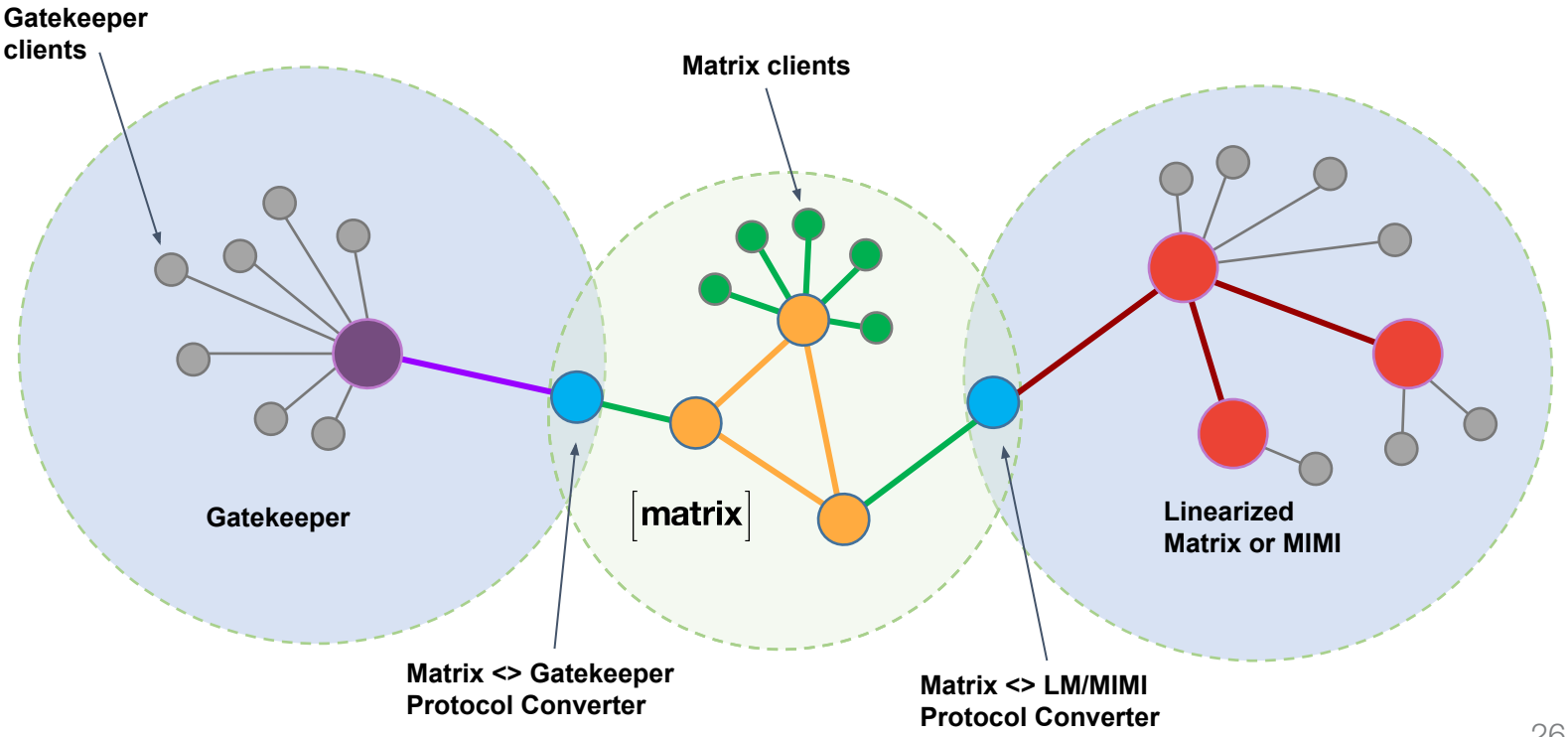
- Matrix uses a Directed Acyclic Graph (DAG) to persist events.
- Fanout is full mesh instead of hub-and-spoke.
- Conflicts in the DAG are resolved using **state resolution**.
- State resolution can also be used to linearize the DAG.
- Through use of a **protocol converter**, centralized systems can be brought into Matrix for further routing.

Protocol conversion

Not bridges

- Bridges necessarily break encryption to convert to the remote network's encryption algorithm.
  - Example: Converting Signal to Matrix (used to) require decrypting both networks *inside* the bridge.
- Protocol converters do *not* decrypt messages. They instead translate the envelope to the appropriate format for the remote network.
- May include translating concepts as well, such as using to-device on Matrix instead of using room events all the time.
- Converters use the appservice API, or are a dual-stack homeserver.
- May use MSC3983 and MSC3984 to bridge cryptographic algorithm differences, specifically querying/claiming keys on a remote network.

Matrix with protocol converters



Missing pieces

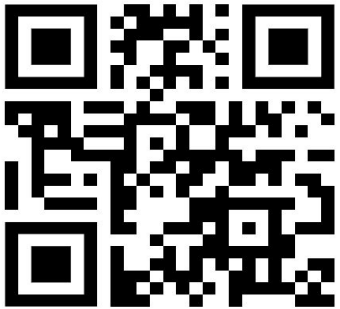
What we haven’t talked about

matrix

- **Identity** - Converting a phone number/email/name into a routable ID.
- **Consent** - People might want to have a bit more control over who messages them.
- **Anti-abuse** - Both reporting and actual anti-spam are needed.
- **Identifiers** - Matrix has known-good identifiers, but MIMI wants to consider new ones.
- **Room metadata** - Where does the room name go? Is it server-visible state, or part of content format? Please use Matrix state events?
- **Ordering** - How guaranteed does the message order need to be? MLS would like it to be strictly linear, but does it really need to be?

What's next?

- No idea! :D
- Linearized Matrix will get updated as an MSC, maybe.
- Gatekeepers will publish their DMA plans by March 7th, 2024.
- Protocol converter concept will continue to be refined.
- MIMI continues to make progress and become refined.
- Funding the Matrix.org Foundation is a great way to support this work.



https://matrix.org/membership/

Thanks

Travis Ralston
Director of Standards Development - matrix.org
@travis:t2l.io | travisr@matrix.org