

[matrix]

Next Gen Matrix

The Matrix Conference 2026

matthew@matrix.org

@matthew:matrix.org

Where do we want to go next?

- **Stop servers controlling group membership.**
- **Minimise metadata on servers.**
- **Get Post Quantum Matrix.**
- **Improve Matrix uptake.**
- **Avoid fragmentation.**

**Mapping out our options for
client controlled group
membership...**

Today

Client

Matrix Events

E2E Encryption
(Megolm)

Server

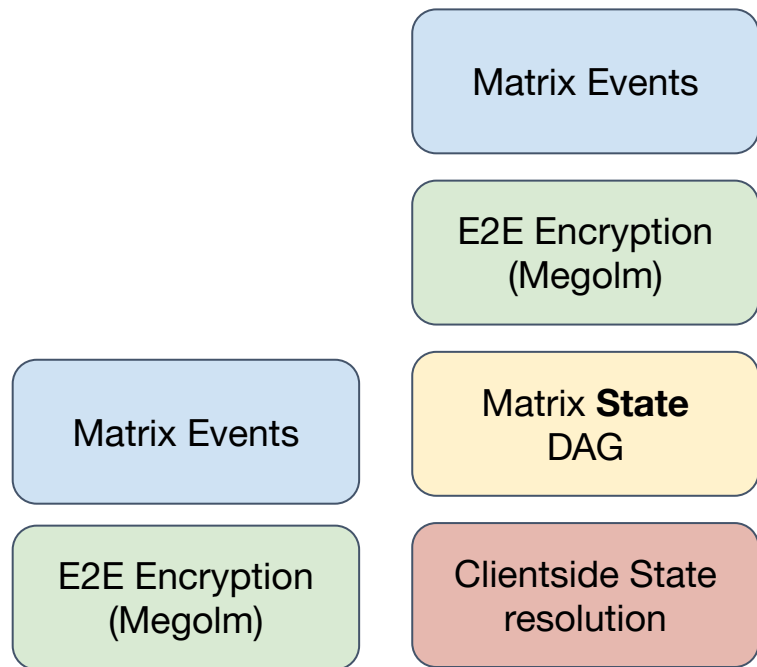
Matrix Event
DAG

Matrix DAG
Replication (State
resolution)

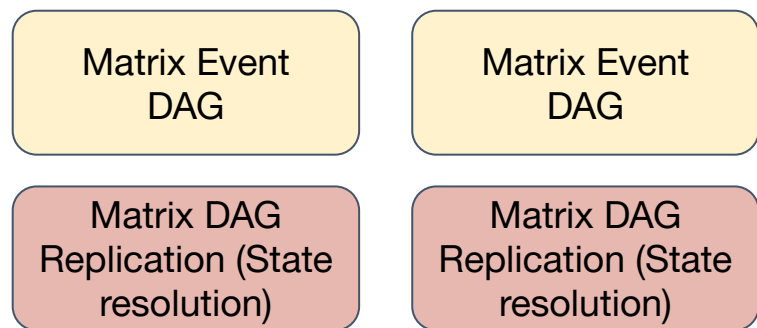
Today

Lepton

Client



Server

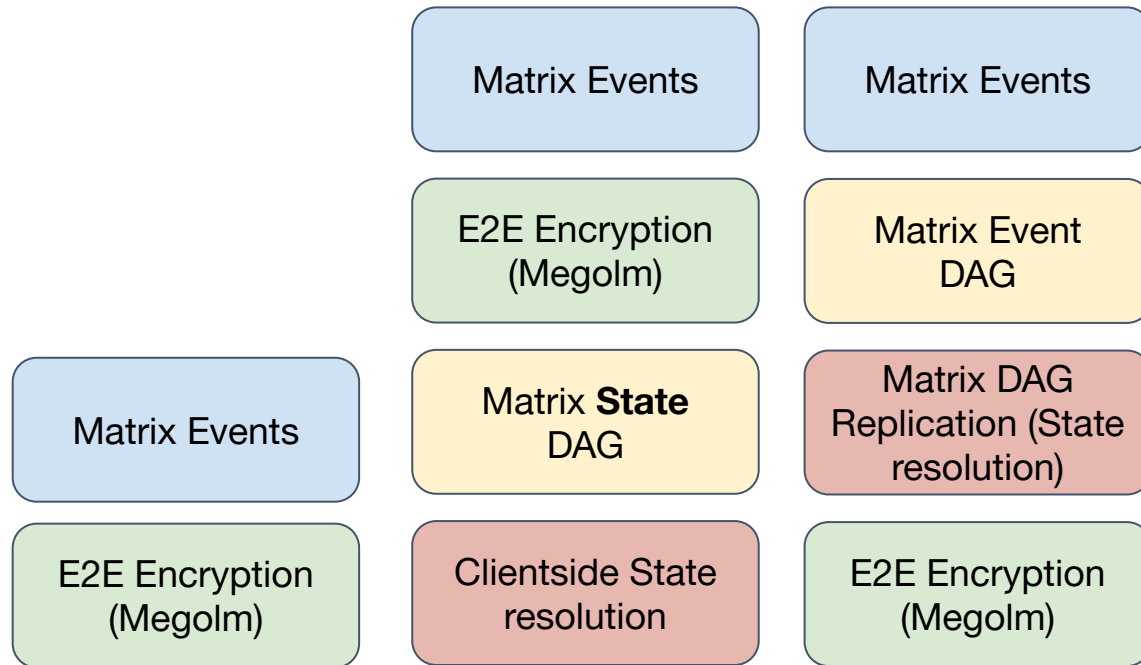


Today

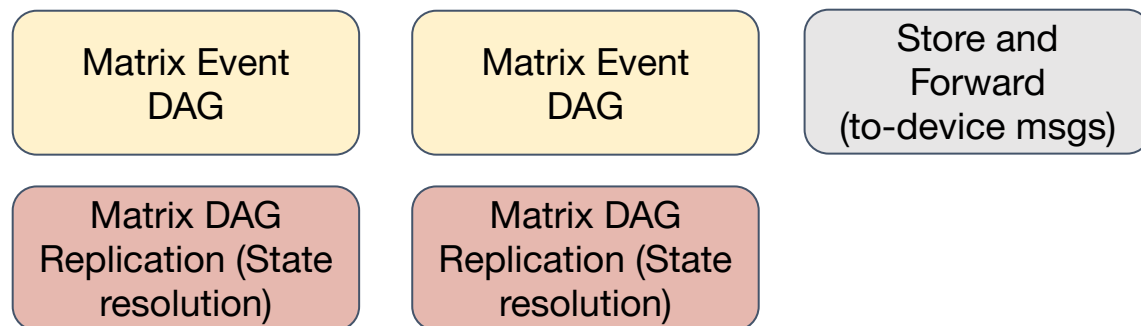
Lepton

Photon

Client



Server



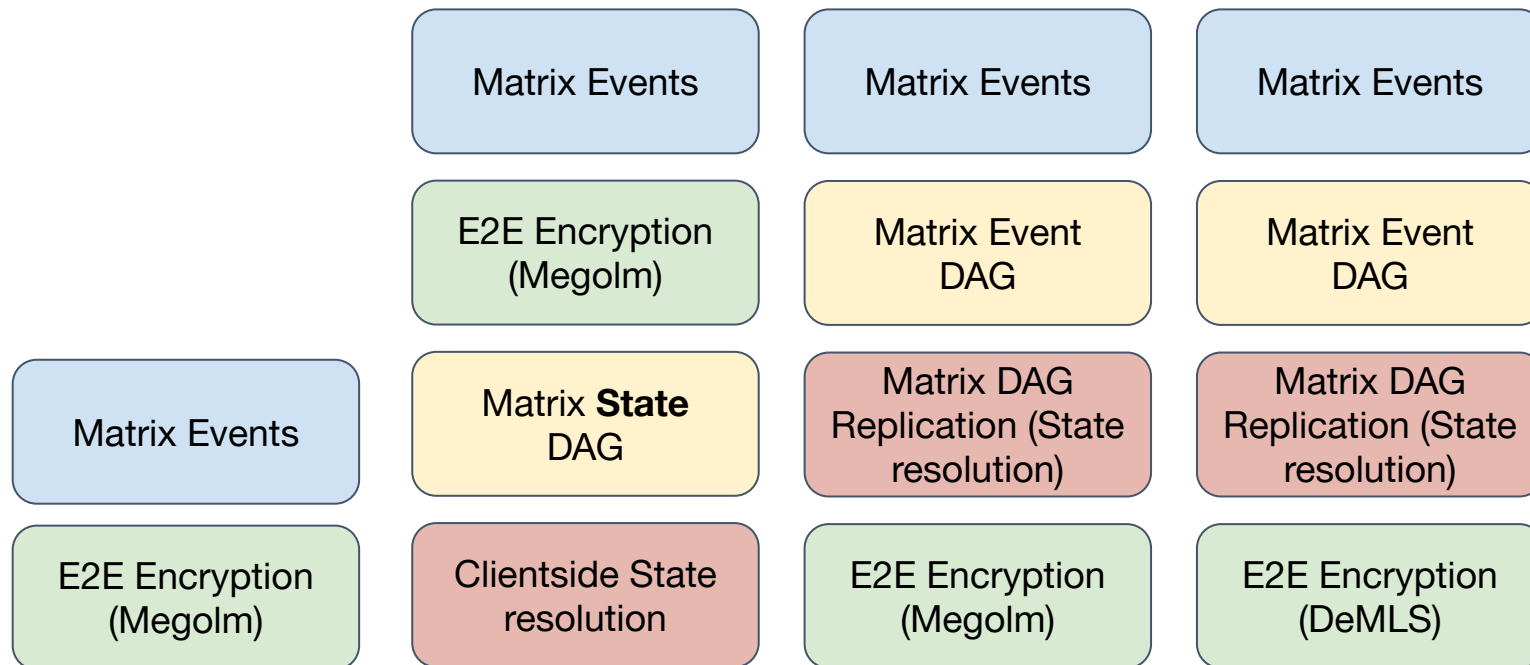
Today

Lepton

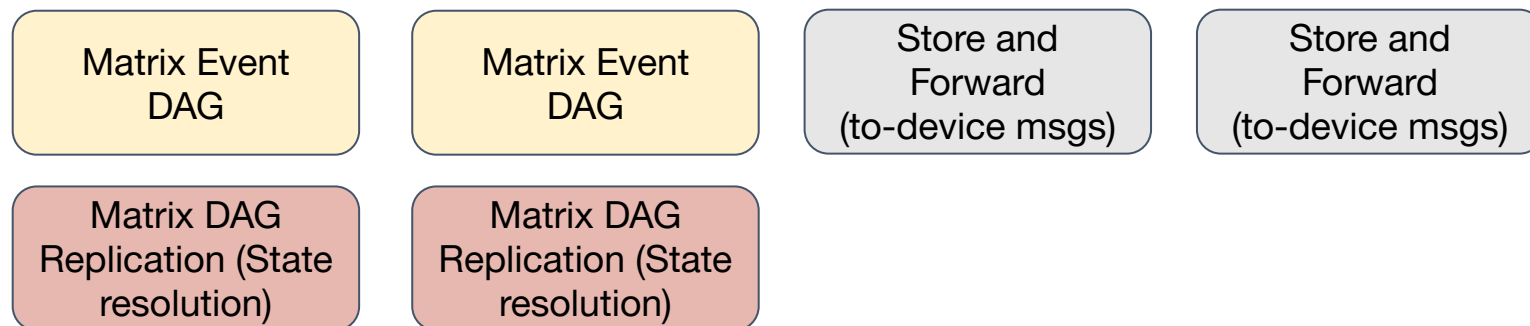
Photon

Tachyon

Client



Server



Today

Lepton

Photon

Tachyon

Neutrino

Client

Matrix Events

E2E Encryption
(Megolm)

Matrix Events

E2E Encryption
(Megolm)

Matrix **State**
DAG

Clientside State
resolution

Matrix Events

Matrix Event
DAG

Matrix DAG
Replication (State
resolution)

E2E Encryption
(Megolm)

Matrix Events

Matrix Event
DAG

Matrix DAG
Replication (State
resolution)

E2E Encryption
(DeMLS)

Matrix Events

E2E Encryption
(Megolm)

Matrix Event
DAG

Matrix DAG
Replication (State
resolution)

Server

Matrix Event
DAG

Matrix DAG
Replication (State
resolution)

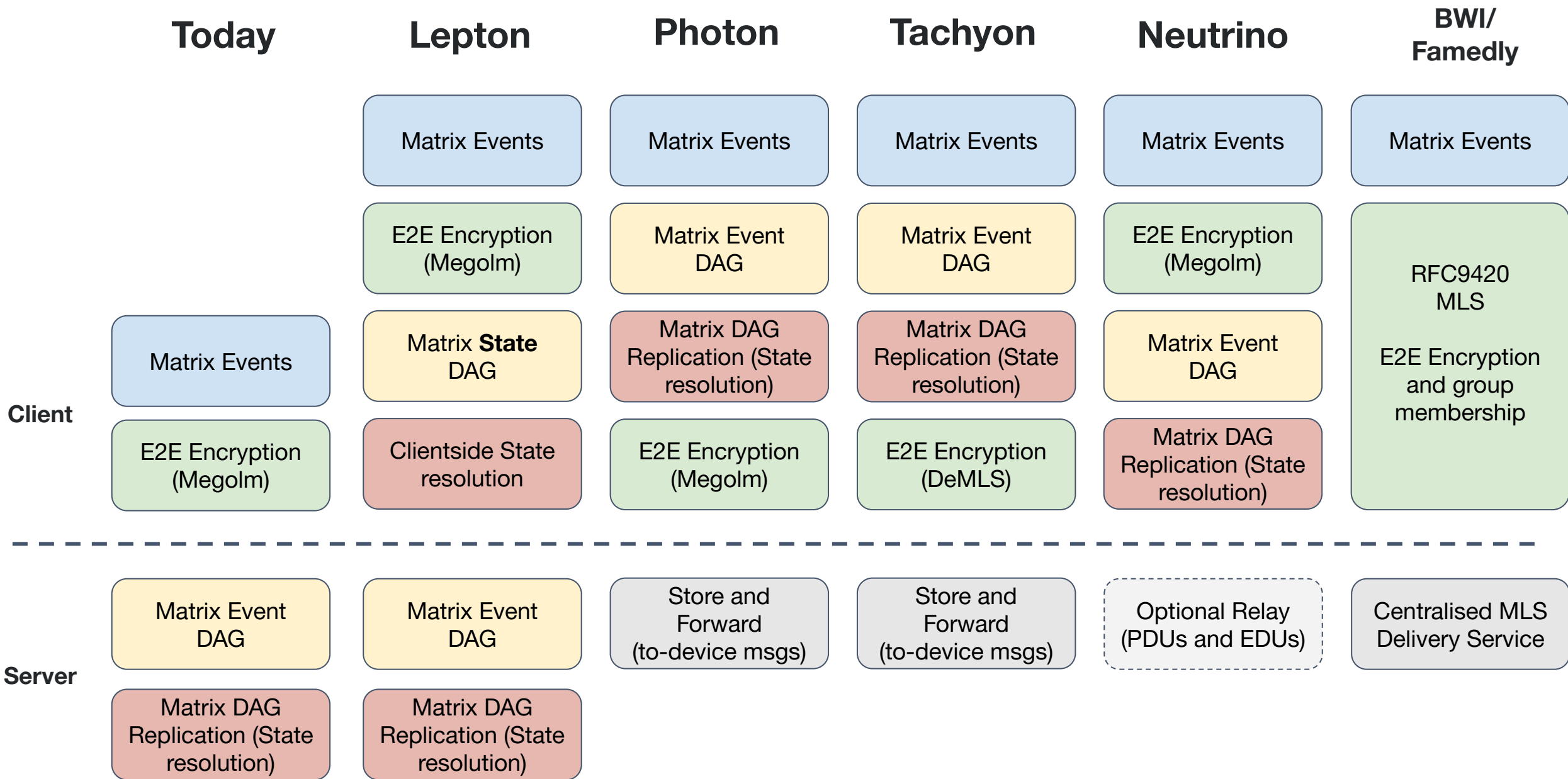
Matrix Event
DAG

Matrix DAG
Replication (State
resolution)

Store and
Forward
(to-device msgs)

Store and
Forward
(to-device msgs)

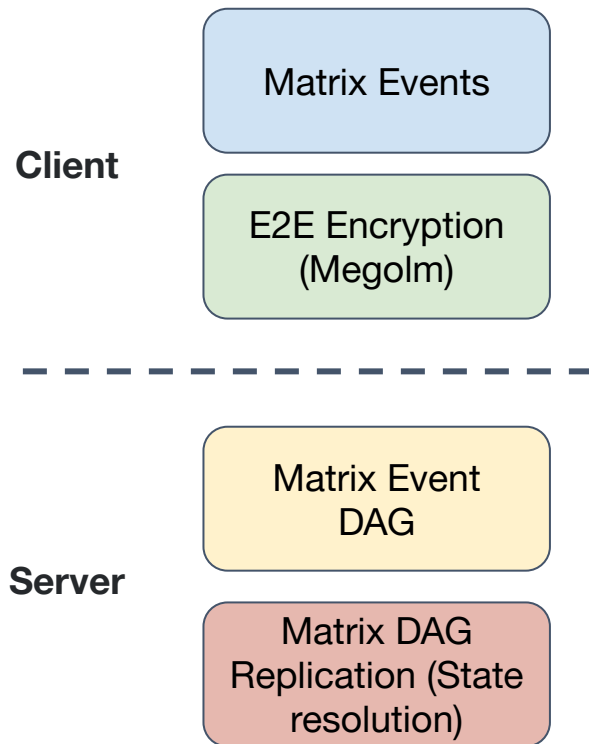
Optional Relay
(PDUs and EDUs)



**It's all about the balance
between client and server.**

Today

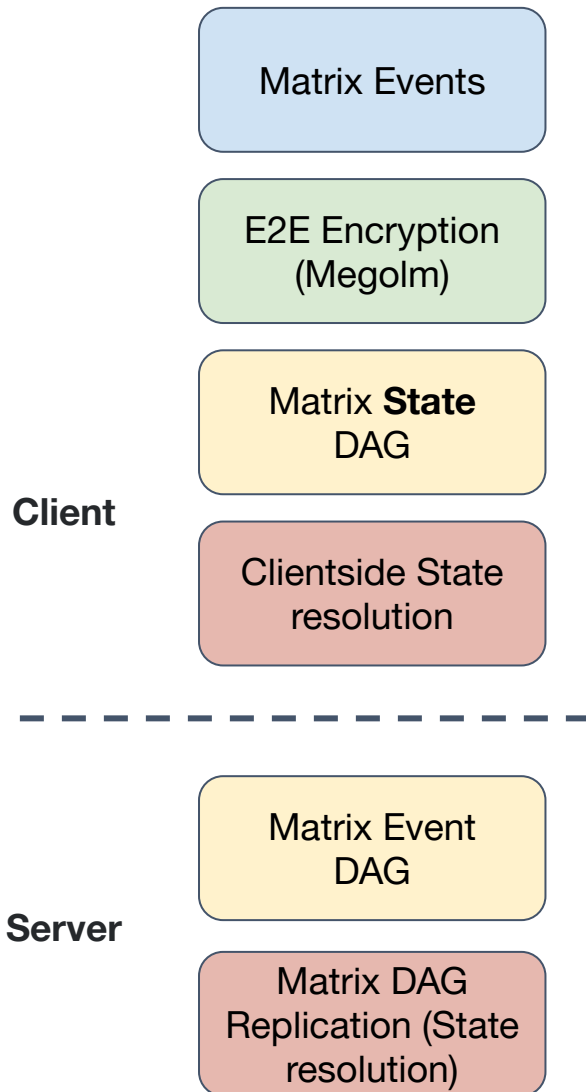
[matrix]



- Originally built pre-E2EE.
- We wanted to make the clients as thin and easy and fun as possible to write, to help Matrix take off.
- (It worked!)
- Very focused on server-side history to preserve knowledge rather than messaging.
- Olm selected for compatibility with other platforms using Signal Protocol (Signal, WhatsApp, Skype, OMEMO, various Google chat apps, etc)
- Matrix State Resolution is a protocol for managing group membership (amongst other key-value state) in a zero-trust decentralised manner.
- However, it runs on the server, not the client.
- The time of clients being simple is long gone.

Lepton: Clientside State Resolution

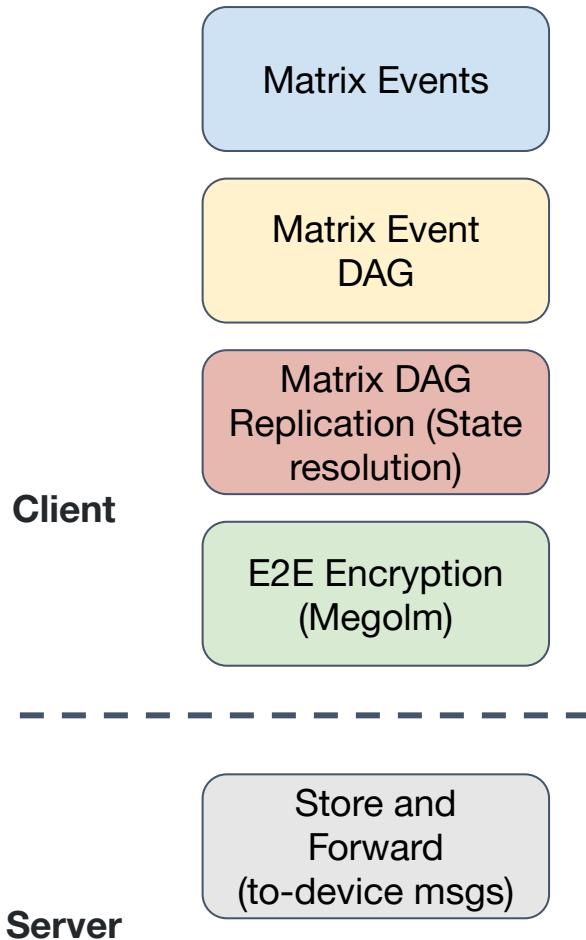
[matrix]



- What if we signed (or attested) membership events on the client, to let the client control them?
- We still need to trust the server not to lie about state resolution.
- So we'd need to participate in state resolution too.
- Lepton is an experiment to expand matrix-rust-sdk to:
 - Stores Matrix State DAG clientside (in event cache + temporal state storage)
 - Receive full federated State DAG events from Synapse
 - Use Neutrino's state res code to calculate its own state irrespective of Syn.
- Advantages:
 - Simplest incremental approach from where we are today
 - +4.2K LOC (+2.1K of Neutrino state res). rust-sdk is around 200K LOC atm.
- Disadvantages:
 - Doesn't help with metadata footprint at all.
 - State res on the server becomes redundant.

Photon: Tunnel Matrix over Megolm

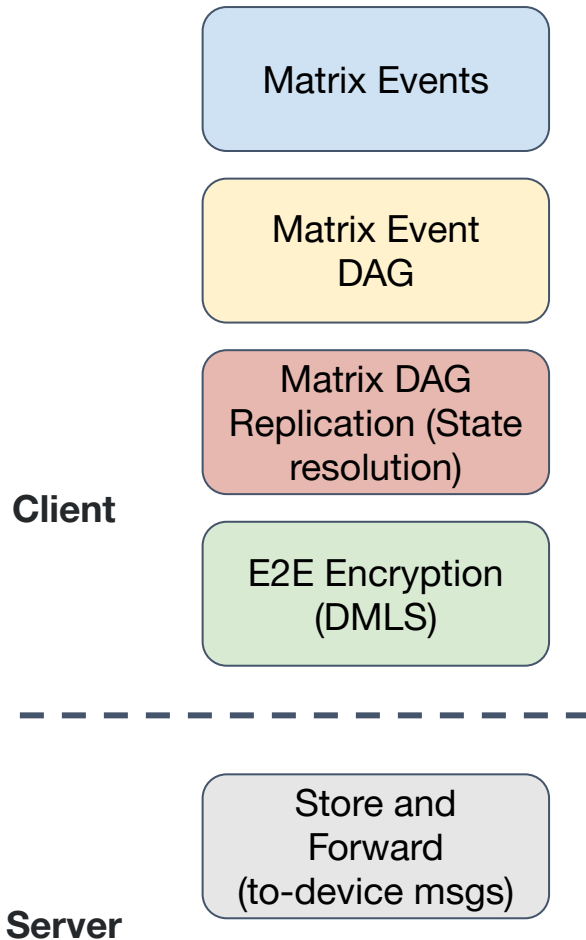
[matrix]



- What if we do State Resolution only on the client, and move encryption **below** all the resulting events - the server only forwards megolm payloads over to-device.
- Advantages:
 - Server sees minimal room metadata; it just store & forwards traffic to devices; same as Signal.
 - You get client-controlled group membership, and you don't have to replace the whole E2EE stack! Retains full decentralisation.
 - Retains E2EE agility; Matrix drives group membership control.
 - +7.1K LOC change (relative to +4.2K LOC for Lepton)
- Disadvantages:
 - No server synchronised history.
 - Clients would have to gossip history between each other (like Signal)
 - Sync gets less efficient, as server doesn't have the metadata to know which events are most important to the client.
 - Doesn't benefit from MLS's post compromise security or alg improvements.

Tachyon: Tunnel Matrix over DeMLS

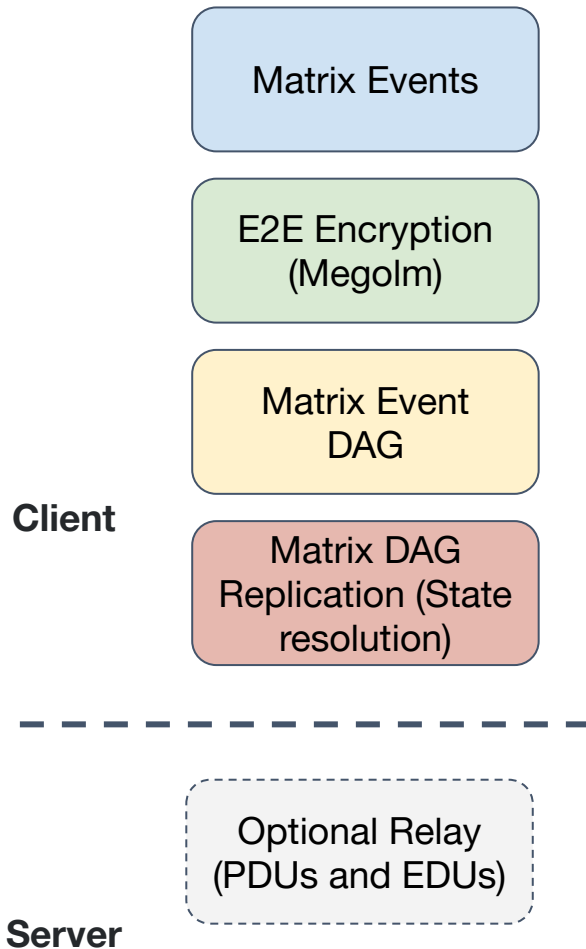
[matrix]



- Same architecture as Photon, but this time we put the traffic over Decentralized MLS (<https://datatracker.ietf.org/doc/draft-kohbrok-mls-dmls/>).
- Advantages:
 - Server sees minimal room metadata; Client retains full decentralization.
 - Benefits from logarithmic scaling of MLS (in theory)
 - Benefits somewhat from academic & formal verification work on MLS
 - Could combine with Hydra Finality to support vanilla MLS
- Disadvantages:
 - Still no server synchronised history.
 - Still less efficient sync.
 - +21.2K LOC in rust-sdk plus 33K LOC of OpenMLS (DMLS branch)
 - DeMLS is experimental.

P2P & Neutrino: A world without servers

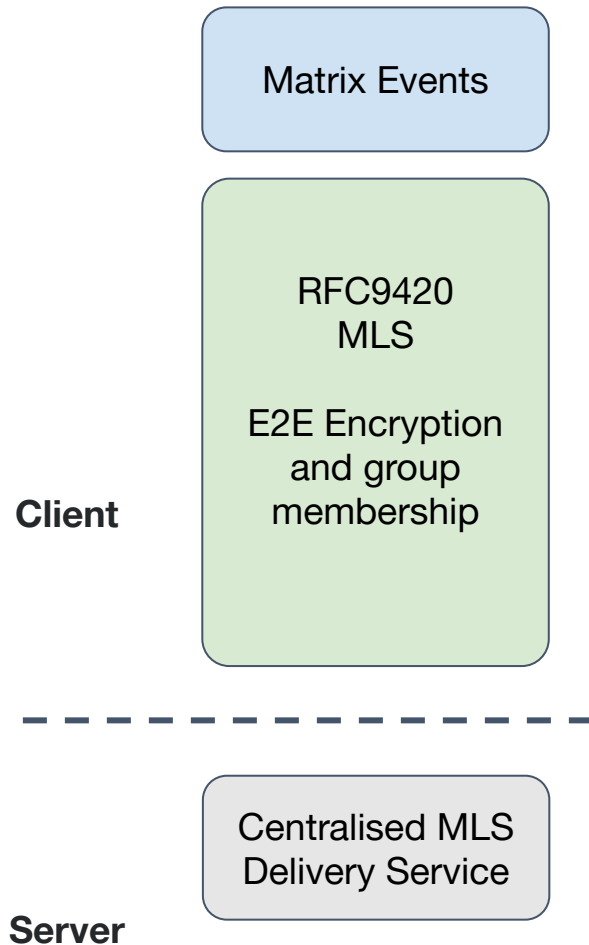
[matrix]



- It's just a tiny clientside rust homeserver: Neutrino.
- Client talks CS API to the local server.
- Server talks SS API to other instances.
- Advantages
 - Zero client changes or encryption changes. (13.8K LOC of Neutrino)
 - Optimal metadata footprint - no servers involved at all if you can talk direct
 - Doesn't require internet.
 - Retains E2EE agility (including no E2EE on trusted networks)
 - Compatibility possible with normal Matrix
- Disadvantages
 - Full-mesh routing is inefficient and doesn't scale, so better approaches are needed (e.g. application-layer fan-out or relay fanout)
 - If you include a serverside node in the room, then metadata benefits evaporate.

BWI & Famedly

[matrix]



- If I've understood MSC4256 "RFC 9420 MLS mode Matrix" correctly...
- Throws out Matrix state resolution in favour of vanilla MLS
- Stores Matrix state as MLS application data
- MLS then requires a (logically) centralised delivery service to sequence MLS commits linearly.
- Advantages:
 - Makes accreditation for VS-NfD classified communication easier.
 - Benefits from academic and formal verification of vanilla MLS
 - Avoids having to sync MLS membership with Matrix membership, as there is no longer any concept of Matrix membership.
- Disadvantages:
 - Hardlocks the protocol to MLS.
 - Centralises rooms on a delivery service.
 - Breaks compatibility with Matrix and decentralised comms.

Other approaches...

- Just replace Olm/Megolm with DMLS
 - Did this, but threw it away, given it doesn't solve client-controlled group membership.
- Keyhive from Ink & Switch
 - Stores E2EE CRDTs serverside but with enough metadata to sync intelligently
- Tardigrade from Erik
 - DeMLS, but persisted in the serverside DAG
 - Encrypts per-user, not per device.
 - Advantages:
 - No more UTDs due to transitive key delivery failures
 - Disadvantages:
 - Data and metadata accumulates on the server, and so metadata protection is harder.
 - Doesn't solve client-controlled group membership (unless combined with Lepton etc)

Let's talk P2P

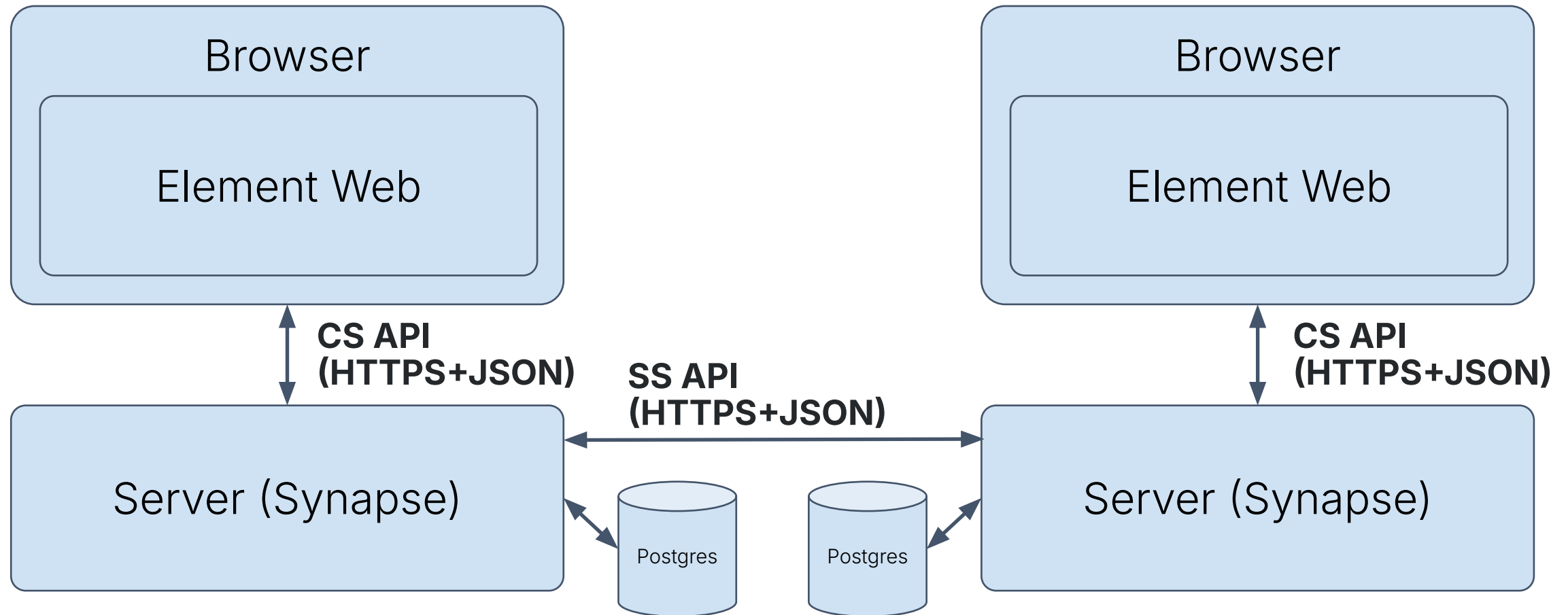
P2P Matrix: 2020-2023

[matrix]

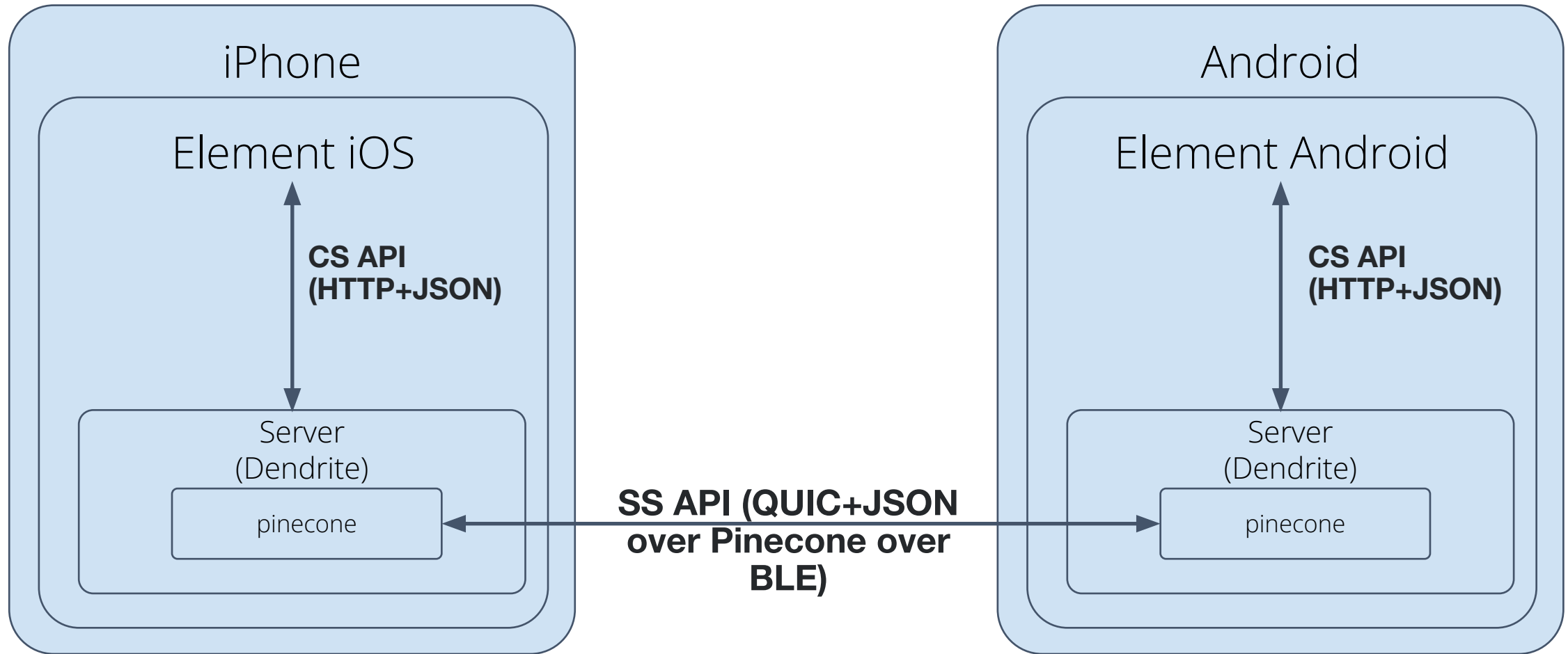
- “What if you ran your Matrix server inside the client?”
- Embedded Dendrite (golang server) inside Element Classic iOS & Android
- Fixes metadata exposure to servers... by removing the server.
- Solves client-controlled cryptographic group membership (as state resolution runs clientside)
- Supports fully disconnected mesh networks, and inoculates against surveillance capitalism
- For connectivity we tried global overlay networks:
 - ...first libp2p...
 - ...then yggdrasil...
 - ...then pinecone (a yggdrasil variant supporting highly dynamic networks)
- Three main things went wrong:
 - We hit the limits of Matrix’s resilience to network partitions (at the time).
 - Building a robust P2P overlay network is an open area of research in its own right.
 - Element ran out of funding for next gen work, focusing instead on keeping the lights on.

Before P2P...

[matrix]



P2P via Dendrite + Pinecone (~2023) [matrix]



P2P Matrix, 2026 edition

[matrix]

- The Dutch Government are funding Element to work on P2P Matrix once more!
 - We've fixed Matrix to be fully resilient to network partitions (Hydra)
 - We're not going to build an overlay network.
 - Instead, we'll target smallish P2P meshes, which can default to using normal Matrix servers for store-and-forward.
- One possible approach:
 - For instance, using multihomed accounts, imagine if @bob:example.com could live on both his normal example.com homeserver but *a/so* on a server on his phone?
 - Then if @alice:example.com wants to talk to him, she could try looking him up on whatever mesh she's on - e.g. using Multicast DNS (Bonjour / Zeroconf), or BLE broadcast, etc.
 - If she can talk direct, then hooray - we have P2P! If not, she can relay via the example.com homeserver like normal Matrix today.

Introducing Neutrino

- **So, we've built a new P2P Matrix stack.**
 - Element X (rather than Element Classic)
 - Powered by matrix-rust-sdk
 - Talking to a new, tiny, embedded Matrix server implementation in Rust: Neutrino
 - <https://github.com/element-hq/neutrino>
 - Uses CoAP as a low bandwidth transport.
 - Uses whatever overlay is available: currently using
 - Iroh for “connect to this key please” and
 - BLEW for BLE mesh connectivity
- This comes with **very important caveats**
 - This is very early and experimental
 - Not built or tested yet for untrusted networks.

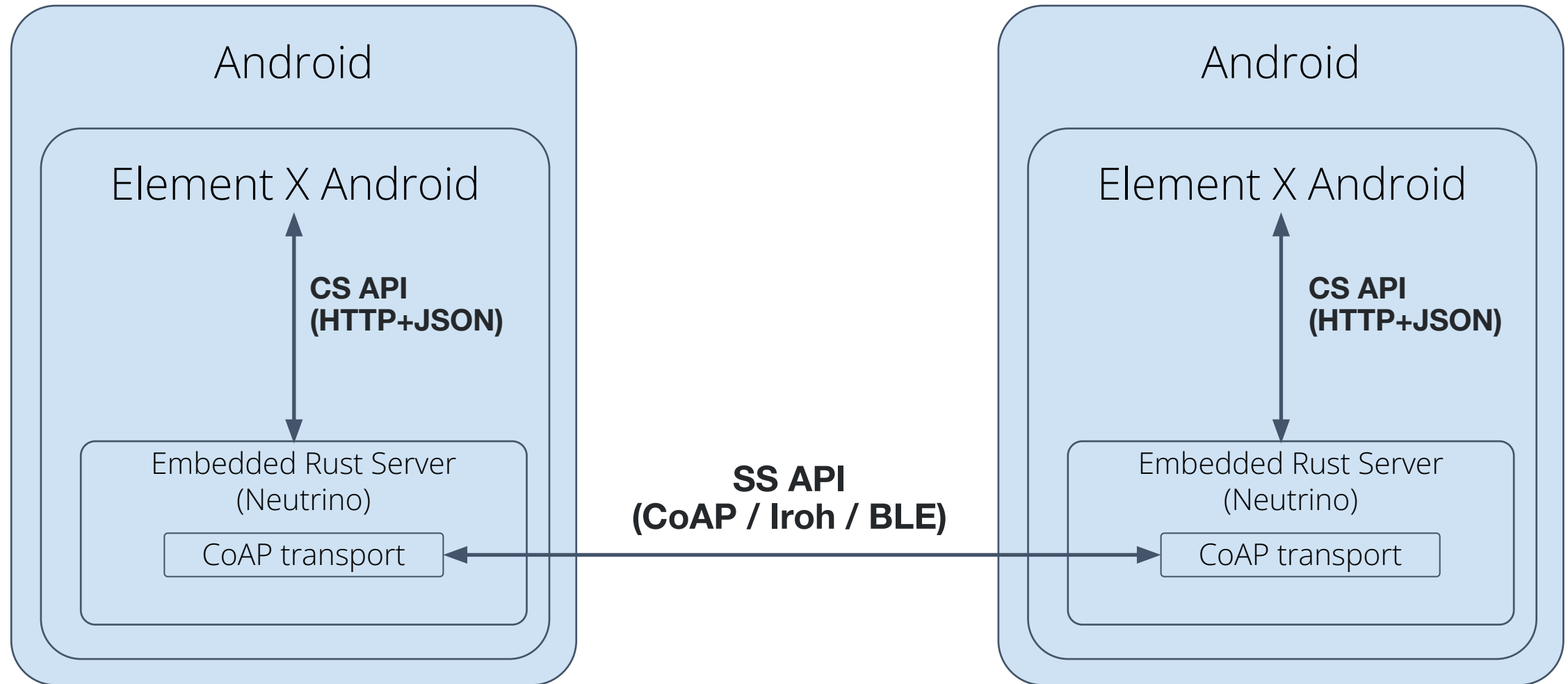
Neutrino P2P Demo

(26.05.2-r93+neutrino.0.8.2)



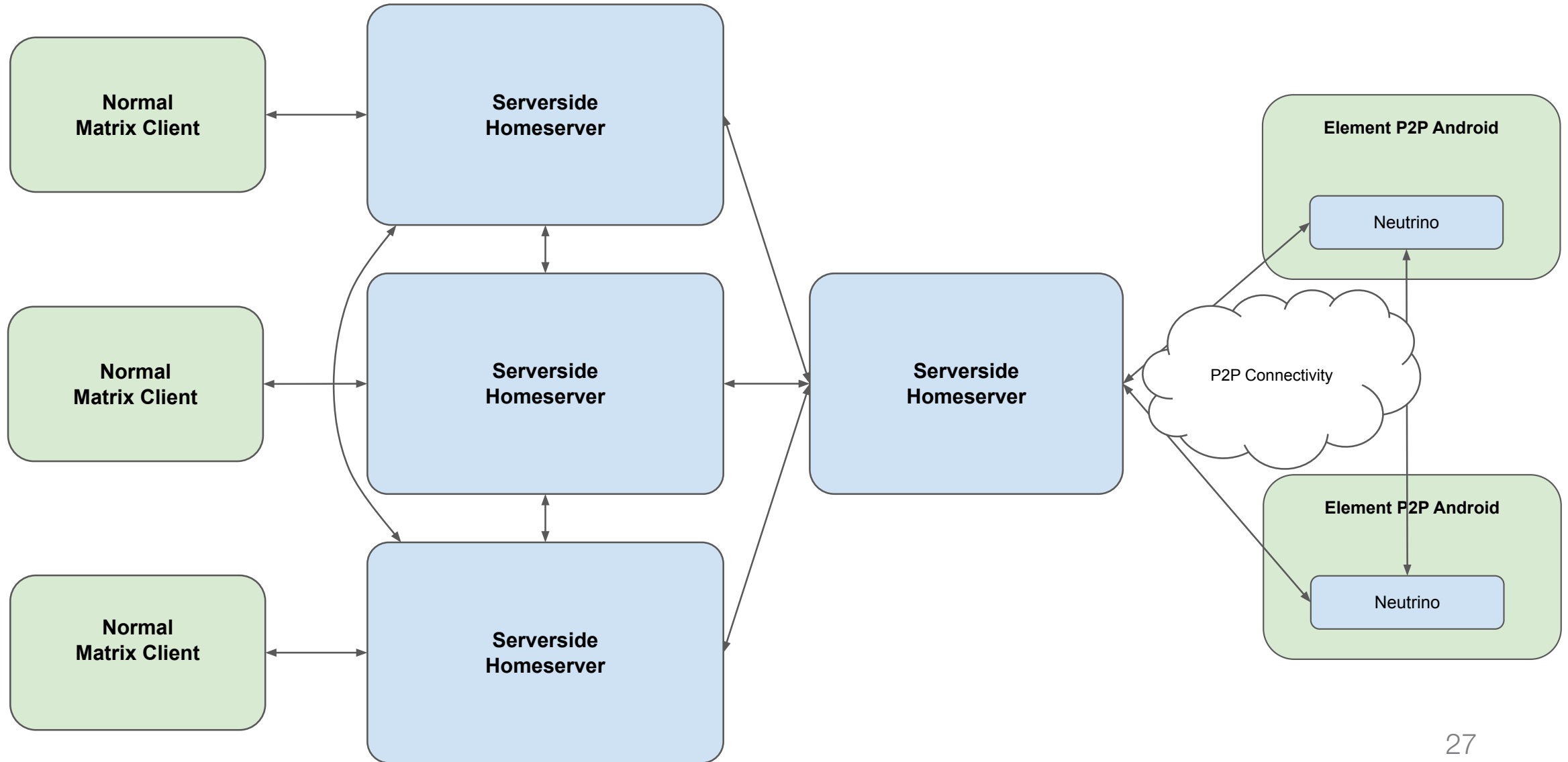
P2P via Neutrino

[matrix]



A brave new P2P Hybrid World

[matrix]



Why Neutrino?

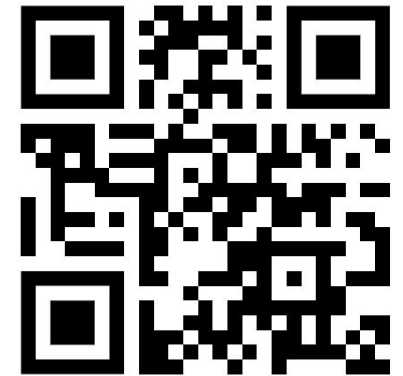
- Written specifically to experiment with P2P Matrix dialects.
- Written specifically as an embedded homeserver rather than forcing a serverside homeserver to run clientside.
- It is **not** intended to run as a standalone homeserver.
- It only implements the very latest room versions (Hydra Phase 2 and 3) - e.g. State DAGs: MSC4242)
- Gives us somewhere to evaluate Hydra and state resets in the harshest conditions (i.e. P2P)
- Neutrino is not the future of Synapse or Synapse Pro, although we expect Neutrino and Synapse to share Rust code in future
- This not a re-run of Dendrite (whose learnings got subsumed into Synapse & Synapse Pro)
- We also want somewhere to experiment freely with new ideas (e.g. temporal state storage) which can get backported and/or shared with Synapse & Synapse Pro.

What's next for Neutrino?

- Firstly, getting it fully deployed for the Dutch use case.
- Then, figuring out store & forward semantics for interop with normal Matrix
- Implementing proper member keys? (MSC4428/4430)
- Implementing multihoming?!
- Figuring out how to converge the codebase more with Synapse
- Adding more features
- Potentially: using it as a testbed for Hydra Phase 3 (ERA Finality)
- Potentially: using it as a testbed for the State Res v3 changes that go with Finality.

For Matrix to prevail, we need your support! [matrix]

- **Use it!** Friends don't let friends use closed chat services!
- **Continue to build cool stuff on Matrix!**
Clients! Servers! Bots! Bridges! Integrations!...
- **Become a Matrix vendor and grow the ecosystem!**
Host! Integrate! Develop! Resell!...
- **Join the Working Groups!**
- Getting better but the Foundation still isn't sustainable!
 - Fundraising **right now** to support core spec work, trust & safety work, running the matrix.org infrastructure and governance work
- **Join the Foundation:** <https://matrix.org/membership/>!
- **Buy Matrix solutions from vendors who fund upstream maintenance**
and development. Otherwise there will be no upstream projects.



[matrix]

Thank you!

[m] @matthew:matrix.org



matthew@matrix.org



<https://matrix.org>



[@matrix@mastodon.matrix.org](https://matrix@mastodon.matrix.org)



@matrix.org