

# [matrix]

## The Matrix State of the Union

The Matrix Conference 2026

[matthew@matrix.org](mailto:matthew@matrix.org)

@matthew:matrix.org

**Matrix has gone from being  
an interesting experiment to  
an existential requirement.**

**Never before has it been so  
important to control our  
own communication.**

EXCLUSIVE

## **US imposes sanctions on ICC hours after former judge wins Nobel Peace Prize**

By **Humeyra Pamuk, Steve Holland and Anthony Deutsch**

October 9, 2026 4:02 PM GMT+2 · Updated October 12, 2026

## **Ukraine says SpaceX limits Starlink operations, disrupting military**

By **Yuliia Dysa**

October 14, 2026 8:34 AM GMT+2 · Updated October 14, 2026

EXCLUSIVE

## **US imposes sanctions on ICC hours after former judge wins Nobel Peace Prize**

By Humeyra Pamuk, Steve Holland and Anthony Deutsch

October 9, 2026 4:02 PM GMT+2 · Updated October 12, 2026

## **Ukraine says SpaceX limits Starlink operations, disrupting military**

By Yuliia Dysa

October 14, 2026 8:34 AM GMT+2 · Updated October 14, 2026

International Criminal Court invests in open infrastructure

## **International Criminal Court switches to open source with openDesk**

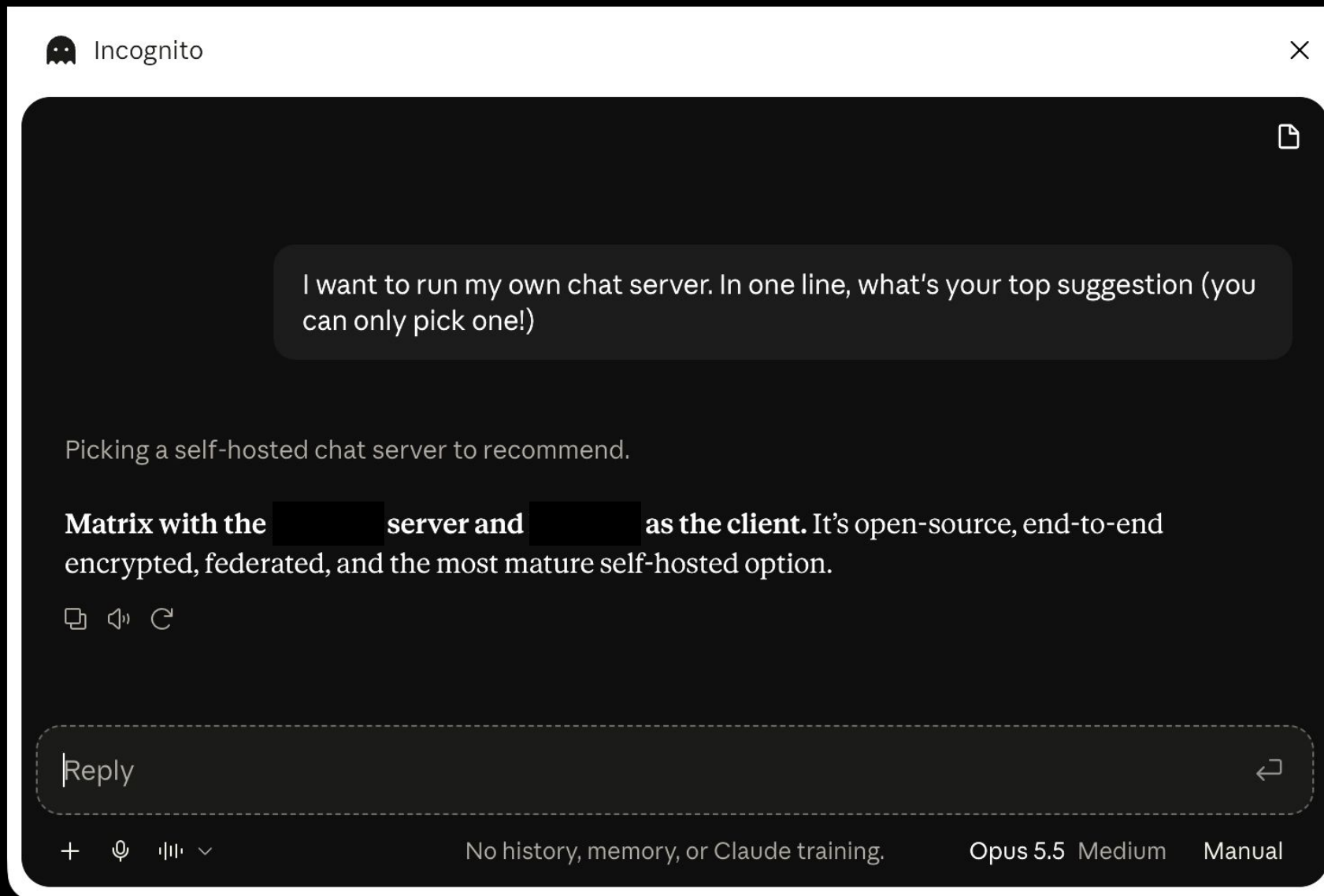
## **Ukraine plans to deploy a constellation of 300 UASAT-NANO communication satellites**

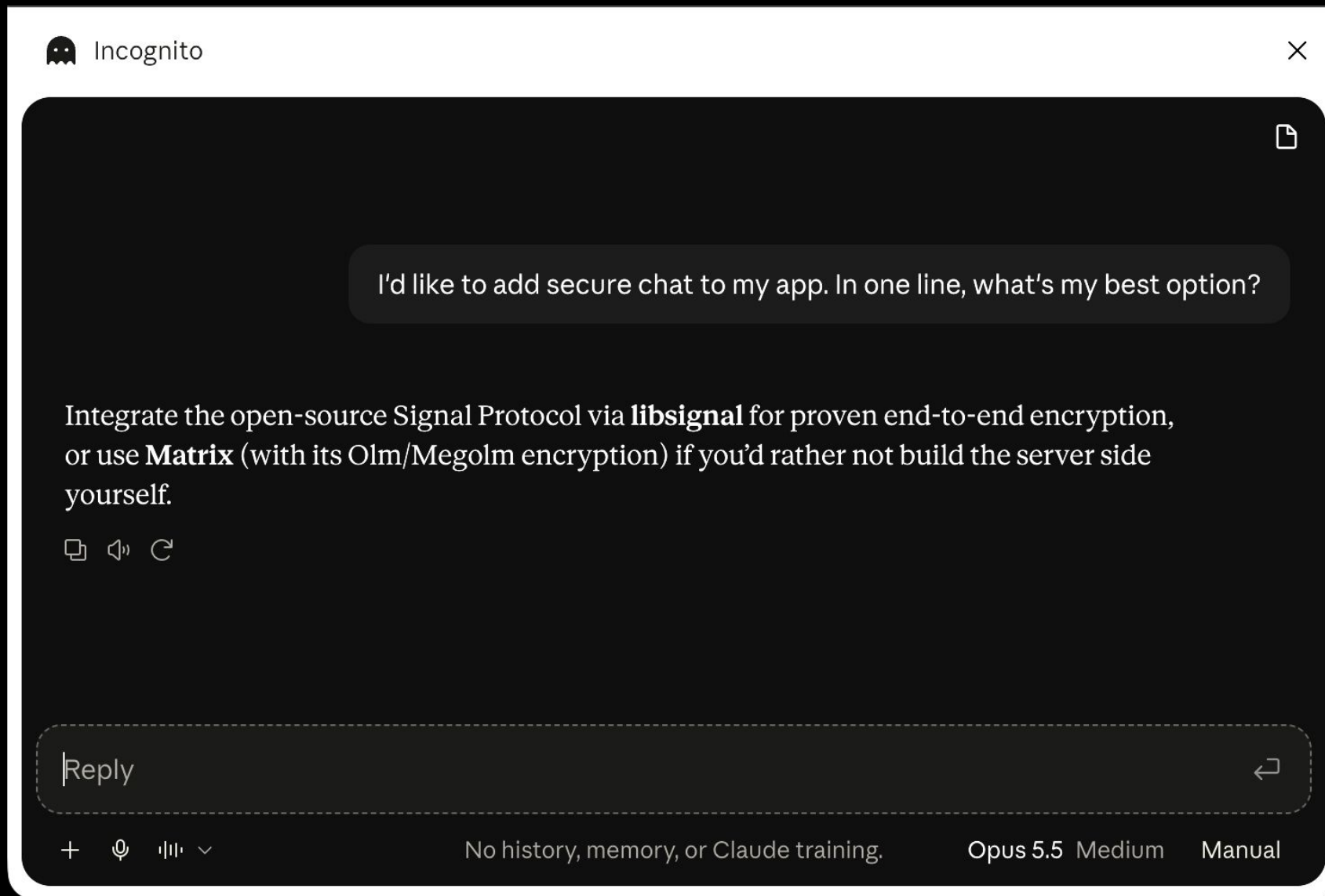
The Ukrainian UASAT LEO program involves the creation of an orbital constellation of 300 communication satellites. The first device is planned to be launched in October 2026, and then the system is planned to be scaled up in stages.



**We're seeing significant  
uptake of Matrix in:**

- 1. Public sector needing digital sovereignty**
- 2. FOSS communities wanting freedom**
- 3. Products which need embedded secure chat**
- 4. Consumer-facing chat apps**
- 5. Agentic communication**





# Meanwhile Matrix is the poster child for Digital Sovereignty, esp in Europe.

[matrix]

Constant flux of invites to participate in panels, workshops, policy feedback and analysis...



and more...

And as of today, breaking news: Amandine was appointed to the **Advisory Board of the Digital Commons EDIC!**



Digital Commons EDIC: a coalition of EU Member States pooling resources to develop, maintain, and scale the open-source tools Europe depends on.

# Funding Matrix's future

[matrix]

AUTOMATIC



element



GOUVERNEMENT

*Liberté  
Égalité  
Fraternité*

Direction  
interministérielle  
du numérique



meedio  
THE SAFE PLACE TO MEET



FlatTurtle



LIBERBYTE  
FREE YOUR DATA

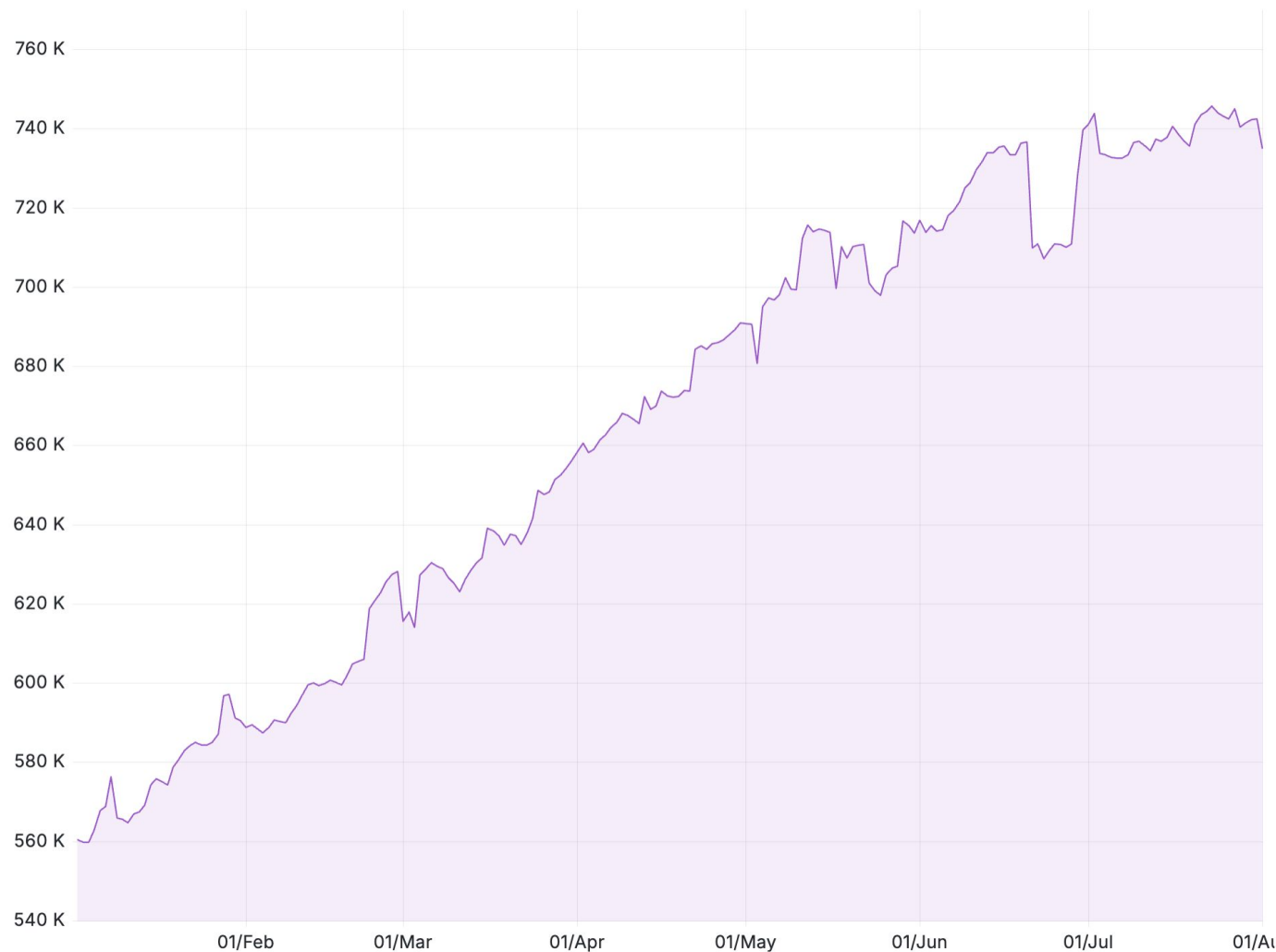


As of October 2026

**Where are we in the  
journey?**

# Retained 30-day users (Based on Synapse phone-home stats)

[matrix]



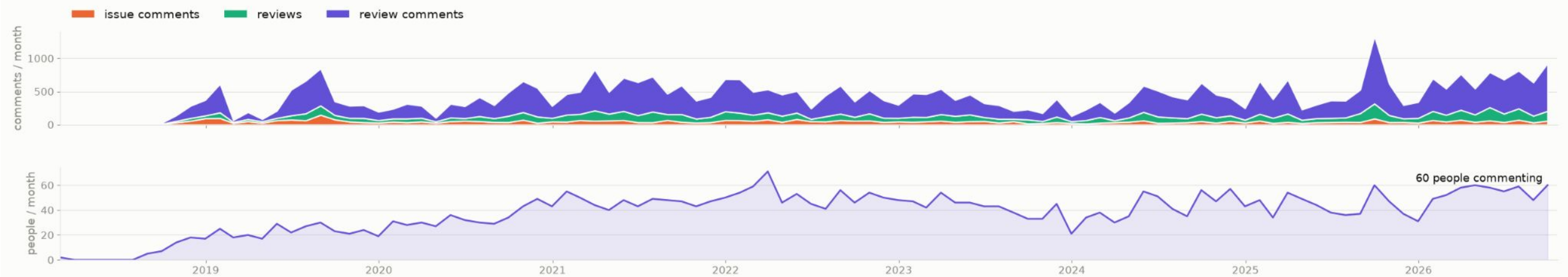
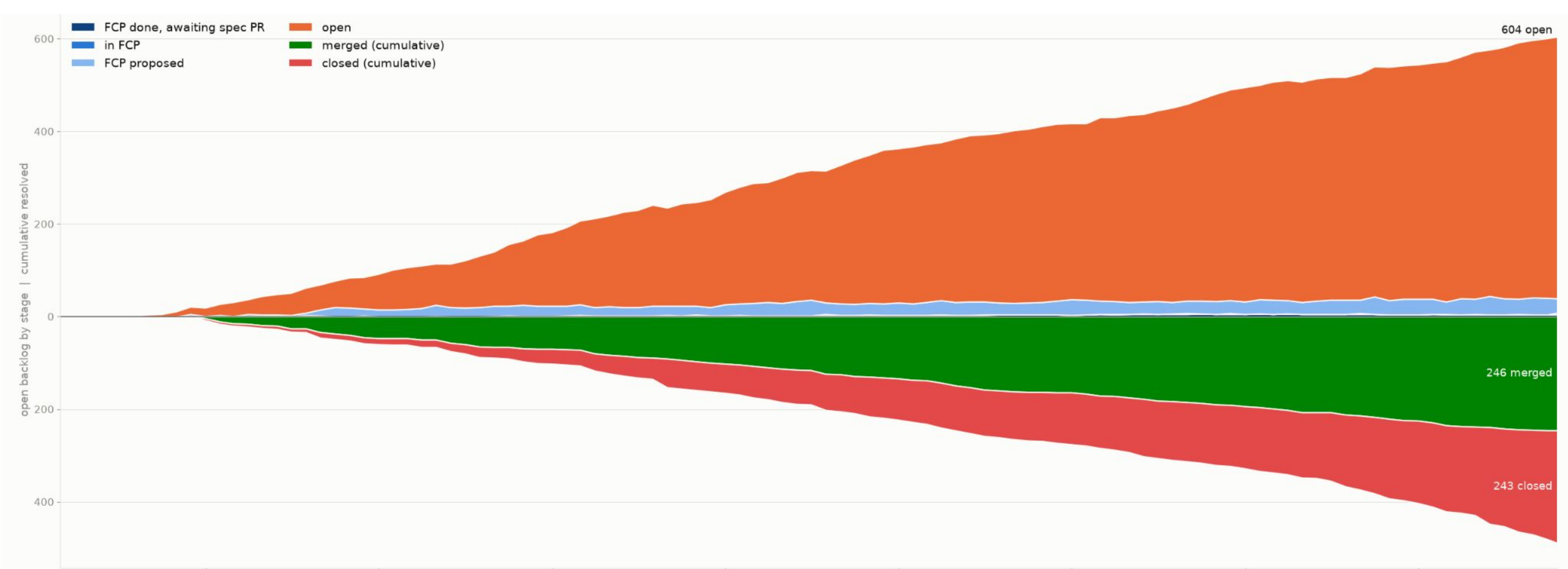
- R30 shows committed users, who have been active for over 30 days.
- Growing 50% YoY
- Stats are increasingly unreliable due to folks turning off reporting.
- (Figures include one 170K R30 deployment which is flickering on & off)

**More implementations than  
we can count!**



More implementations than  
we can count!

**Meanwhile, on the spec...**



# The Final Leg of Matrix 2.0

[matrix]

- Top tip: if you think that 4 unrelated massive projects constitute a version 2.0 of something, tell people that *after* they all land :D

|                                           | Final MSC | MSC proposed for FCP | MSC passed FCP | MSC merged in the spec |
|-------------------------------------------|-----------|----------------------|----------------|------------------------|
| <b>Next Gen Auth</b><br>MSC3861           | ✓         | ✓                    | ✓              | ✓                      |
| <b>Simplified Sliding Sync</b><br>MSC4186 | ✓         | ✓                    | ✓              |                        |
| <b>MatrixRTC</b><br>MSC4143               | ✓         | ✓                    | ✓              |                        |
| <b>Invisible Crypto</b><br>MSC4153        | ✓         | ✓                    | ✓              | ✓                      |

# Simplified Sliding Sync: SSS (MSC4186)

[matrix]

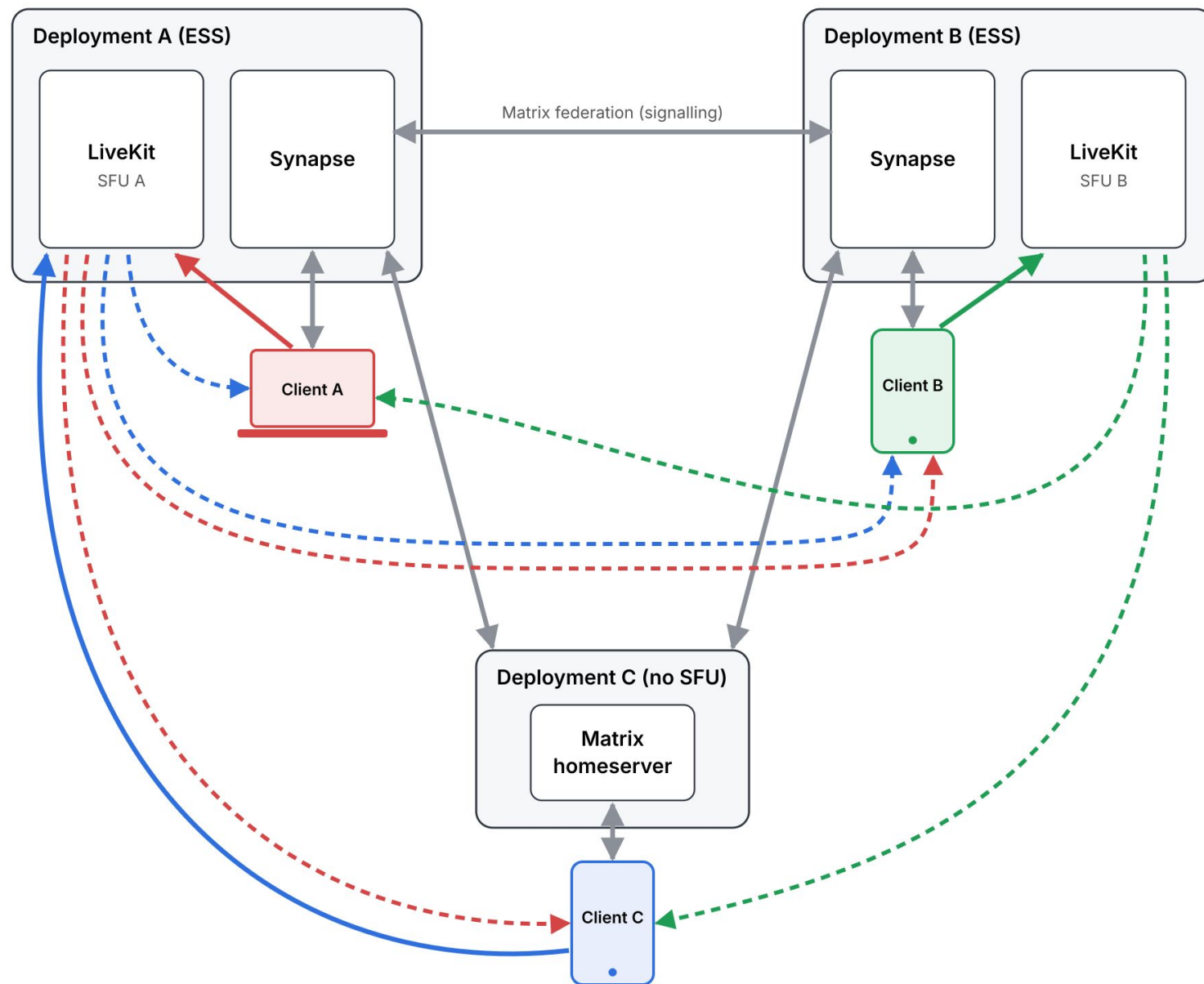
- Instant login, instant launch and instant sync.
- The main MSC4186 eventually passed FCP on June 29th.
- We then had to sort out and FCP all the extensions:
  - MSC4508: Sliding Sync Extensions: Common format and Typing
  - MSC4538: Sliding Sync Extensions: To-Device messages
  - MSC4543: Sliding Sync Extensions: End-to-end encryption
  - MSC4548: Sliding Sync Extensions: Account data
  - MSC4551: Sliding Sync Extensions: Receipts
- Just need to actually write the spec PRs now! 😄
- Meanwhile, could it be simplified further still?
- MSC4525 Paginated Sync is one possible next iteration...

# MatrixRTC (MSC4143)

[matrix]

- **The world's first multiparty E2EE federated VoIP protocol!**
- ...and it has passed FCP, and so is finalised and headed for the spec!
- This has been a **lot** of work:
  - Adding Sticky Events (MSC4534) as a whole new primitive turned out to be huge.
  - “What if we could guarantee delivery of events for up to an hour?”
  - All it took was 700 line MSC, 127 commits and 474 comments...  
31 PRs with 284 comments in Synapse implementations, not to mention clients.
- Delayed events, delegated delayed events, and delegated delayed sticky events ended up being fun too - but now give us proper call timeouts.
- Getting the real-life clients aligned with the final MSC has ended up being a full migration project too.

- A reminder of how it works:
  - Admins enable MatrixRTC in rooms by creating “slots” (`m.rtc.slot` state events)
  - Users then participate in these slots to exchange encrypted media by sending `m.rtc.member` sticky events
  - Each session has an application (e.g. `m.call` for voice/video calls - MSC4196)
  - And a transport for handling the media (e.g. `m.livekit` - MSC4195).
  - To signal the lifecycle of a session (e.g. notify or decline participation in a session), you send `m.rtc.invite` or `m.rtc.decline` events. (MSC4075)
    - MSC4075 hasn't made it into FCP yet, but you can still use MatrixRTC without it.
- This lets us decouple the mechanics of the underlying transport implementation from the process of negotiating the session.

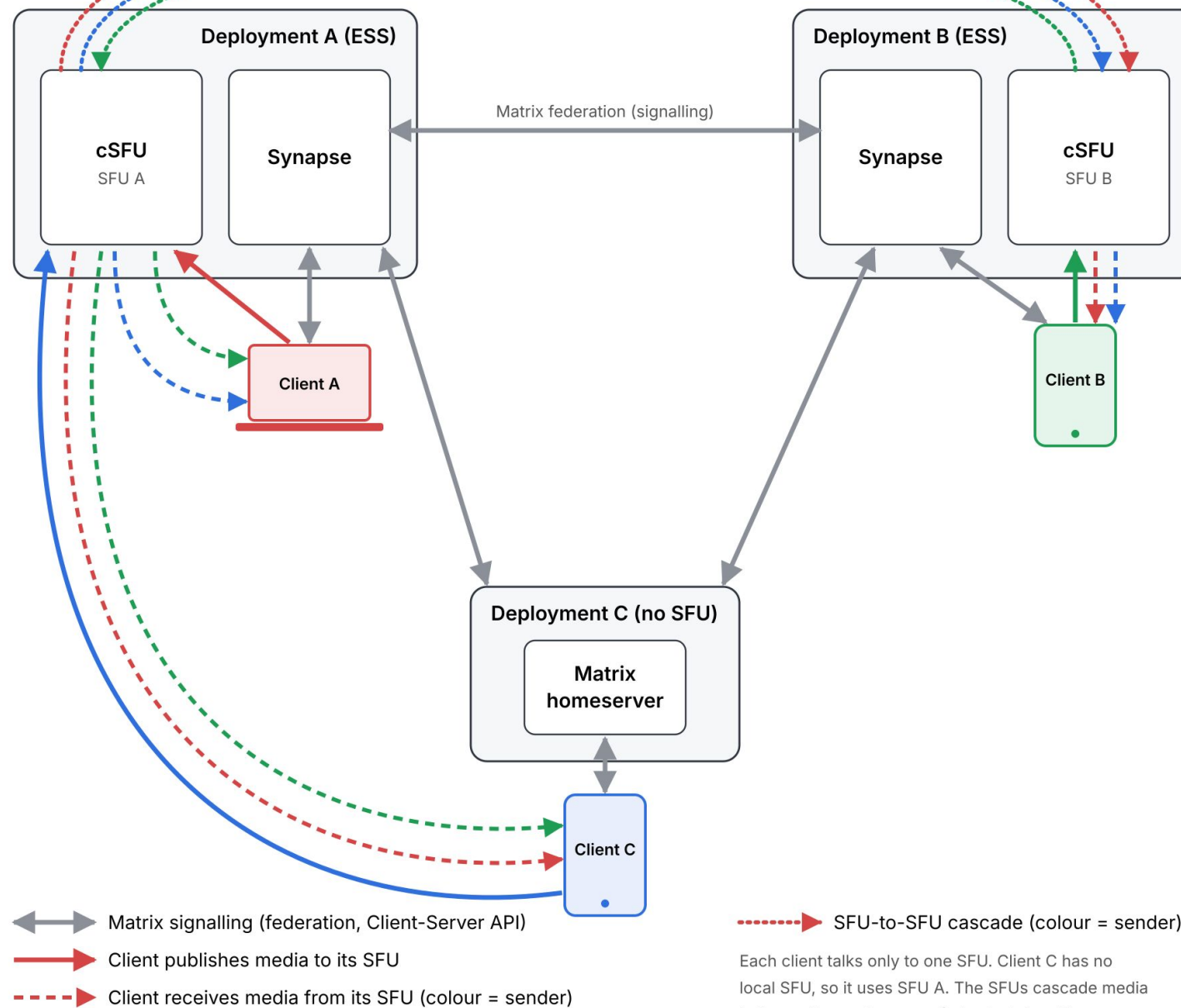


- Matrix signalling (federation, Client-Server API)
- Client publishes media to an SFU
- Client receives media from an SFU (colour = sender)

Client C has no local SFU, so it publishes to SFU A.  
Every client subscribes to every SFU carrying a participant it wants to receive.

**Demo!**

- So what about other transports?
  - LiveKit is fantastic and we love it, but it's not an open standard.
  - Can we do better than WebRTC + LiveKit?
  - Sneak preview: **Cascade**.
  - Written by Florian at Element
  - Built as a next generation Rust SFU which could define an open SFU protocol as an open MatrixRTC transport.
  - Clients publish to their local SFU. SFUs subscribe to each other.
  - Media cascades between the SFUs.
  - Streams are WebTransport + WebCodec rather than WebRTC
  - You subscribe to streams via hash of key & sender; no OpenID, no JWT etc.



**Demo!**

# Project Hydra

# Project Hydra

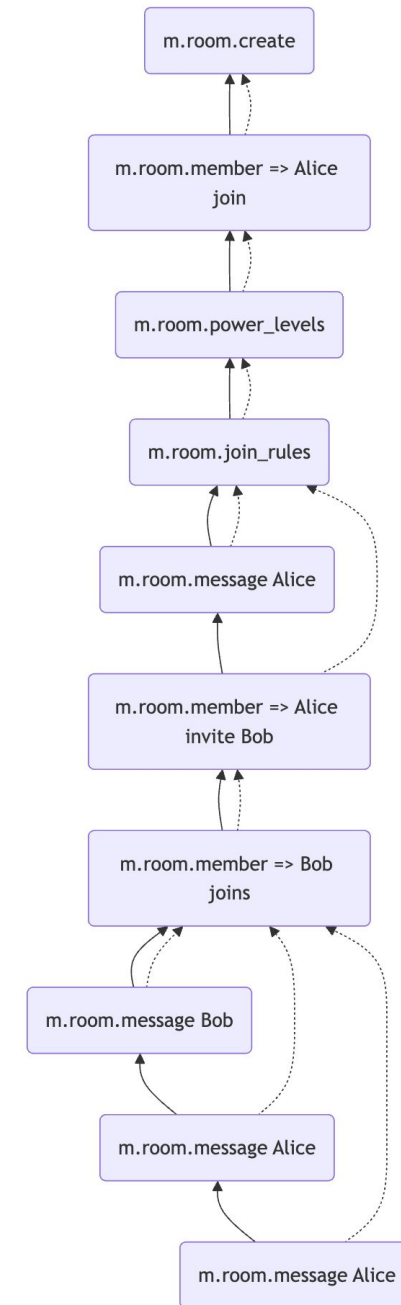
- Hydra is the project to fully solve state resolution issues in Matrix.
  - **Phase 1: Analysis & baseline (shipped in Room v12, Aug 2025)**
    - Make it explicit that room creators are privileged (MSC4289)
    - Make room IDs hashes of the create event, to stop attacks where different servers claim false m.room.create events (MSC4291)
    - State Resolution v2.1, which stops many classes of state resets (MSC4297)
  - **Phase 2: Additional robustness (shipping 2026)**
    - Switch to synchronising full state DAGs to improve robustness (MSC4242)
    - Switch to user IDs as account keys (MSC4430)
  - **Phase 3: Finality (~~late 2026~~ 2027)**
    - Add finality, to stop conflicts caused by backdating events
    - Updating State Resolution as needed (v3)

# Hydra Phase 2 - State DAGs

- State DAGs ensure that room state (key/value data) can never get out of sync between servers.
- Closes a whole class of potential state reset causes.
- Huge PR series landing in Synapse right now:

|                          |                                                                                                                                                                                                                                    |                                                                                           |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|
| <input type="checkbox"/> |  <b>MSC4242: State DAGs (inbound invites)</b> <span>A-Federation</span> <span>A-Invite</span> <span>A-State-DAGs</span>                           |  1     |
|                          | #20226 · by <a href="#">kegsay</a> opened on Sep 16 <span>Member</span> · 5/5 · 1 of 3 tasks · Review required · <span>✗</span> 43/47                                                                                              |                                                                                           |
| <input type="checkbox"/> |  <b>MSC4242: State DAGs (inbound events)</b> <span>A-Federation</span> <span>A-State-DAGs</span>                                                  |  17    |
|                          | #20221 · by <a href="#">kegsay</a> opened on Sep 11 <span>Member</span> · 4/5 · 2 of 3 tasks · Review required · <span>✗</span> 41/46                                                                                              |                                                                                           |
| <input type="checkbox"/> |  <b>MSC4242: State DAGs (inbound joins)</b> <span>A-Federated-Join</span> <span>A-Federation</span> <span>A-Join</span> <span>A-State-DAGs</span> |  152   |
|                          | #20194 · by <a href="#">kegsay</a> was merged 4 days ago <span>Member</span> · 3/5 · 3 tasks done · Approved · <span>✓</span> 47/47                                                                                                |                                                                                           |
| <input type="checkbox"/> |  <b>MSC4242: State DAGs (serving)</b> <span>A-State-DAGs</span>                                                                                   |  60    |
|                          | #20133 · by <a href="#">kegsay</a> was merged on Sep 11 <span>Member</span> · 2/5 · 3 tasks done · Approved · <span>✓</span> 47/47                                                                                                 |                                                                                           |
| <input type="checkbox"/> |  <b>MSC4242: State DAGs (fed client)</b> <span>A-Federation</span> <span>A-State-DAGs</span>                                                    |  11  |
|                          | #20127 · by <a href="#">kegsay</a> was merged on Sep 4 <span>Member</span> · 3 tasks done · Approved · <span>✓</span> 47/47                                                                                                        |                                                                                           |
| <input type="checkbox"/> |  <b>MSC4242: State DAGs (storage)</b> <span>A-State-DAGs</span>                                                                                 |  85  |
|                          | #19718 · by <a href="#">kegsay</a> was merged on Sep 2 <span>Member</span> · 1/5 · 3 tasks done · Approved · <span>✓</span> 47/47                                                                                                  |                                                                                           |
| <input type="checkbox"/> |  <b>MSC4242: State DAGs (Federation)</b> <span>A-Federation</span> <span>A-State-DAGs</span>                                                    |  70  |
|                          | #19425 · by <a href="#">kegsay</a> was closed on Apr 22 <span>Member</span> · 1 of 3 tasks · Review required · <span>✗</span> 36/41                                                                                                |                                                                                           |
| <input type="checkbox"/> |  <b>MSC4242: State DAGs (CSAPI)</b> <span>A-State-DAGs</span>                                                                                   |  229 |
|                          | #19424 · by <a href="#">kegsay</a> was merged on Apr 16 <span>Member</span> · 3 tasks done · Approved · <span>✓</span> 46/46                                                                                                       |                                                                                           |

[matrix]



# Hydra Phase 2 - Accounts as Keys.

[matrix]

- Servers currently sign events, meaning you have to check the server's signing key to verify them.
- This is slow and may be unreliable.
- Also, user IDs are not GDPR compliant as they contain undeletable direct personal information.
- We've now been through 8(!) possible ways to solve this.
- Current winner is: **MSC4428: Stable IDs for Room Members** + **MSC4430 Member Keys**
- MSC4428 explains that **mxids may change over time**, and defines a stable ID for them.
- MSC4430 defines per-user, per-room “member keys” stored on the server.
  - Servers sign events by the member key over federation, making them self-verifying!
  - Servers link these to their members in **m.room.member** events, signed by the server key.
- User IDs become that user's public `account_key` over federation and in the DAG:
  - `"sender": "@l8Hft5qXKn1vfHrg3p4-W8gELQVo8N13JklUMfmn2sQ"`
- The “stable\_id” mapping to the “account name” is then included in the CS API responses:
  - `"member_info": {  
 "@l8Hft5qXKn1vfHrg3p4-W8gELQVo8N13JklUMfmn2sQ": { "user_id": "@alice:example.com" }  
}`

# Hydra Phase 2 - Accounts as Keys.

[matrix]

- Look Ma - no domains!!

```
{
  "content": {
    "body": "hello world",
    "msgtype": "m.text"
  },
  "event_id": "$URgg8ykJZ75f0cCESIjsPTPzONGxpopoqUzny6ZgpmQ",
  "origin_server_ts": 1792349229125,
  "prev_state_events": [
    "$VKVUxIqgEiRA8WHJmr7s-6MtitoQeIgsUK-g5D45_uo"
  ],
  "sender": "@NWITftElHfx6bD0jnBbdp3twOZh5F4Xh1lm-sSRGacA",
  "type": "m.room.message",
  "unsigned": {
    "membership": "join",
    "org.matrix.msc4428.stable_id": "@NWITftElHfx6bD0jnBbdp3twOZh5F4Xh1lm-sSRGacA"
  },
  "room_id": "!9ypwRs7EpV0A_D02ideYdYJSQe3lhvj8KK6G1-cByjc"
}
```

**Demo!**

# Hydra Phase 3 - Finality and ERA

[matrix]

- The final piece of the puzzle is Finality - stopping servers backdating events.
- This is the final possible cause of state resets in Matrix.
- Novel academic solution: Epoch-Resolved Arbitration:  
<https://dl.acm.org/doi/10.1145/3806077.3806691>
  - "[Kegan was right](#)" 😊 – Martin Kleppman
- Published and presented at PaPoC 2026.
- Spoke about this a lot at the Matrix Community Summit.
- The idea is that rooms can optionally have a “finality server” (probably the most senior admin’s server) which sends out “epoch events” which finalise the DAG up to that point.
- This lets us curate the level of decentralisation: the rate at which the finality server seals the room history defines how out of sync it can get.
- Up next after phase 2!

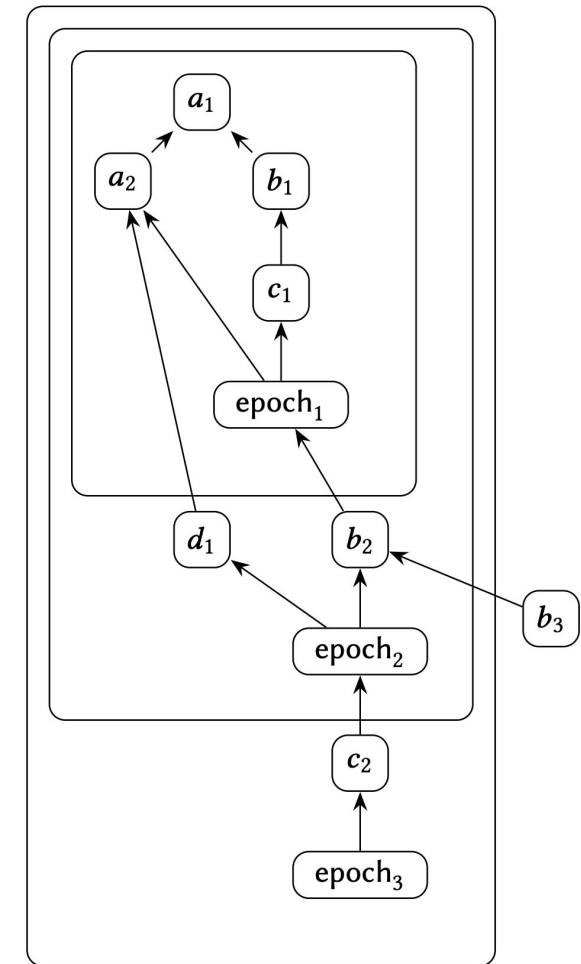


**Figure 6.** Epochs define a *closed past* [7] to prevent backdating. Event  $b_3$  is in a *pending epoch*, despite it pointing to an earlier event in a previous epoch.

# Finality isn't just about backdating.

- Can speed up implementations enormously
  - No need to store historical state snapshots
  - No need to conduct state resolution over finalised DAGs
- Provides transcript consistency as a cryptographic property (at last!)
- Supports the full spectrum of Matrix deployment environments from P2P to Interplanetary to VS-NfD without fragmenting the network or breaking interop.
- Lets clients delete any historical E2EE key material associated with finalised history (if needed)
- ...or you could delete the finalised history itself from the server, for privacy.
- **Unlocks vanilla MLS** if needed, by enabling selective centralisation.

# MLS and Encryption

# Where does MLS fit in?

- Messaging Layer Security (RFC9420) is an IETF standard for group E2EE.
- We don't want Matrix to be hard-locked to *any* given E2EE system.
  - Some environments don't need E2EE at all (e.g. trading off low-bandwidth > privacy)
  - Olm/Megolm may not be perfect, but nowadays it works.
  - MLS is still relatively new, and is also not perfect (e.g. no deniability; the need for Partial MLS)
- MLS benefits:
  - Many eyeballs thanks to IETF process.
  - Provides a solution to cryptographic group membership (but so does Hydra)
  - Better Post Compromise Security (if a sender starts a new session, all senders update)
  - Better performance in best-case conditions, but we're not seeing it in practice.
  - Beloved by the BSI.

# Where does MLS fit in?

- MLS drawbacks:
  - Vanilla MLS needs group changes to be centralised for a given room, creating a SPOF.
  - Particularly problematic for larger or unreliable (e.g. P2P) networks.
  - MLS expects to drive the group membership from that centralised point, missing the whole point of Matrix's decentralised access control & state resolution.
  - Large key packages (somewhat addressed by Partial MLS)
  - No cryptographic deniability (unlike Signal or Olm/Megolm)
- Stuff that can and should be improved irrespective of MLS:
  - Post Quantum encryption
  - Minimising metadata, e.g.
    - Accounts-as-keys
    - Sealed sender, Zero-knowledge group membership
  - Encrypting room state

# Where does MLS fit in?

- Spec Core Team's current approach:
  - Support MLS efforts, but do not hardcode Matrix to MLS (let alone centralised MLS).
  - Ensure state resolution is indestructible, for the benefit of all Matrix; today & tomorrow.
  - Having finally got state resolution in a good place, build on it and make the most of it.
  - Hydra Finality effectively gives a means of **variable decentralisation**.
  - At one extreme, the room can be fully centralised on the epoch server, and **vanilla RFC 9420 MLS can be used for encryption** - e.g. for VS-NfD.
  - At the other extreme, the room is fully decentralised, allowing infinite netsplits (and backdating), requiring Decentralized MLS - e.g. for a relatively trusted but highly disconnected federation (e.g. emergency responders)
  - ...Or you can go in-between, surviving netsplits up to N minutes/hours to taste.
- Improve metadata irrespective of MLS (e.g. accounts-as-keys).
- Improve cryptographic group membership irrespective of MLS (e.g. clientside state res)

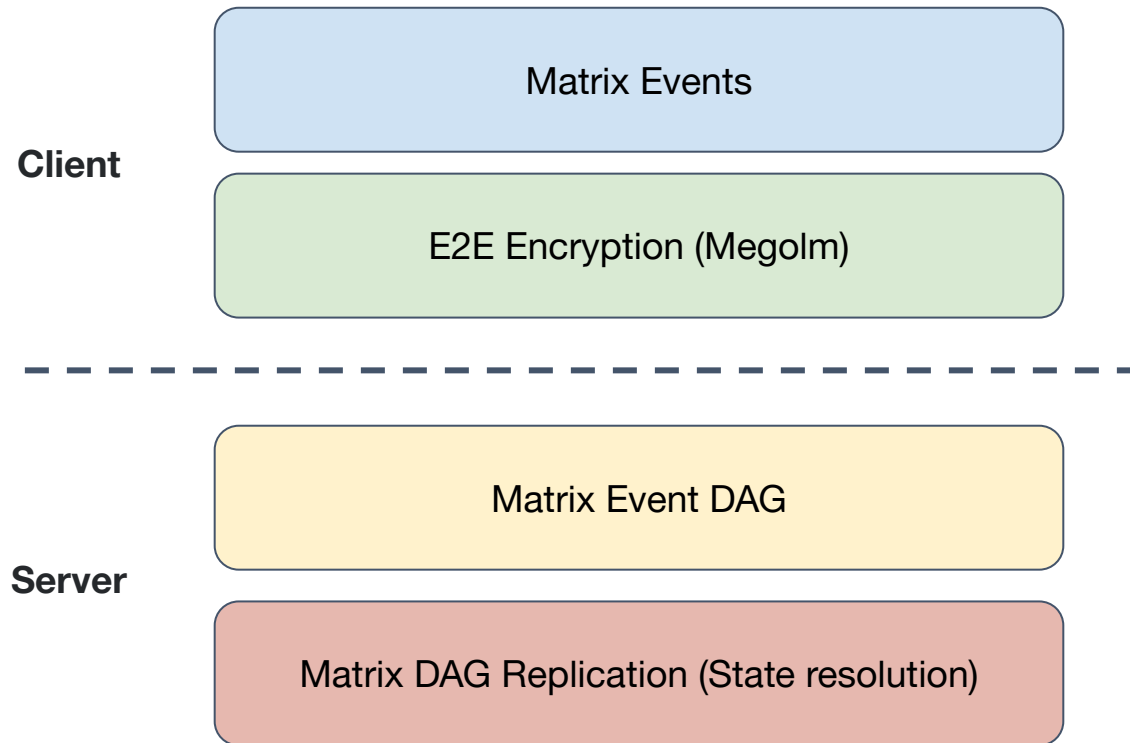
# Client-controlled group membership.

[matrix]

- Probably the biggest defect of Matrix today is the fact that the server unilaterally controls who participates in its conversation.
- Instead, the only people who should be able to add new users should be users within those rooms who have permission to do so.
- By running Matrix state resolution in the server in the client you can let the client prove that they are allowed to do certain operations.
- **What if we could make Matrix entirely metadata-protecting while also solving the client-controlled group membership problem?**

# Today's Matrix

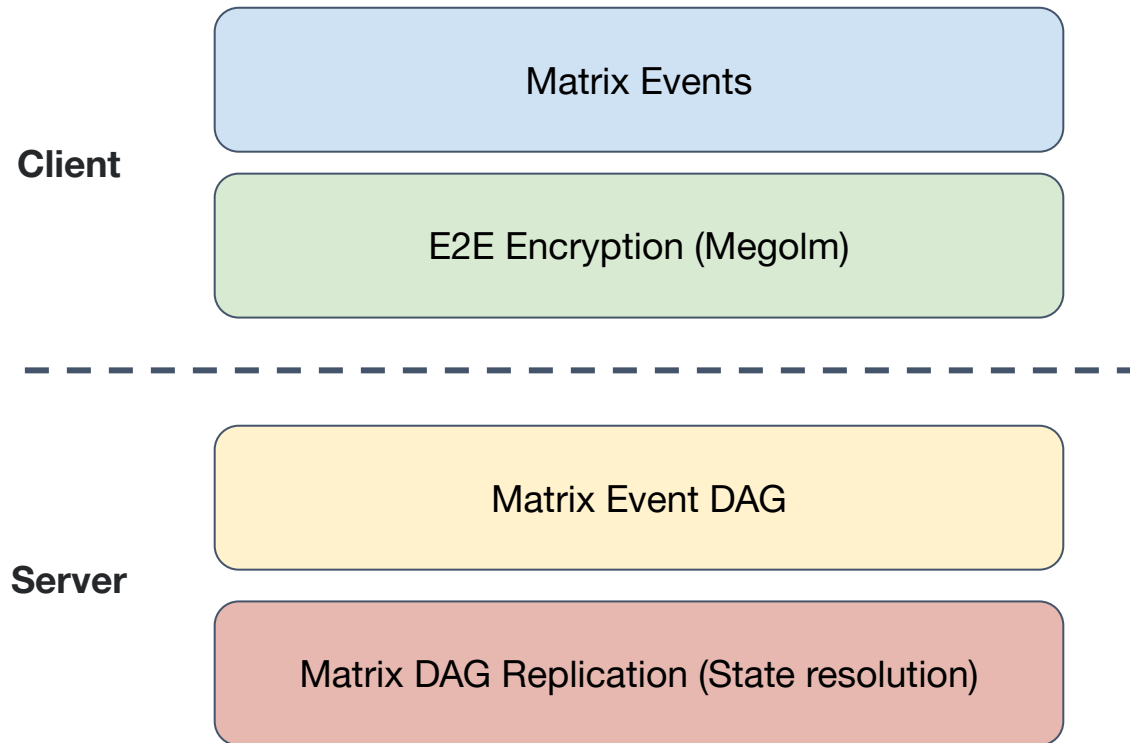
[matrix]



# Today's Matrix

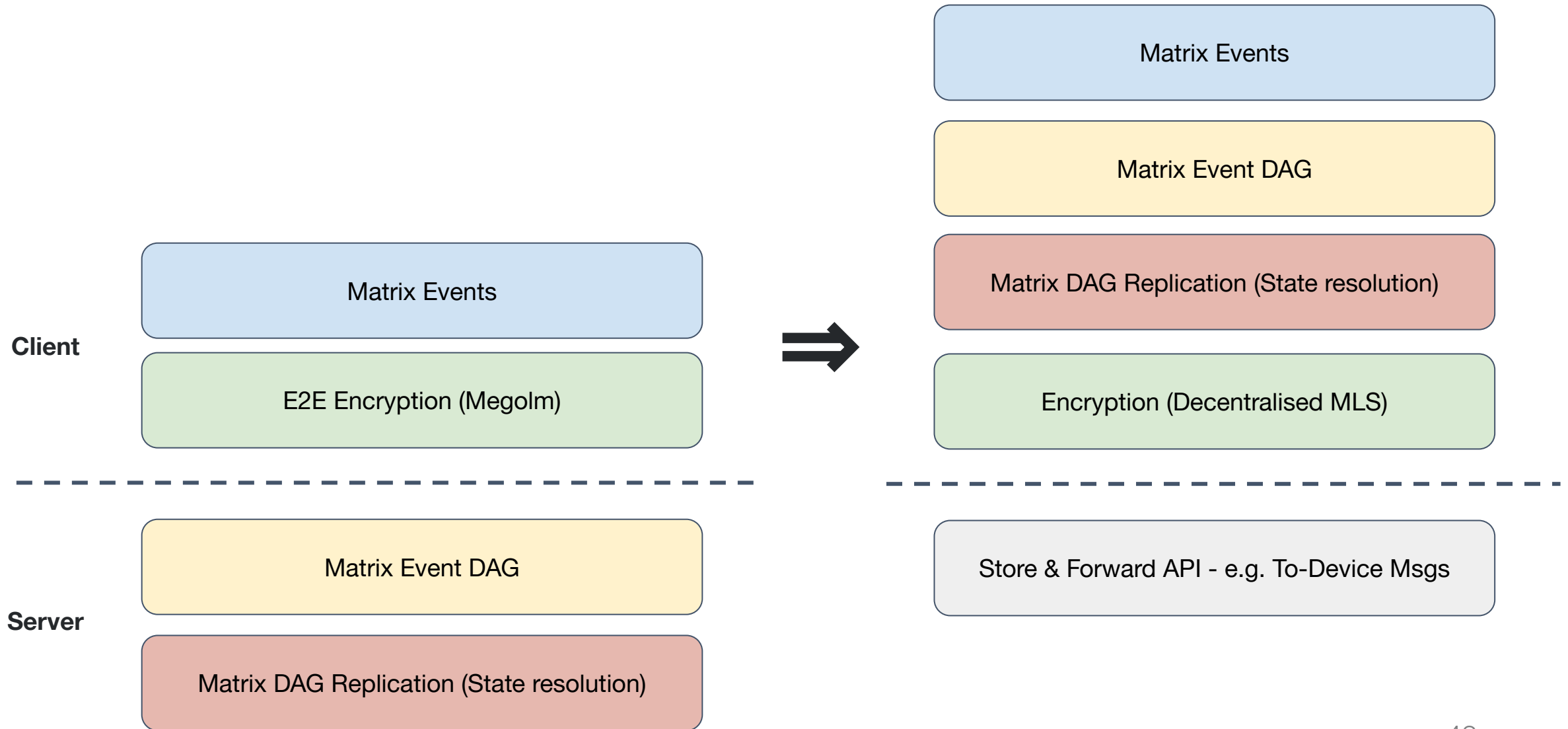
[matrix]

## What if...



# Client-side State Res with DMLS

[matrix]



# Tachyon

[matrix]

- Tachyon is an experiment to demonstrate Matrix tunnelled over Decentralised MLS.
- Decentralised MLS synchronises keys for E2EE across a set of devices without a central control point
- Tachyon extends matrix-rust-sdk to speak DMLS.
- **Not** a clientside homeserver.
- Sends and receives DMLS packets over normal Matrix to-device messages
- Inside the packets is the normal Matrix DAG, controlled by the client - not the server.
- Data is stored in the matrix-rust-sdk event cache.
- The Matrix DAG defines which users are in the room via normal decentralised Matrix access control.
- DMLS then restricts which devices participate in the room, driven by the Matrix DAG within the DMLS payloads.
- **As a result, you get fully metadata-protected Matrix conversations which look like normal Matrix rooms, but the server doesn't even know they exist. They're just a cloud of DMLS.**
- This approach could be used on top of **any** store and forward mechanism: SMTP, XMPP, ActivityPub, AT Proto private stores, etc.

# Tachyon Demo

<https://aurora-tachyon.netlify.app>

# What's next with Tachyon?

- What if we did the same technique with Olm/Megolm? => Photon (see my talk at 16:45 in Sofya!)
- What if we just did the client-side state res bit of it? => Lepton (also in the later talk!)
- Meanwhile on Tachyon:
- The current code is not secure or audited, and **should not be used** for anything serious yet.
- Try to tame the code into something publishable (currently a 120KLOC PR against matrix-rust-sdk)
- Figure out how to encrypt (or not) invite handshakes.
- Get it secure.
- Get it resilient to DoS.
- Consider structuring it as a Pantalaimon-style daemon so any Matrix client can use it as a way to get metadata-protected traffic (rather than requiring clients to use rust-sdk)?
- Consider doing a demo of layering it over another protocol to illustrate how it could be used to unleash metadata-protected E2EE messaging for other protocols?

# Meanwhile, business-as-usual Encryption... [matrix]

- Sharing historical keys on invite landed!
- E2EE Search has landed in matrix-rust-sdk!
- Many verification improvements to unlock excluding unverified devices.
- Recovery key management improvements underway currently
- Device dehydration underway, aiming for end of 2026.
- PQXDH is yet to resume - still looking for committed funding.

# **P2P Matrix is back!**

**come see it at 16:45 in Sofya**

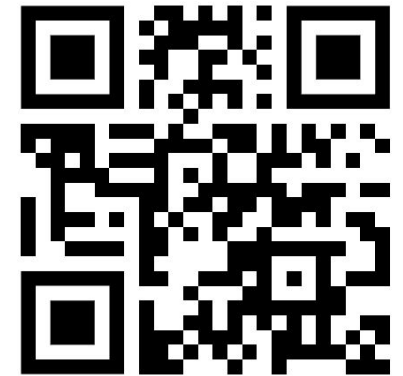
**Spoiler: we wrote a  
P2P-specific Rust homeserver  
for called Neutrino and  
plugged it into Element X**

# What's next?

- Actually bump the spec to 2.0!
- Ship Hydra Phase 2
- Finish Hydra Phase 3
- Fix client-controlled cryptographic group membership.
  - By whatever combination of MLS, DMLS, or client-side state resolution.
- Improve metadata posture
  - Member keys, encrypted state events, Tachyon...
- Keep shipping P2P Matrix
- Lots more Trust & Safety - come see Jim's talk at midday in Peter!

# 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](mailto:@matthew:matrix.org)



[matthew@matrix.org](mailto:matthew@matrix.org)



<https://matrix.org>



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



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

## In practice, this means:

- Finish Hydra.
- Fix client-controlled cryptographic group membership.
  - Either by MLS, DMLS, or client-side state resolution.
- Improve metadata posture
  - Pseudo IDs, encrypted state events, maybe leveraging MLS
- Improving Trust & Safety
- Ensuring that ecosystem-driven features like custom profiles, custom emoji, etc actually land in the spec.
- Figuring out how Matrix can be used by other ecosystems more effectively, to try to avoid ATproto/ActivityPub/Nostr/Beehive etc reinventing Matrix.

# Trust & Safety

- Reporting over federation
- Reporting to community safety teams
- Appeals
- Improved clarity on reasons for enforcement
- Shared language of harms
- Resources for users
- Transparency reporting
- Reporting the content of encrypted events

## Longer term...

- **P2P Matrix is not dead.**
- In fact, Element is funded to resume work on it as of... yesterday.
- P2P solves *many* problems:
  - Keeps metadata entirely on the clients.
  - Provides resilience against blocking/surveillance on servers.
  - Provides connectivity even if the wider network is unavailable.
  - Mandates very robust and resilient federation (hello Hydra!)
- However, it's still an open area of research:
  - Resilient global P2P overlay networks are still a Hard Problem.
  - Naively, you have to sync all possible data to the client, chewing bandwidth and battery. Unless you have smart relays, which then start to sound like servers.

# Matrix 3.0 ideas?

- Options are still up in the air:
  - Focus on Trust & Safety?
  - Focus on P2P and metadata protection?
  - Focus on MLS and metadata protection?
  - Focus on “Portable Matrix” targeting other platforms?
  - Just keep on 2.x and only nudge the major version if we get really bored?
- Suggestions welcome!