



MatrixRTC in 2026

Bringing MatrixRTC to the Matrix ecosystem

Robin T. · Timo K. · Element · The Matrix Conference, Malmö · October 2026



MatrixRTC "The Vision"



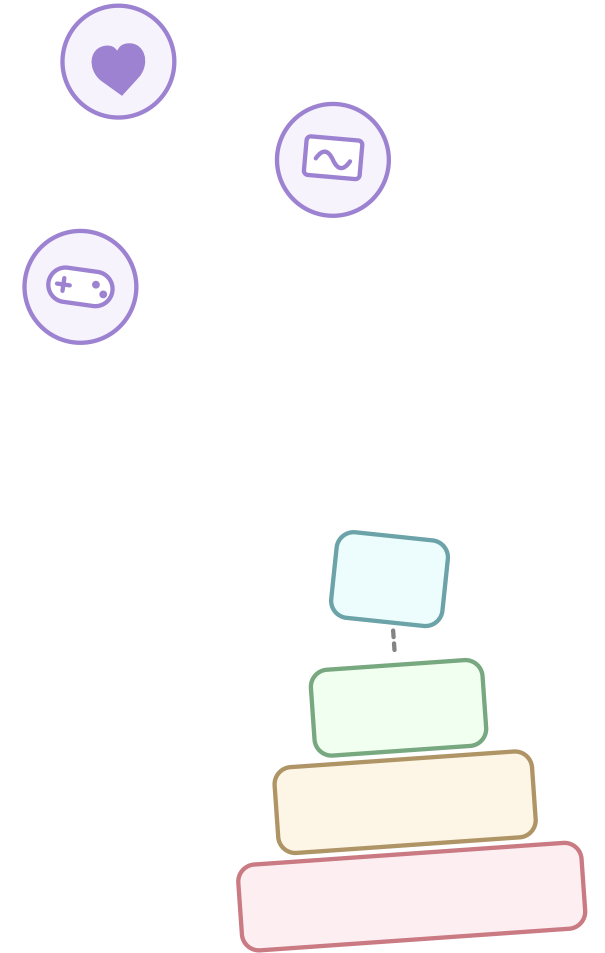
The best possible RTC solution based on Matrix



MatrixRTC "The Vision"

technical aspects

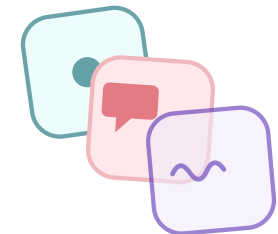
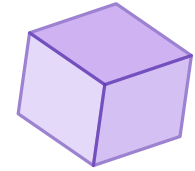
- A new base building block for Matrix
- Leverage what we have, don't re-introduce existing concepts
- Built to last and grow (no hardcoded transport)
- More than calls



MatrixRTC "The Vision"

ecosystem and adoption

- Good documentation and SDKs to build on top of
- Have it adopted beyond chat
- Tools and extensions (recording, transcription, ...)
 - Don't miss Matthew's talk, or check out the recording!
- Great apps/implementations!



MatrixRTC "The Vision"

where we are

PROJECTS

Clients



Element X
iOS · Android



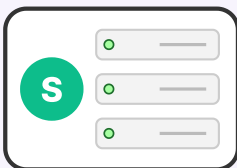
Cinny
web



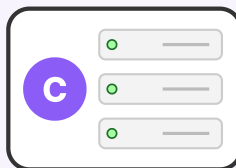
Tammy
Connect2X



Servers



Synapse



Continuuity



sticky events + delayed events

Non-VoIP apps

MatrixRTC beyond calls



Neoboard



FlappyRoyal



FloorIt!psum



whiteboards · games · 2025 hackathon

BUILT ON

SDKs



matrix-js-sdk
Element Web · Element Call



trixnity
Tammy

1 • History of VoIP in Matrix



2014 to present



2014: Peer-to-peer WebRTC calls

- ✓ End-to-end encrypted
- ✓ Decentralised
- ✓ Open specification
- ✗ ~~Group calls~~
- ⚠ Requires best-effort search for events

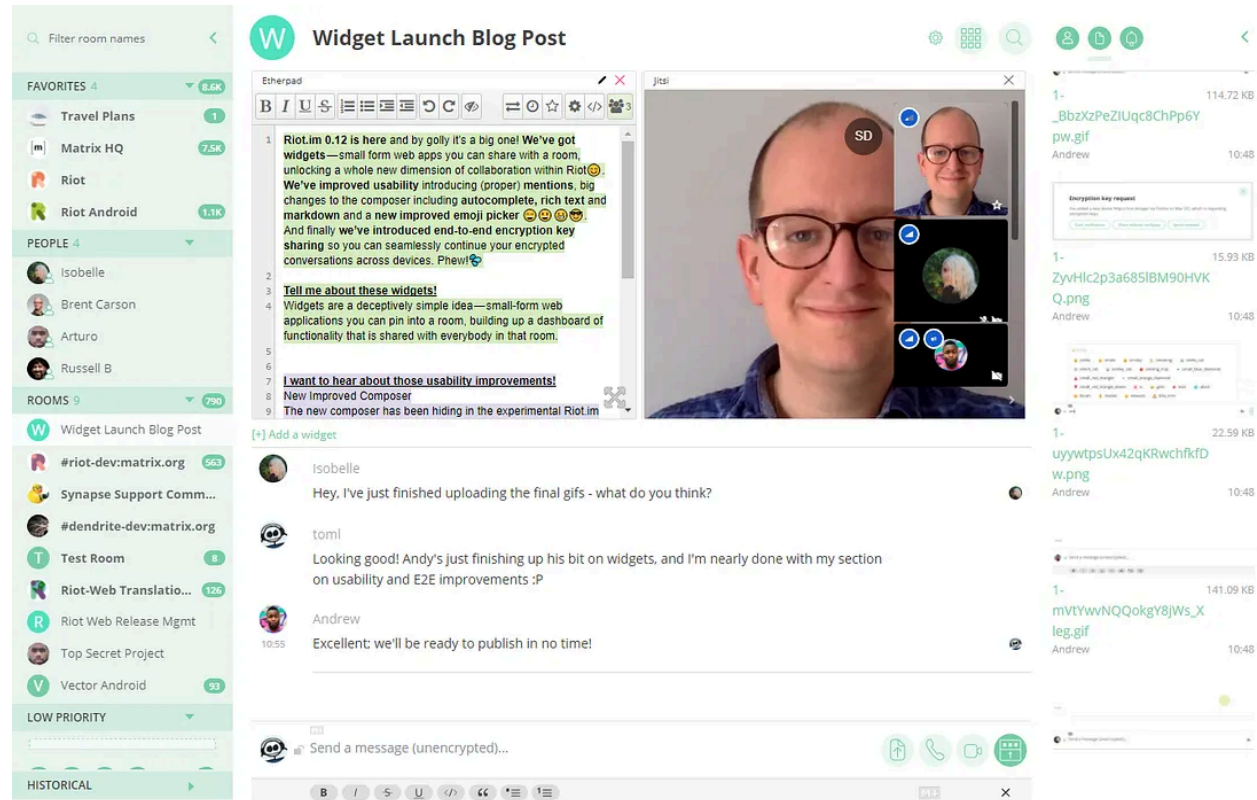
```
+-----+           +-----+
| Caller |           | Callee |
+-----+           +-----+

|
(Places Call)
|----- m.call.invite ----->|
|----- m.call.candidate ----->|
|----- [..candidates..] ----->|
|
|                                     (Answers call)
|<----- m.call.answer -----|
|--- m.call.select_answer --->|
|
| . (Call is active and ongoing) .
|
|                                     (Ends call)
|<----- m.call.hangup -----|
```

<https://spec.matrix.org/latest/client-server-api/#voice-over-ip>

2017: Jitsi widgets

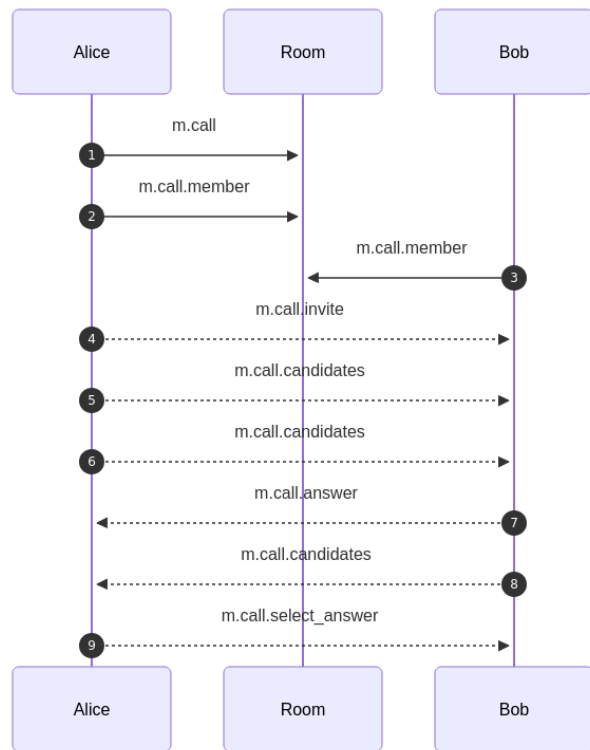
- ✗ ~~End to end encrypted~~
- ✗ ~~Decentralised~~
- ✗ ~~Open specification~~
- ✓ Group calls
- ⚠ Call metadata not visible from Matrix



2021: Peer-to-peer WebRTC group calls

- ✓ End-to-end encrypted
- ✓ Decentralised
- Open specification (proposed; abandoned)
- ✓ Group calls
- ⚠ Performs poorly in large groups

MSC3401

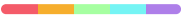


2024: MatrixRTC



- ✓ End-to-end encrypted
- ✓ Decentralised
- ✓ Open specification (proposed; coming soon in Matrix 2.0!)
- ✓ Group calls

2024: MatrixRTC



- ✓ End-to-end encrypted
- ✓ Decentralised
- ✓ Open specification (proposed; coming soon in Matrix 2.0!)
- ✓ Group calls
- ✧ Arbitrary data
- ✧ Arbitrary transports
- ✧ Metadata visible from Matrix

2 · How MatrixRTC works

An extensible protocol for real-time experiences



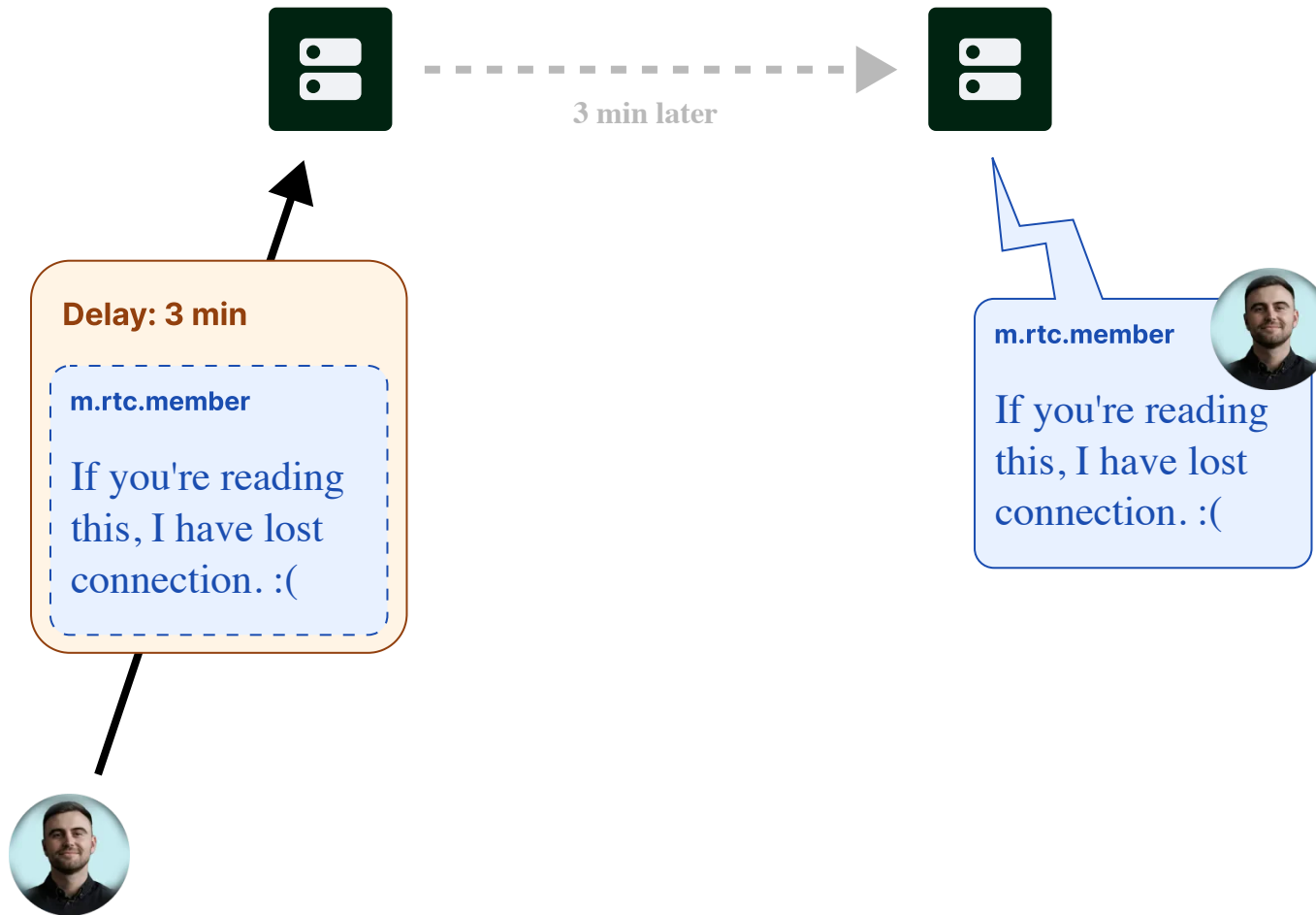
The specification

Coming soon in Matrix 2.0!

Proposal	Topic	Status (19 Oct 2026)
MSC4140	Delayed events	Merged
MSC4354	Sticky events	Merged
MSC4143	MatrixRTC core	Merged
MSC4195	LiveKit transport	Merged
MSC4196	Voice & video calling application	Merged
MSC4480	Sliding sync ext.: sticky events	FCP proposed · 1/8
MSC4075	Invites & notifications	In review · needs impl.
MSC4519	Transports registry	Out of 2.0 scope

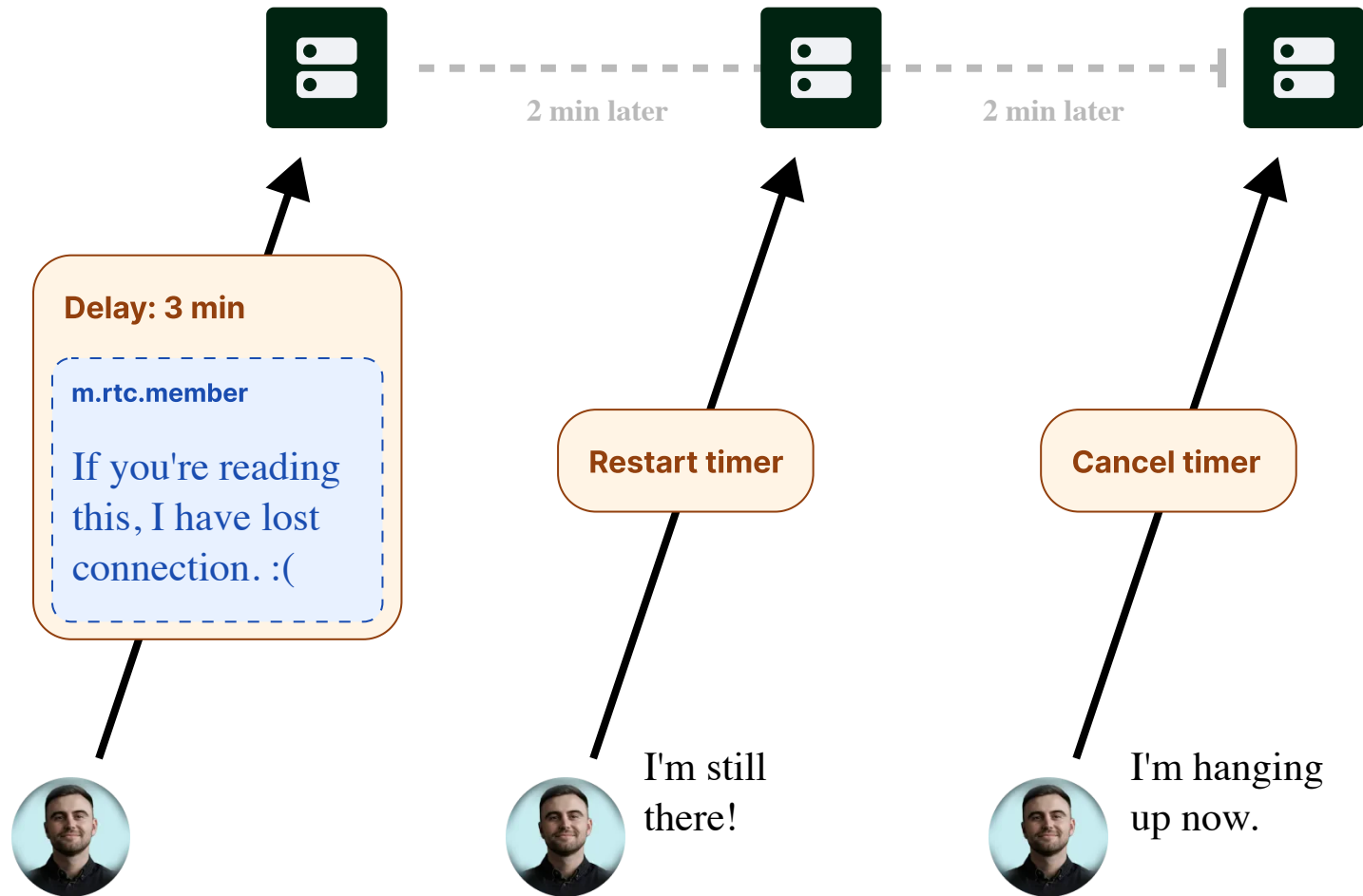
MSC4140: Delayed events

Schedule an event for later, cancel if needed



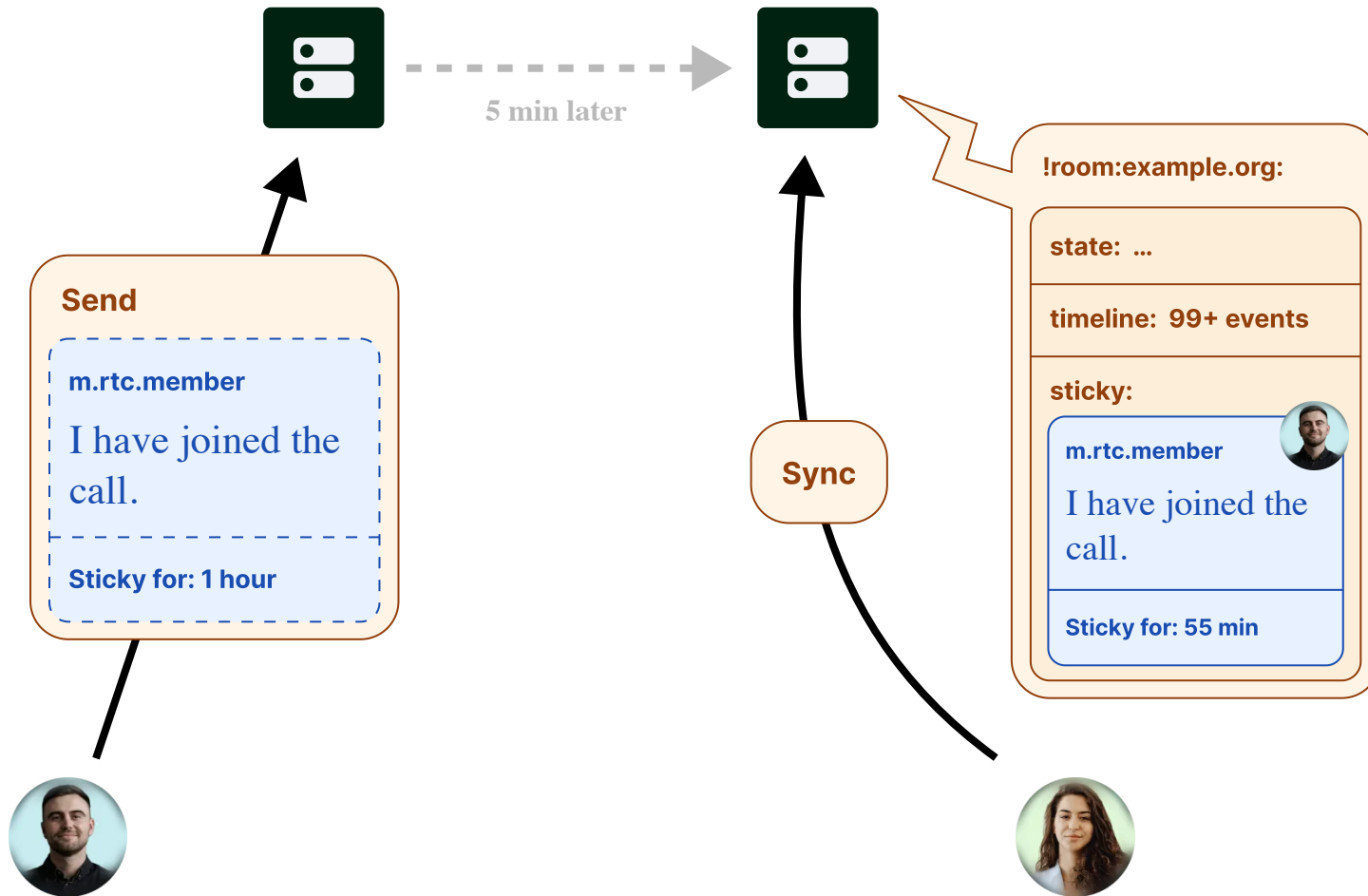
MSC4140: Delayed events

Schedule an event for later, cancel if needed



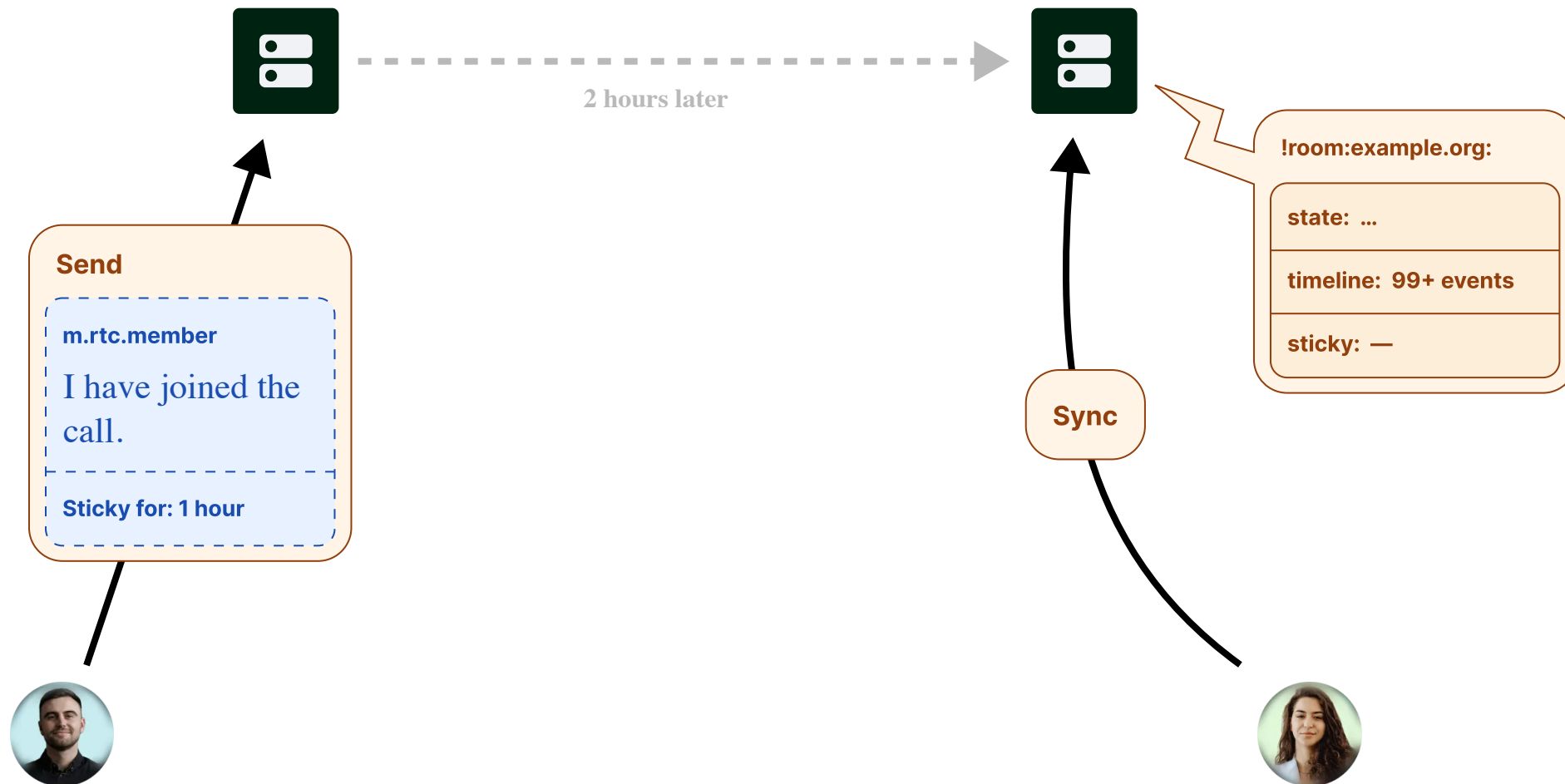
MSC4354: Sticky events

Room events that are eagerly pushed to clients for a limited duration



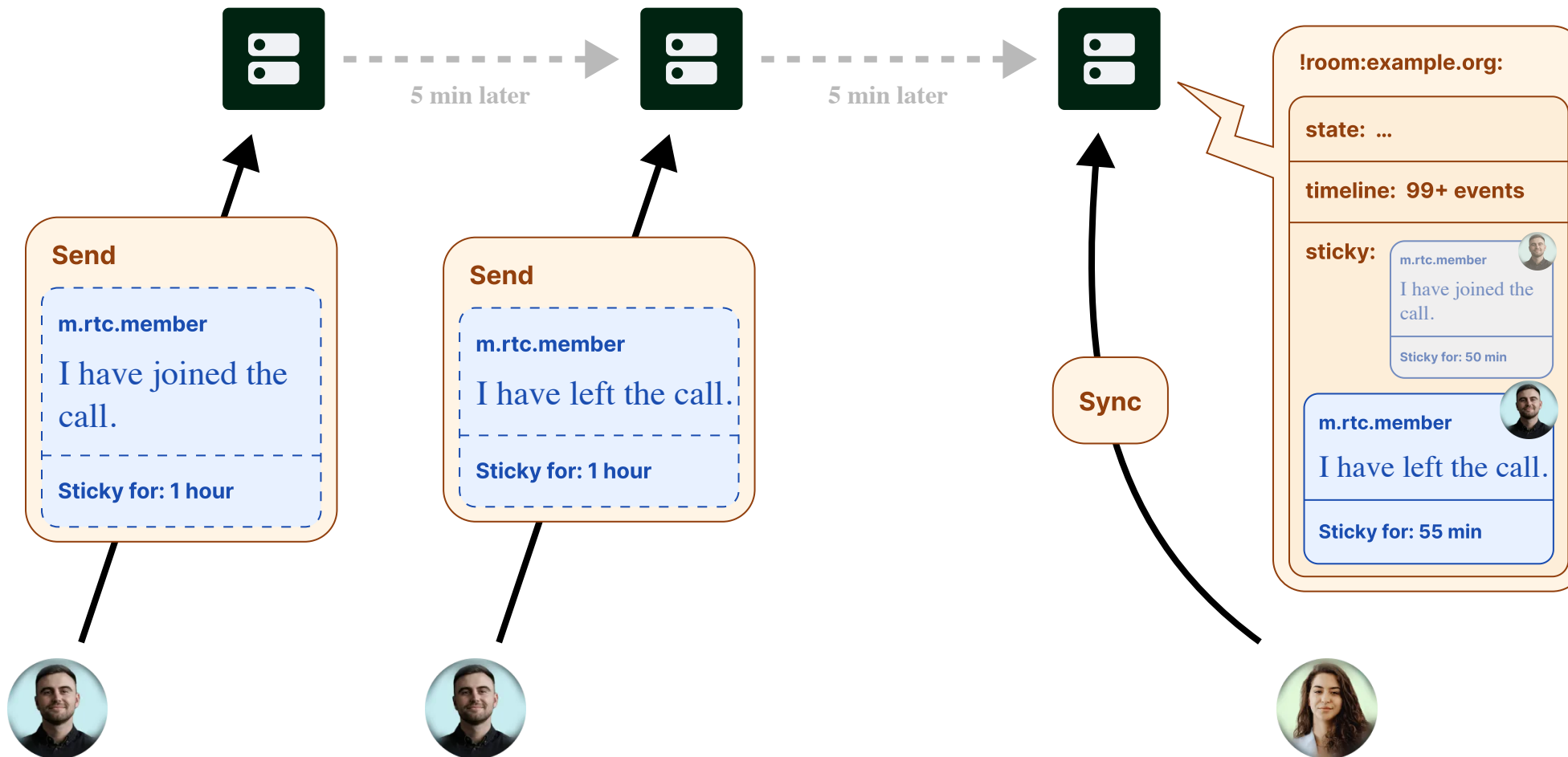
MSC4354: Sticky events

Room events that are eagerly pushed to clients for a limited duration



MSC4354: Sticky events

Room events that are eagerly pushed to clients for a limited duration



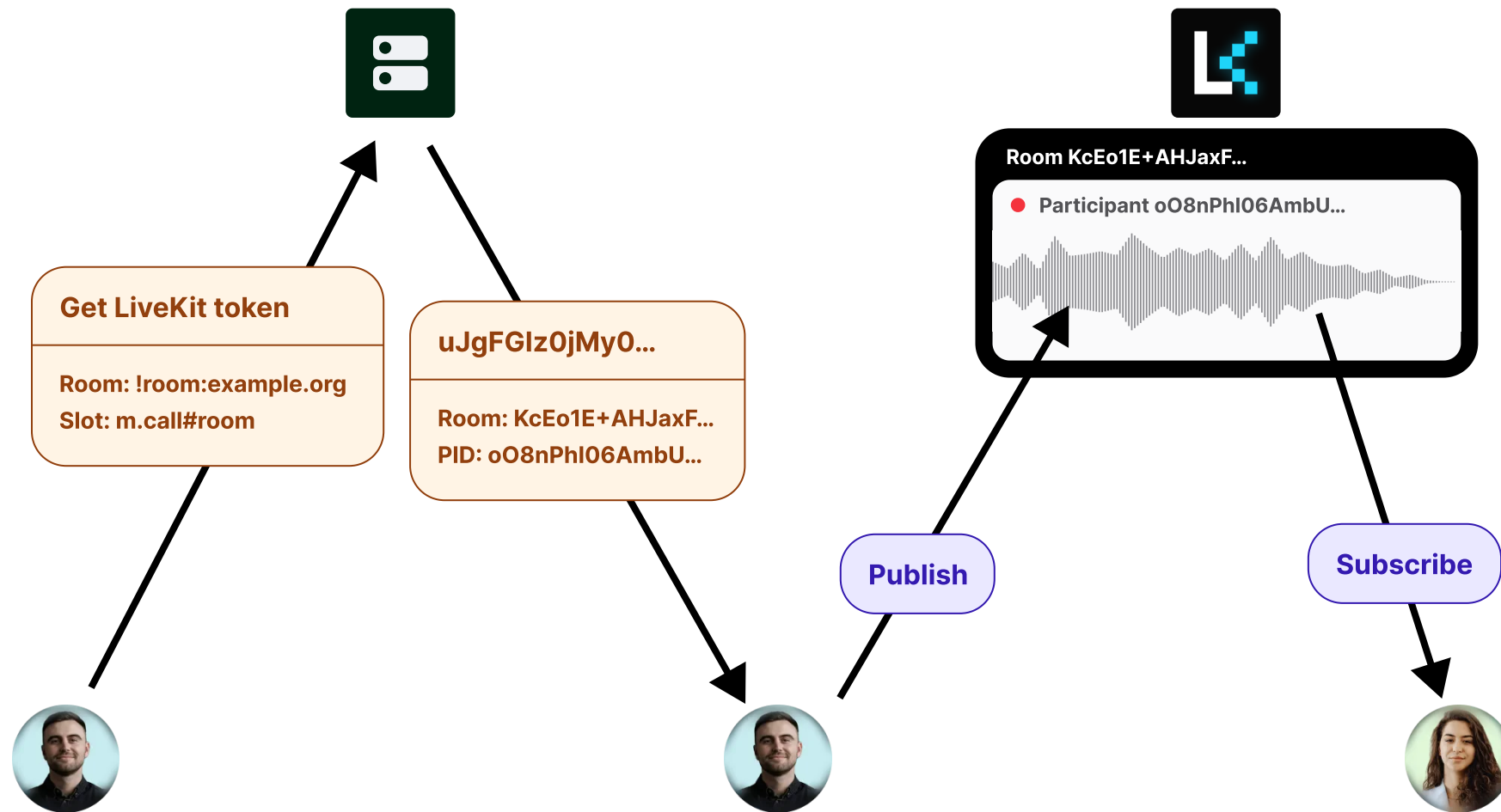
MSC4143: MatrixRTC

Configure a slot, send a sticky membership event to join



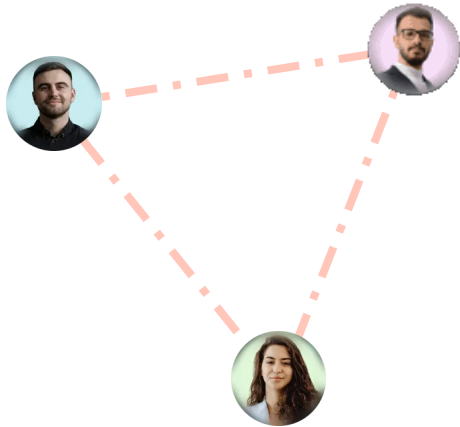
MSC4195: LiveKit transport

Publish media to your homeserver's LiveKit SFU

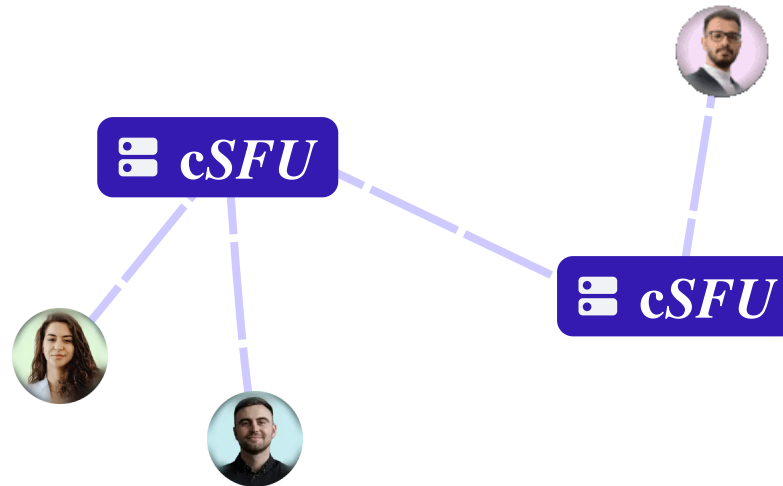


Other transports?

Peer-to-peer
for small groups



Cascading SFUs
for optimal media distribution



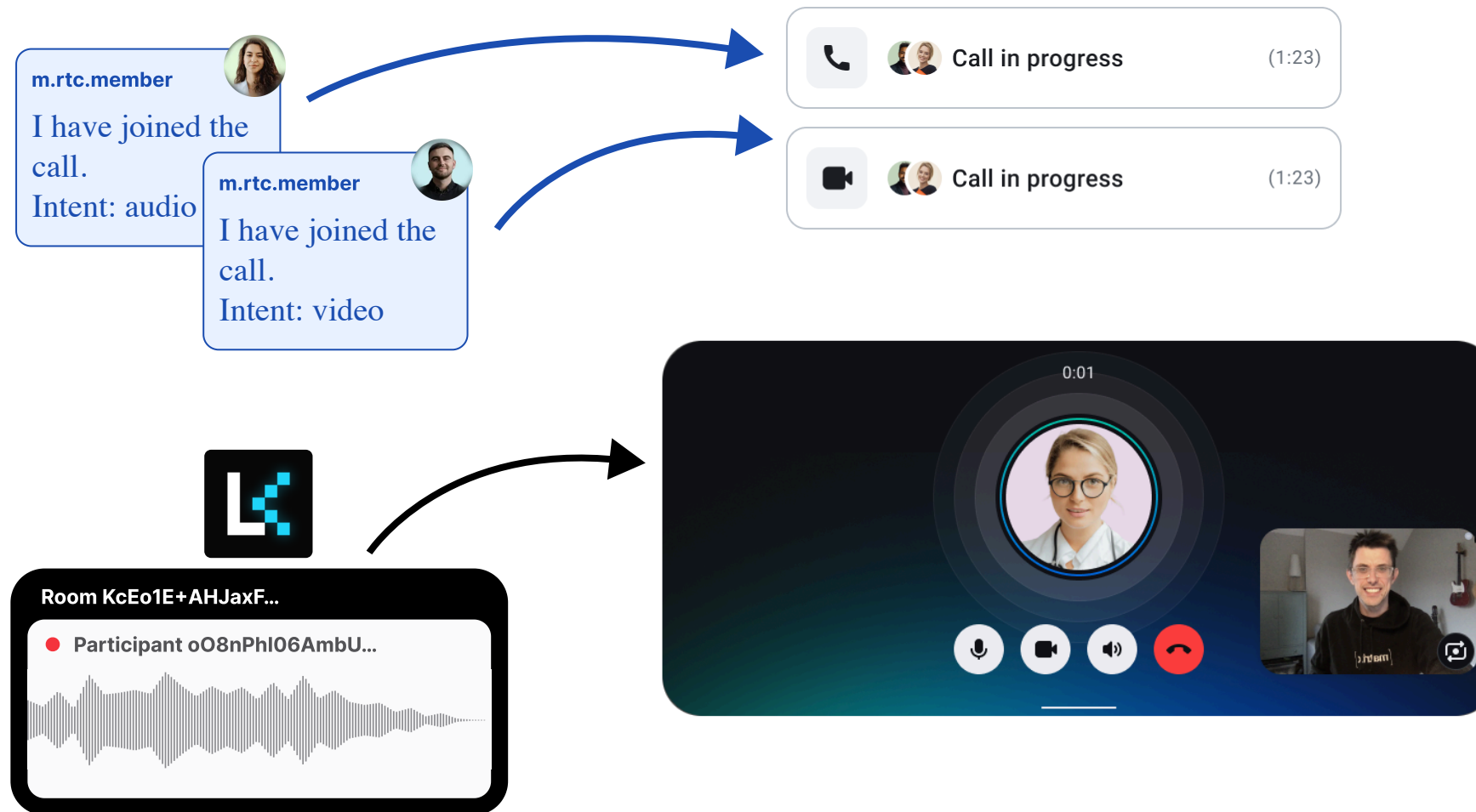
WebRTC



WebTransport

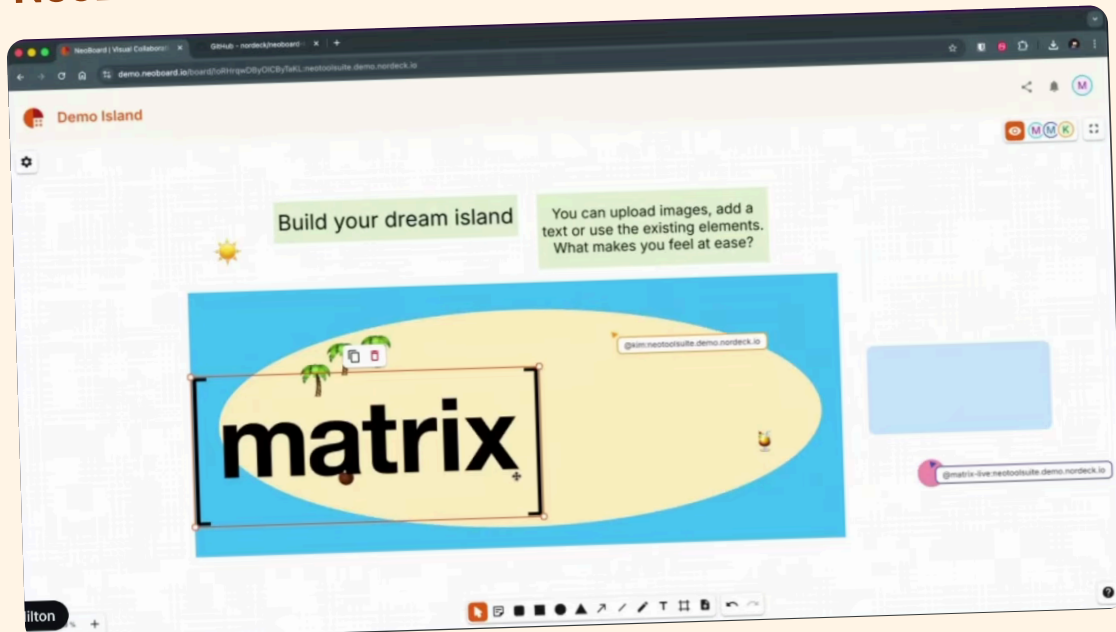
MSC4196: `m.call` application

Tell others how to interpret your media



Other applications?

NeoBoard · collaborative whiteboard

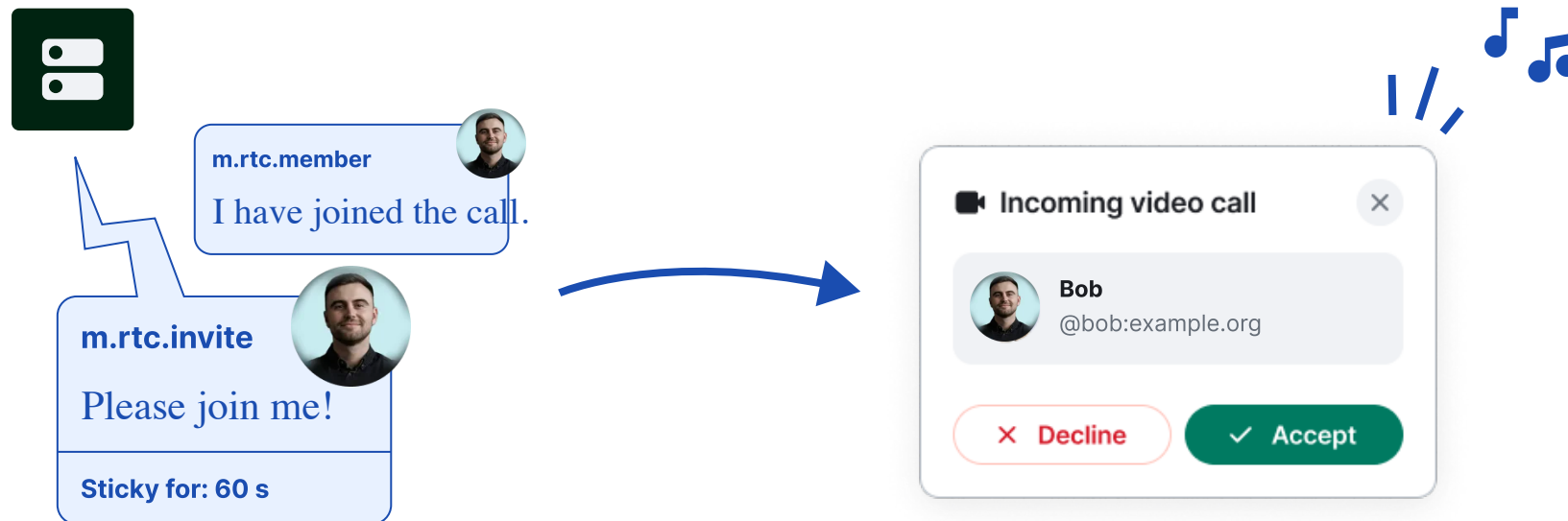


Flappy Royale · built with Godot



MSC4075: Invites & notifications

Invite others to an RTC session



(previously was `m.rtc.notification`)

3 • Ecosystem adoption

Better building blocks for adopting MatrixRTC

(we will talk about the "why" later)



Reminder

MatrixRTC "The Vision"

We want:

- Good documentation and SDKs to build on top of
- Have it adopted beyond chat
- Tools and extensions (recording, transcription, ...)
 - Don't miss Matthew's talk, or check out the recording!
- Great apps/implementations!

matrix-rust-rtc

One Rust codebase, exposed to Kotlin, Swift and the web.

Responsibilities:

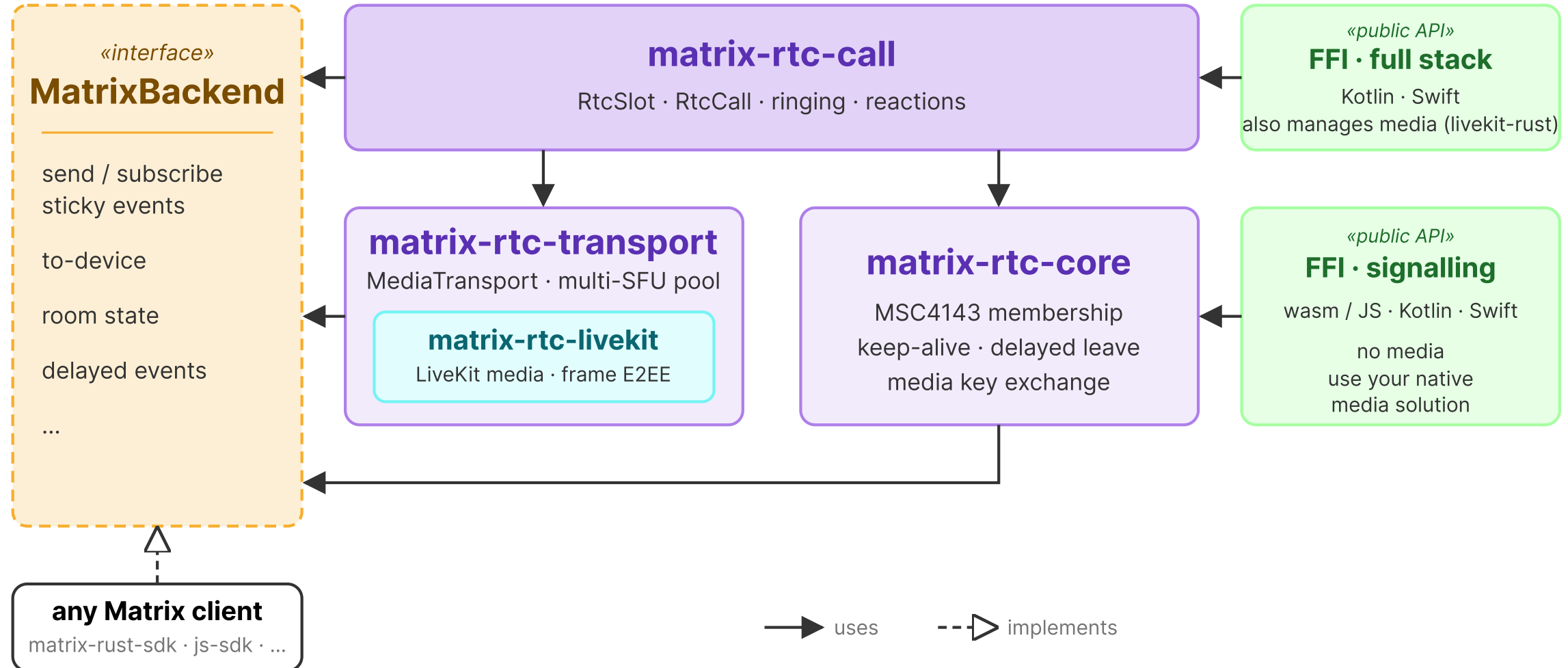
- Core
 - Slot, applications, memberships: abstract the primitives
 - Keep-alive, sticky refresh and delayed-leave handling
 - Media key exchange over Olm to-device, **key rotation at the deadline**
- Media
 - LiveKit abstraction
 - MSC4195 multi-SFU
- Calls
 - MSC4075 ringing, reactions, raised-hand queue
 - A ranked, smoothed **tile roster**: one tile per stream, screen shares first
 - Simulcast layer selection driven by rendered size and visibility

`matrix-rust-rtc` • Why? **For the ecosystem**

- It's a platform-agnostic implementation of MatrixRTC
- It's even Matrix-SDK agnostic

As flexible as possible: the ideal building block for MatrixRTC adoption.

matrix-rust-rtc · architecture



`matrix-rust-rtc` • Why? **For Element**

We want Element X to have a native implementation:

- No proper **CallKit** integration: locked-phone calls, system UI
- Screen sharing on mobile
- Startup cost of booting a second web app per call
- Audio routing and background audio bugs we could not fix from JS

We also want: *"Great apps/implementations!"*

We need it to ship a fully native mobile call experience.

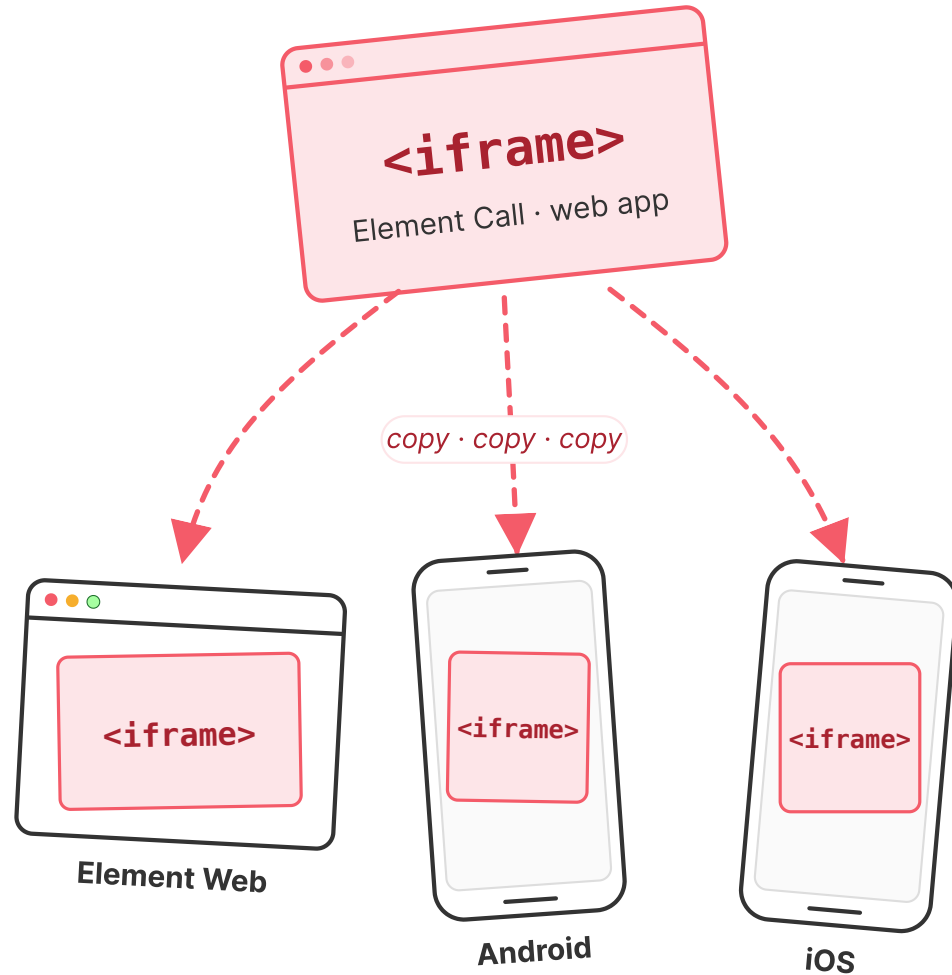
4 • Native calls on iOS & Android

The best possible Element X mobile experience



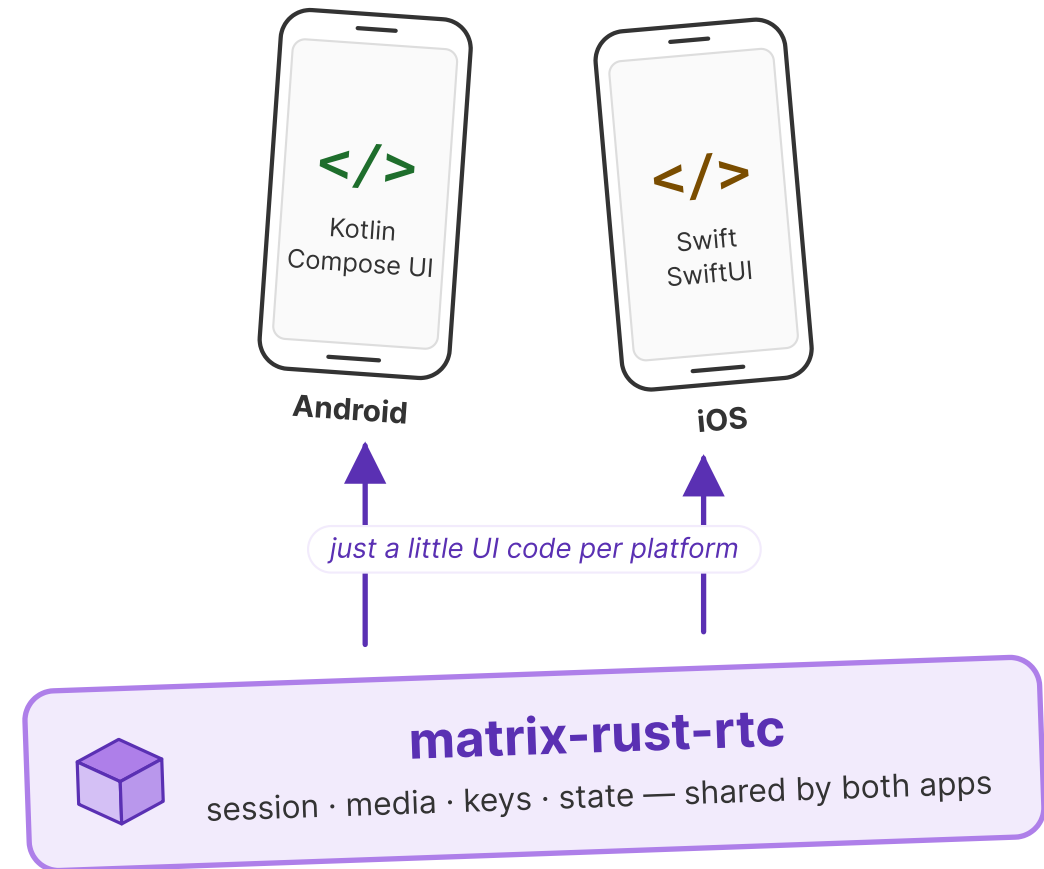
How we use `matrix-rust-rtc`

Before: one web widget everywhere



the same web view, wrapped three times · no real OS integration

Goal: native UIs, one shared core



native look and OS features everywhere · the hard parts stay shared

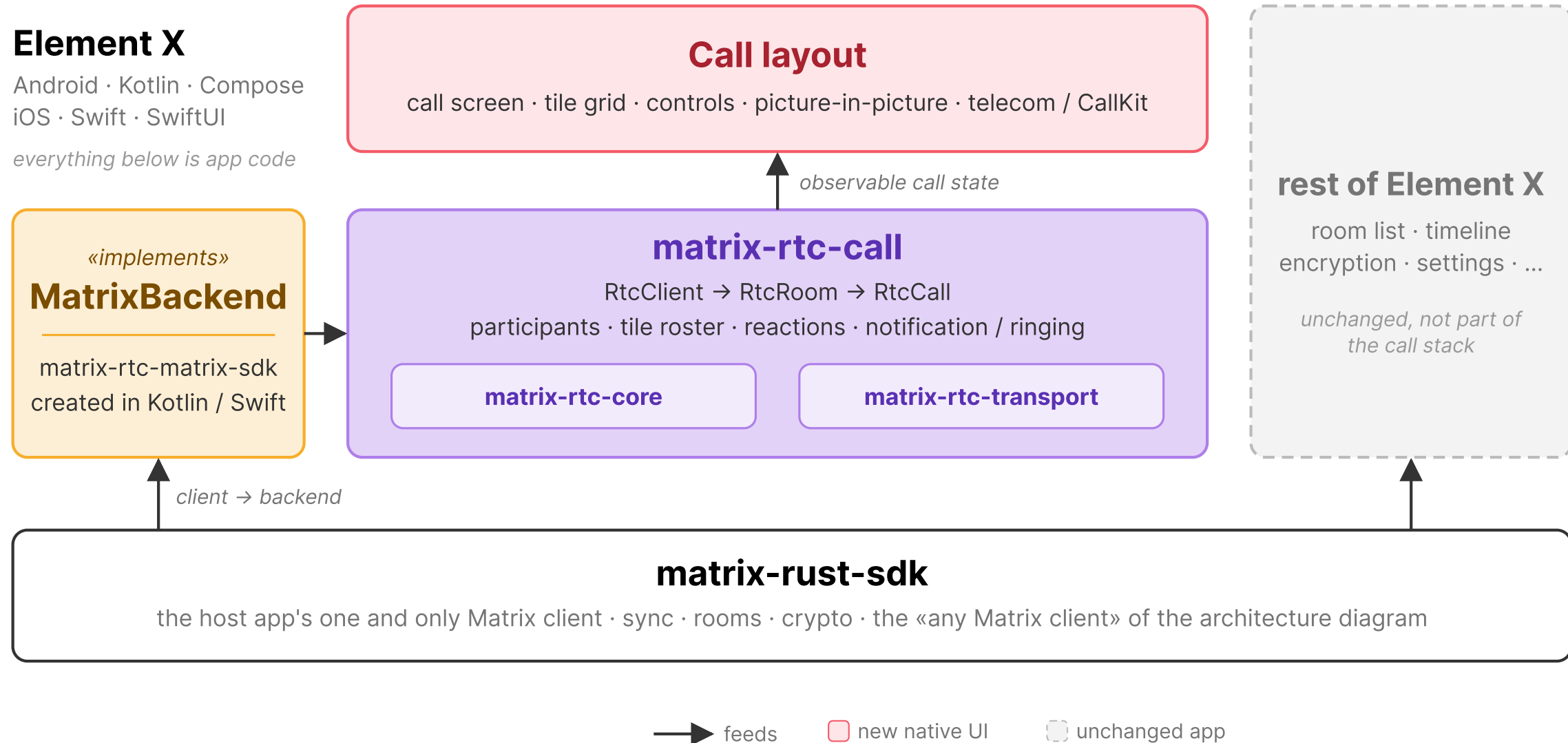
How it works

Element X

Element X

Android · Kotlin · Compose
iOS · Swift · SwiftUI

everything below is app code



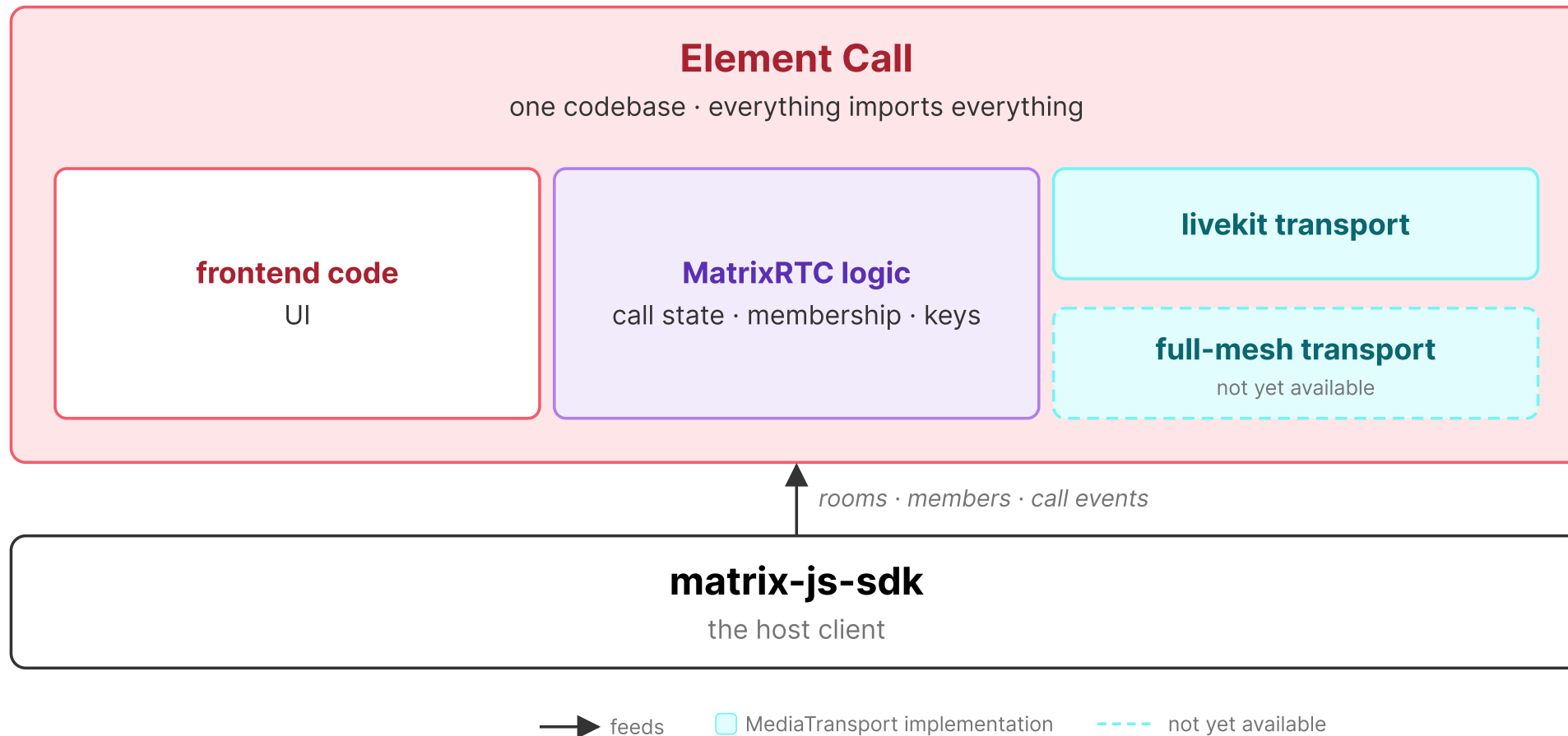
5 • What about Web?

- **Element X** needs `matrix-rust-rtc`
- Currently, `matrix-js-sdk` reimplements MatrixRTC 😓
- → **Element Call Web** should be built on `matrix-rust-rtc`

What about Web?

same Rust SDK as mobile · shared code · same interface

0 Last release: one codebase



What about Web?

same Rust SDK as mobile · shared code · same interface

0 Last release: one codebase

1 Today: split model and frontend

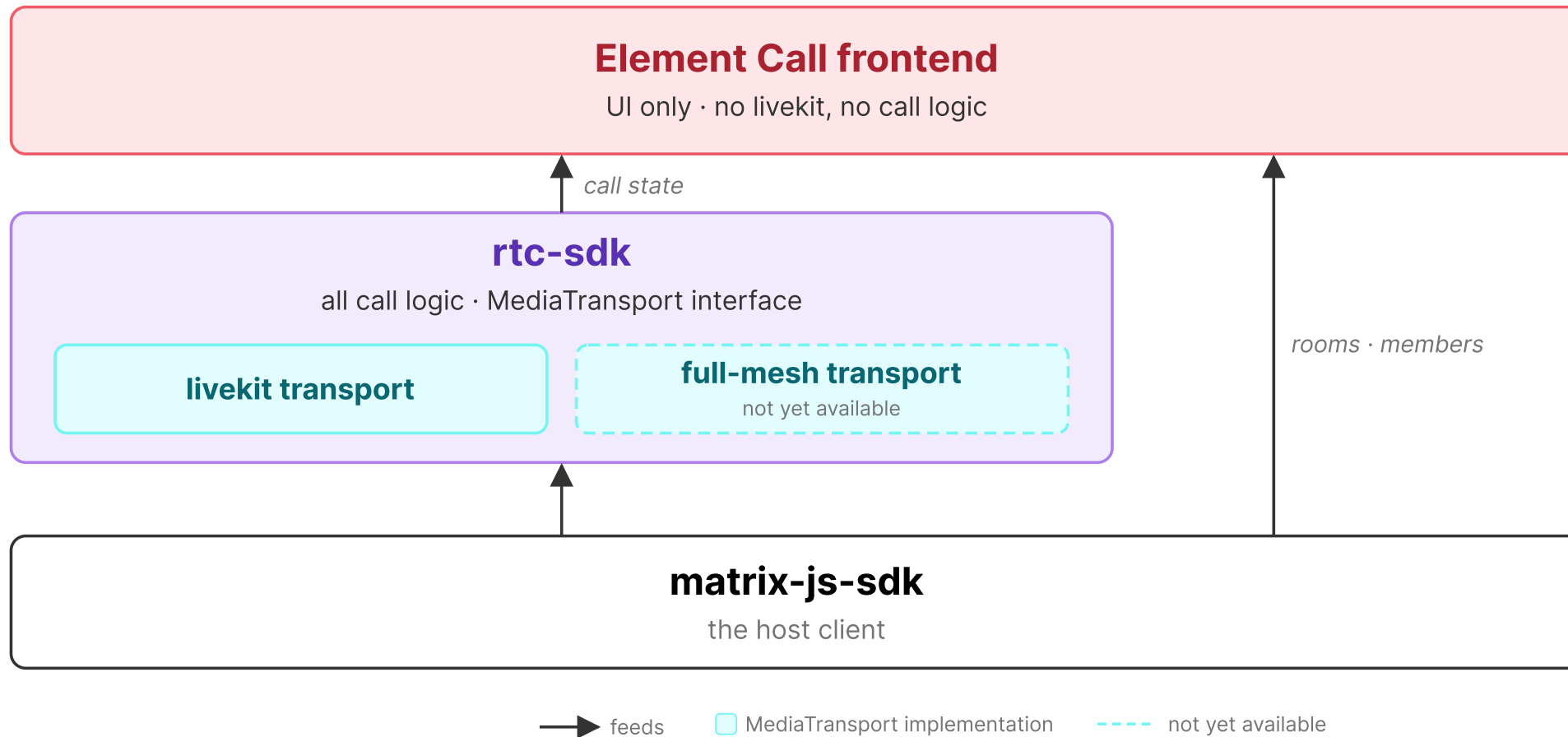
- **SDK module**: call independent, very high level, same interface as `matrix-rust-rtc`
- **Element Call**: only the frontend, gets much simpler

What about Web?

same Rust SDK as mobile · shared code · same interface

0 Last release: one codebase

1 Today: split model and frontend



What about Web?

same Rust SDK as mobile · shared code · same interface

0 Last release: one codebase

1 Today: split model and frontend

2 Next: SDK module on the Rust SDK

- Refactor the **SDK module** to use `matrix-rust-rtc` (via wasm)
- Element Call is **unchanged**

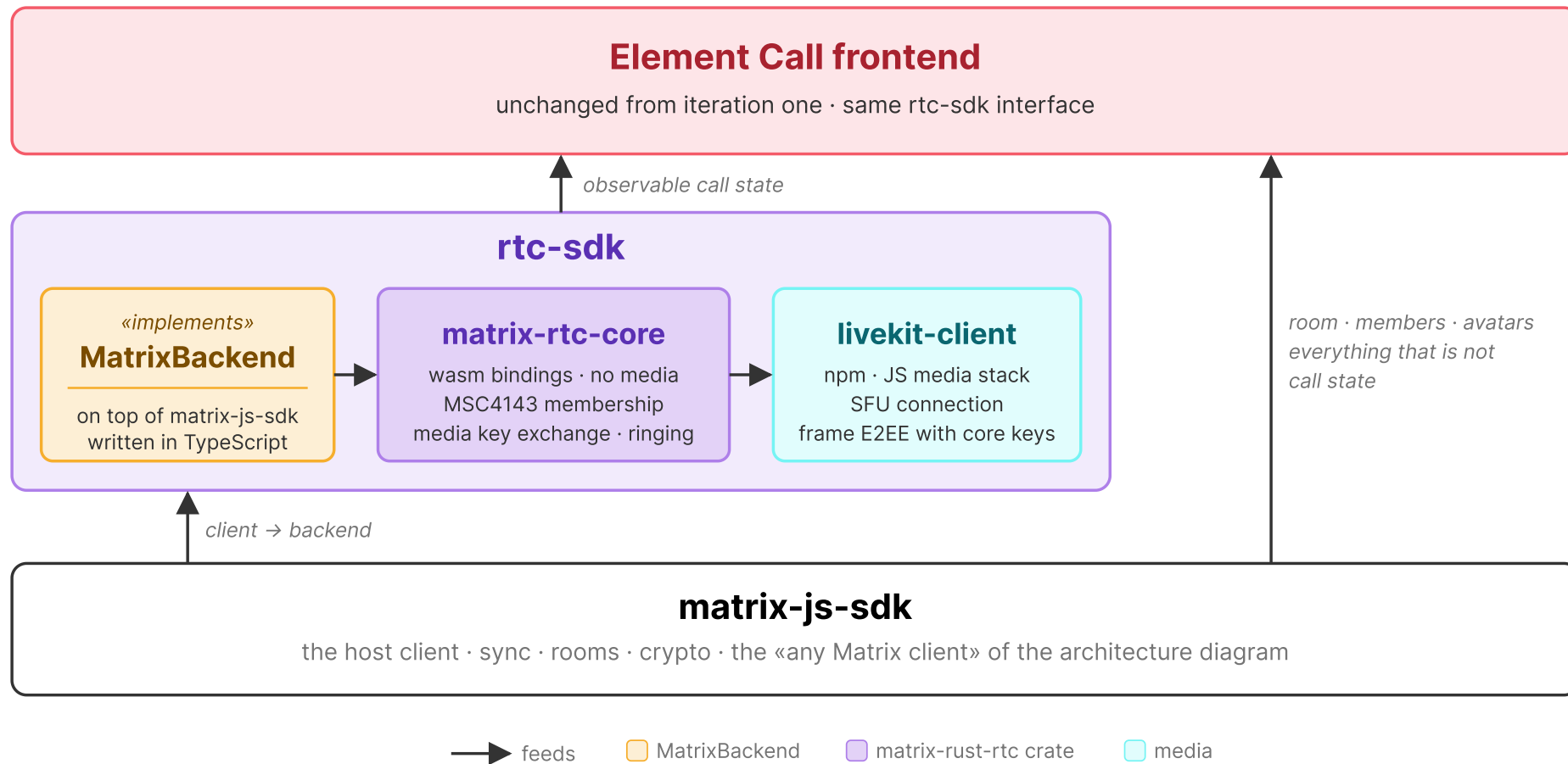
What about Web?

same Rust SDK as mobile · shared code · same interface

0 Last release: one codebase

1 Today: split model and frontend

2 Next: SDK module on the Rust SDK



MatrixRTC JS SDK example

```
const slot = createRTCslot(
  scope,
  client, // a matrix-js-sdk MatrixClient, logged in and syncing
  room, // the matrix-js-sdk Room, from client.getRoom() once the join has synced
  {
    encryptionSystem: { kind: E2eeType.PER_PARTICIPANT },
    matrixRTCMode: MatrixRTCMode.Compatibility,
  },
);
const participation = slot.join({
  publish: [{ source: "microphone" }, { source: "camera" }],
});

// A `video` for each remote member's camera and an `audio` for its microphone.
// A track plays on an element once attached to it; detach before the element
// leaves the DOM. (`tileFor`, `setName` and `removeTilesExcept` are the host's.)
participation.remoteMembers$.subscribe((members) => {
  for (const member of members) {
    const { video, audio } = tileFor(member.memberId);
    trackBySource$(scope, member.tracks$, "camera").subscribe((track) =>
      track?.attach(video),
    );
    trackBySource$(scope, member.tracks$, "microphone").subscribe((track) =>
      track?.attach(audio),
    );
  }
  removeTilesExcept(members.map((m) => m.memberId));
});
```

MatrixRTC JS SDK

what's to like

- Automatically picks up `matrix-rust-rtc` improvements
- Application agnostic by design (great for other projects)
 - Similar to the approach used for **FlappyRoyal** and **FloorItIpsum**
- Allows us to move fast with the frontend
- Code is shared via SDK
 - Widget → React component

It also enables pop-out!

The iframe was the last blocker for OS picture-in-picture



It's happening

A fully federated, per-sender-key encrypted, backend-agnostic RTC solution for Matrix.

WHAT'S NEXT

Spec

Merged! Docs are coming

Clients

More MatrixRTC clients are coming

Tools

SDKs for Swift, Kotlin and JS

We can't wait to see the ecosystem adopt **MatrixRTC**

Thanks

- MSC4143 · github.com/matrix-org/matrix-spec-proposals/pull/4143
- Element Call · github.com/element-hq/element-call
- `matrix-rust-rtc` · github.com/element-hq/matrix-rust-rtc
- `element-call-ios` · github.com/element-hq/element-call-ios
- `element-call-android` · github.com/element-hq/element-call-android

#matrixrtc:matrix.org

