matrix

RTC

The (not anymore) missing piece to enable great video conferences

timok@element.io
@toger5:matrix.org

Why I am excited about giving this talk?

Proposals are converging

Initial implementations are becoming stable

The **MatrixRTC** approach checks more boxes for VoIP than ever before

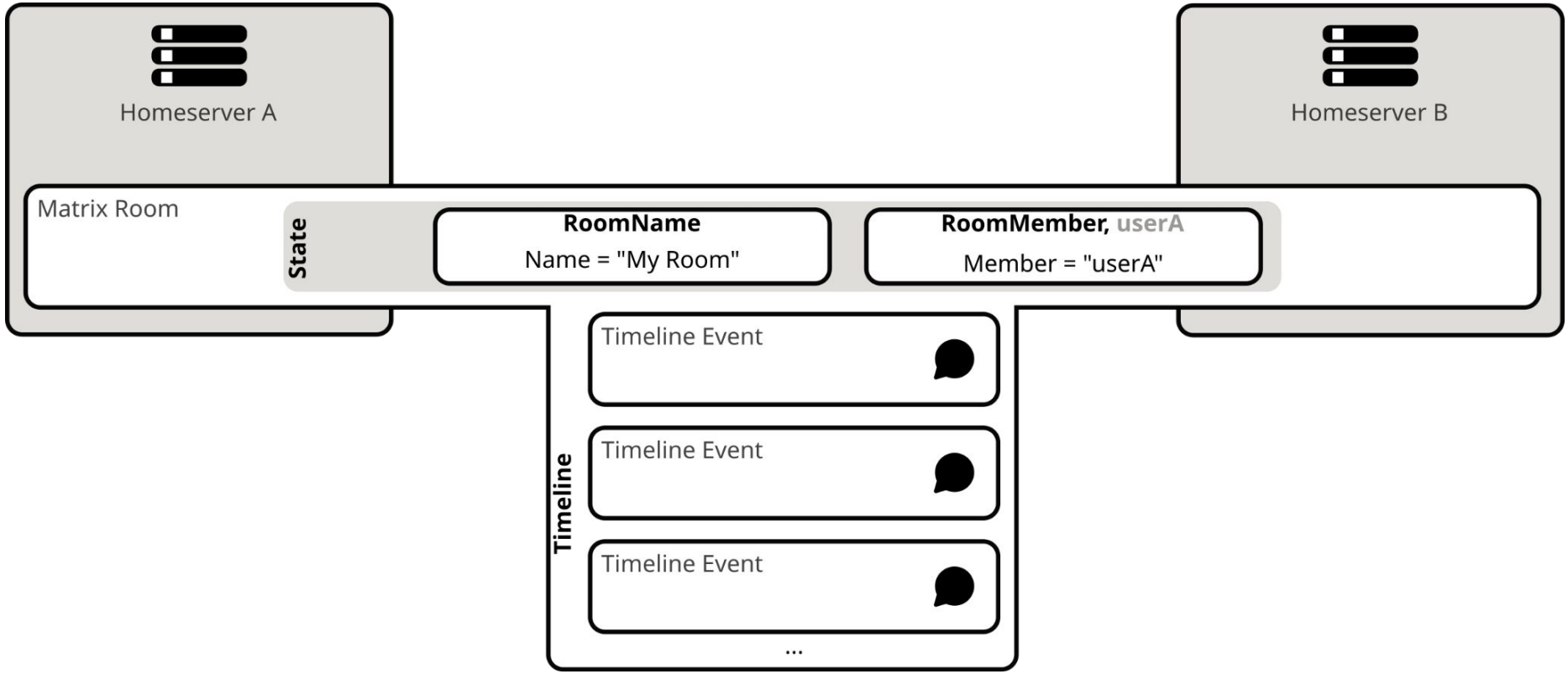
- ✅ Scalable
- ✅ Interchangeable components
- ✅ Very flexible! (can be used for much more than VoIP)
- ✅ Secure, Federated, verified identities ...

The first time where we can have a complete technical summary of the MatrixRTC modules that enable VoIP over Matrix

Which we will do now!

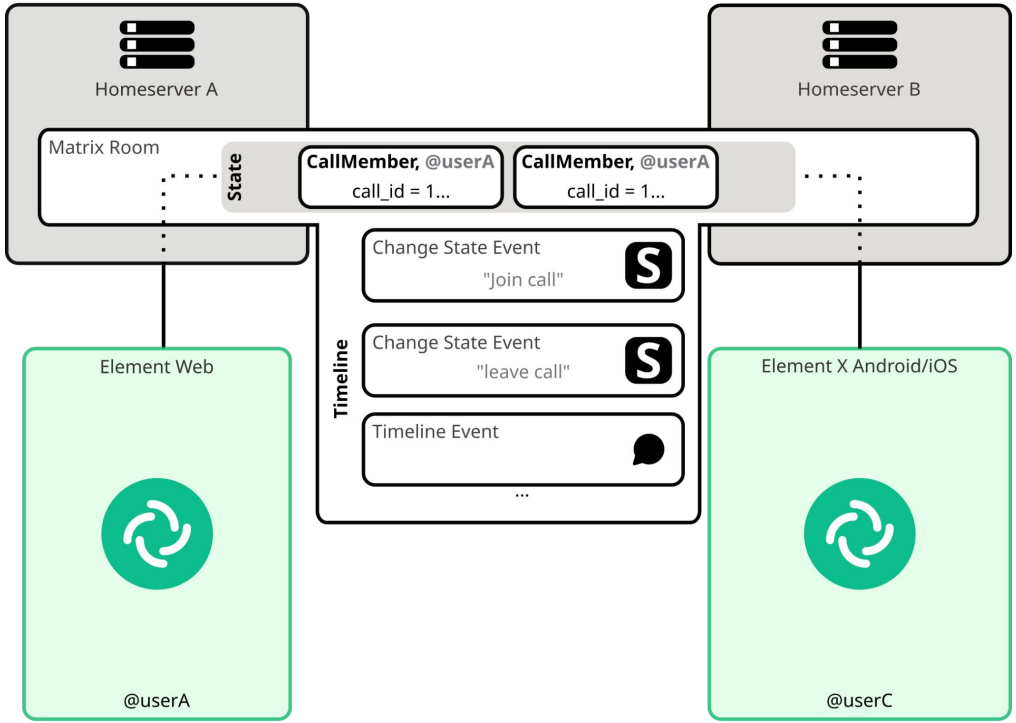
Let’s look at Federated Chat Rooms

Where does MatrixRTC fit in?



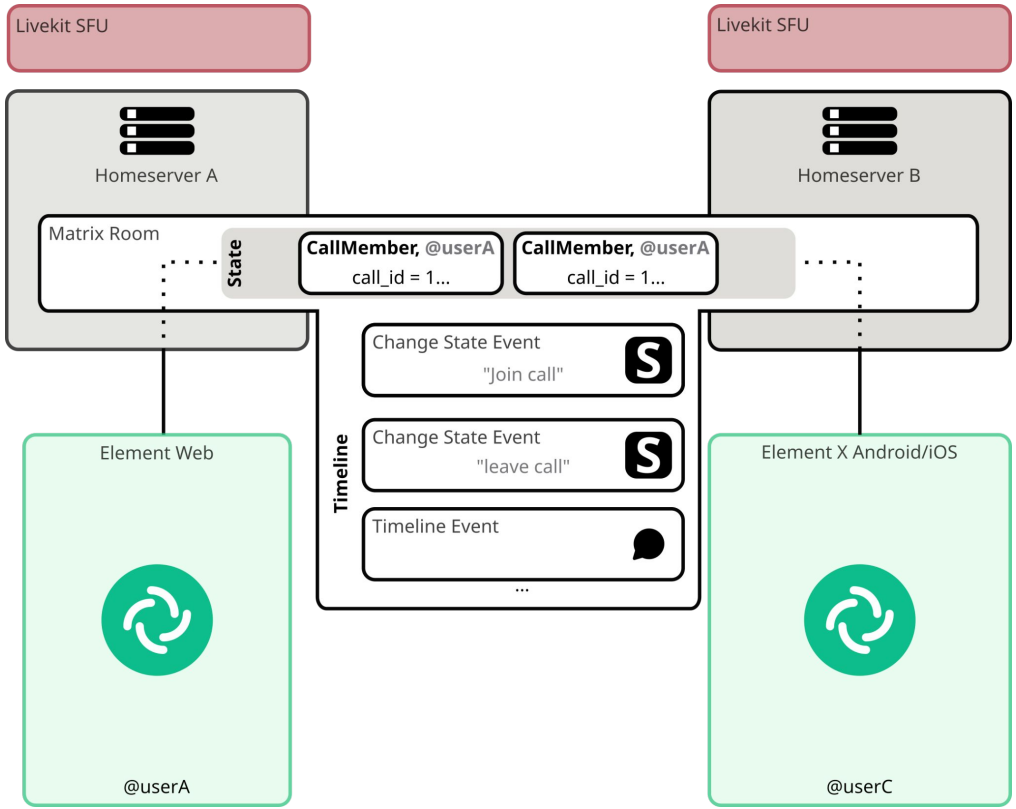
MatrixRTC Components (Agenda)

- **WebRTC infrastructure**
- **Signaling**
  - Exchange Backend information (SFU)
  - communicate call participation
- **Metadata**
  - Call History
  - Ringing
- **Encryption**



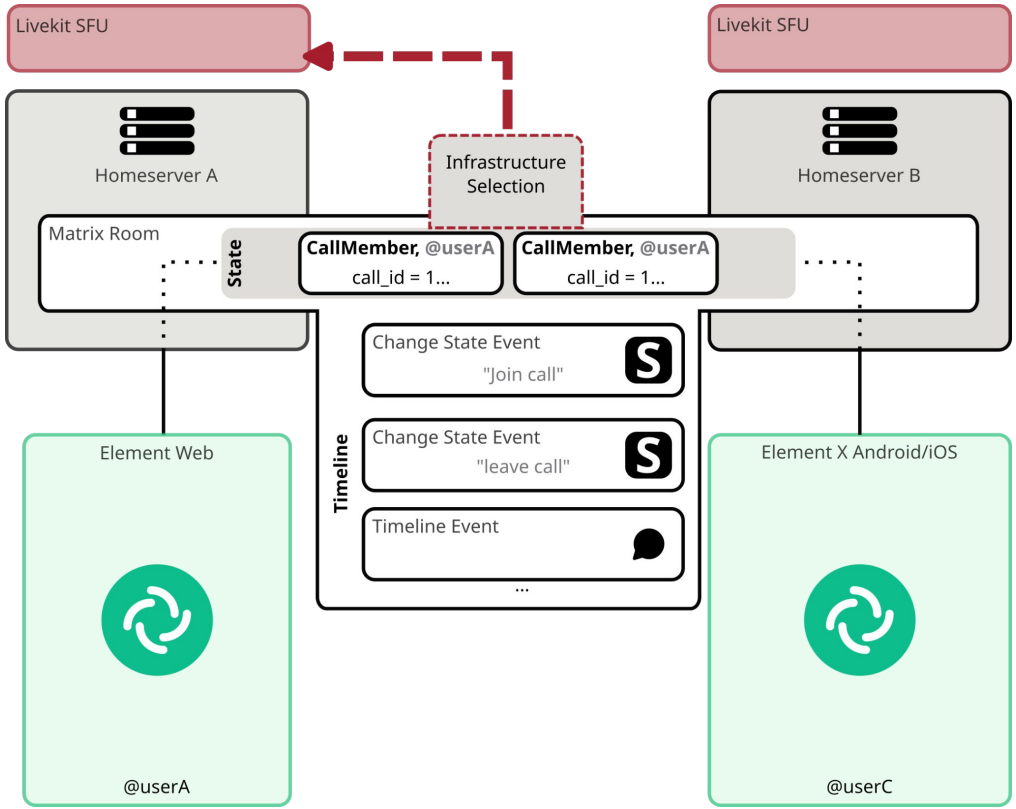
MatrixRTC Components (Agenda)

- **WebRTC infrastructure**
- **Signaling**
  - Exchange Backend information (SFU)
  - communicate call participation
- **Metadata**
  - Call History
  - Ringing
- **Encryption**



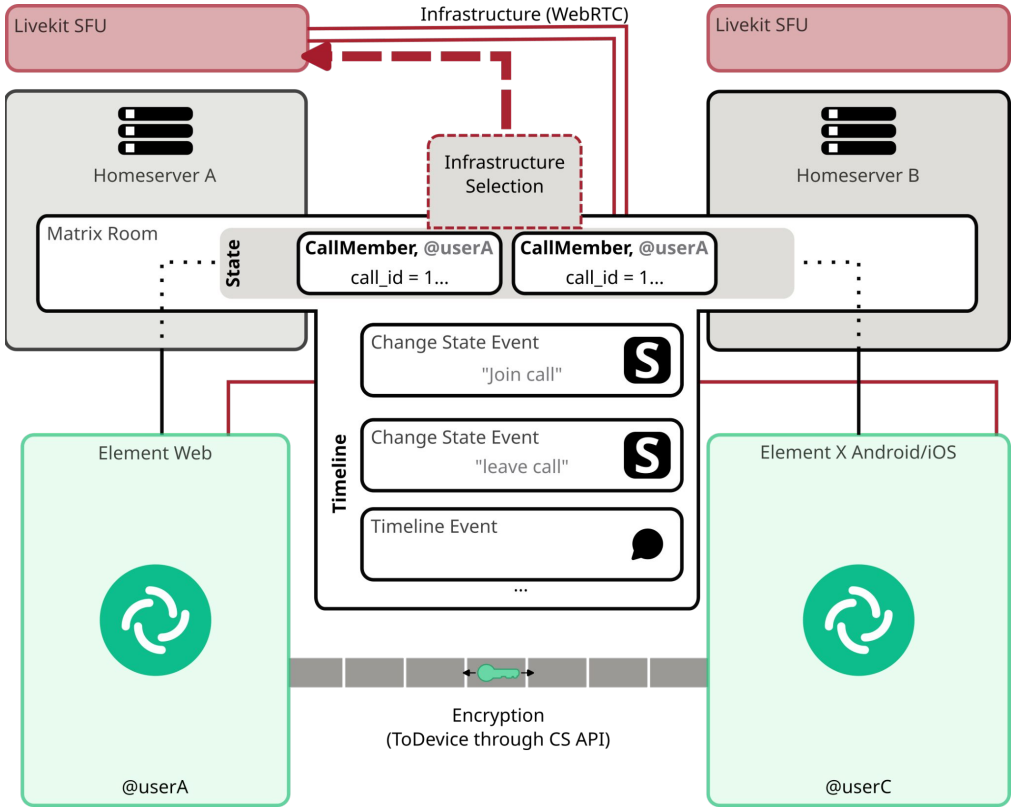
MatrixRTC Components (Agenda)

- **WebRTC infrastructure**
- **Signaling**
  - Exchange Backend information (SFU)
  - communicate call participation
- **Metadata**
  - Call History
  - Ringing
- **Encryption**



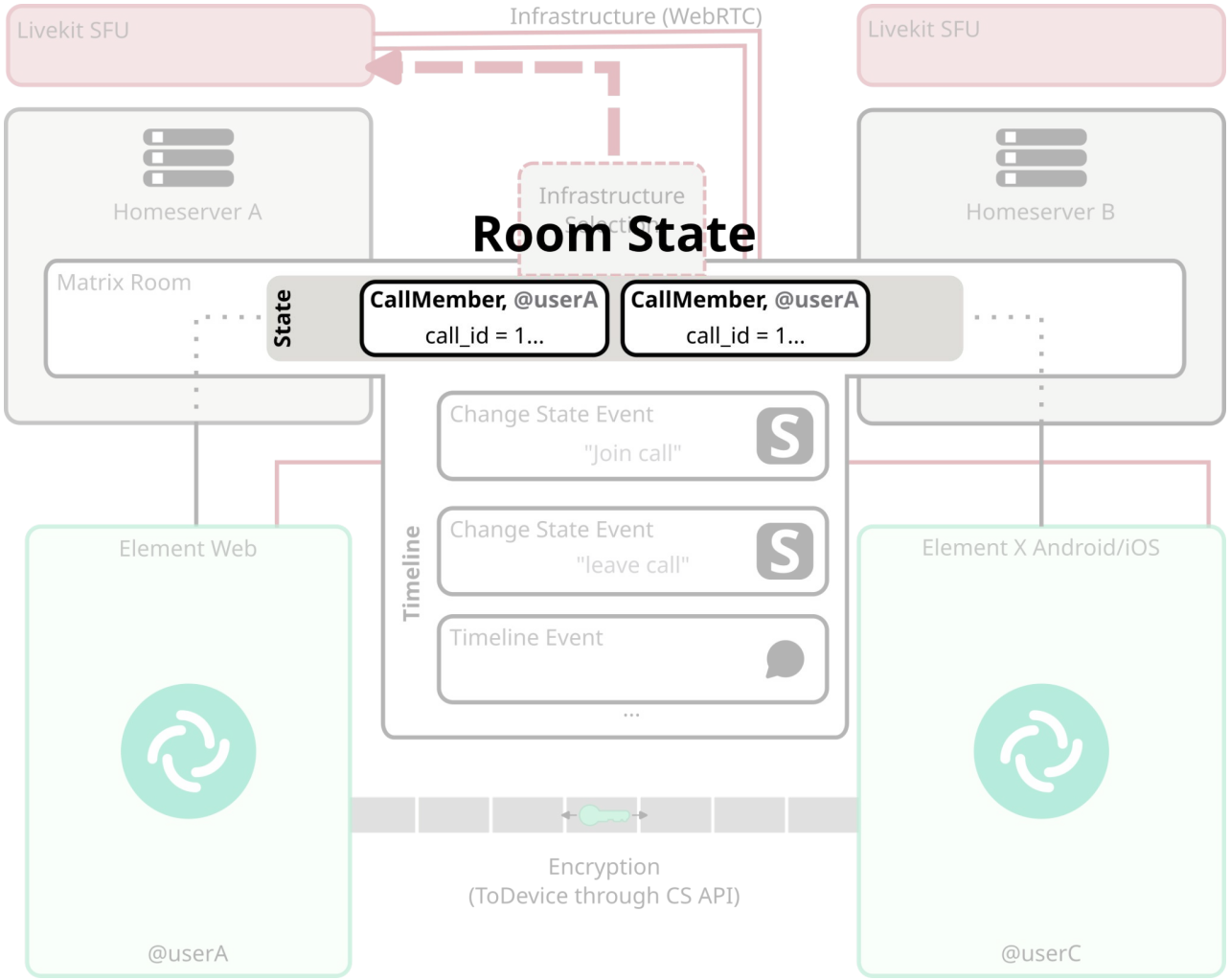
MatrixRTC Components (Agenda)

- **WebRTC infrastructure**
- **Signaling**
  - Exchange Backend information (SFU)
  - communicate call participation
- **Metadata**
  - Call History
  - Ringing
- **Encryption**



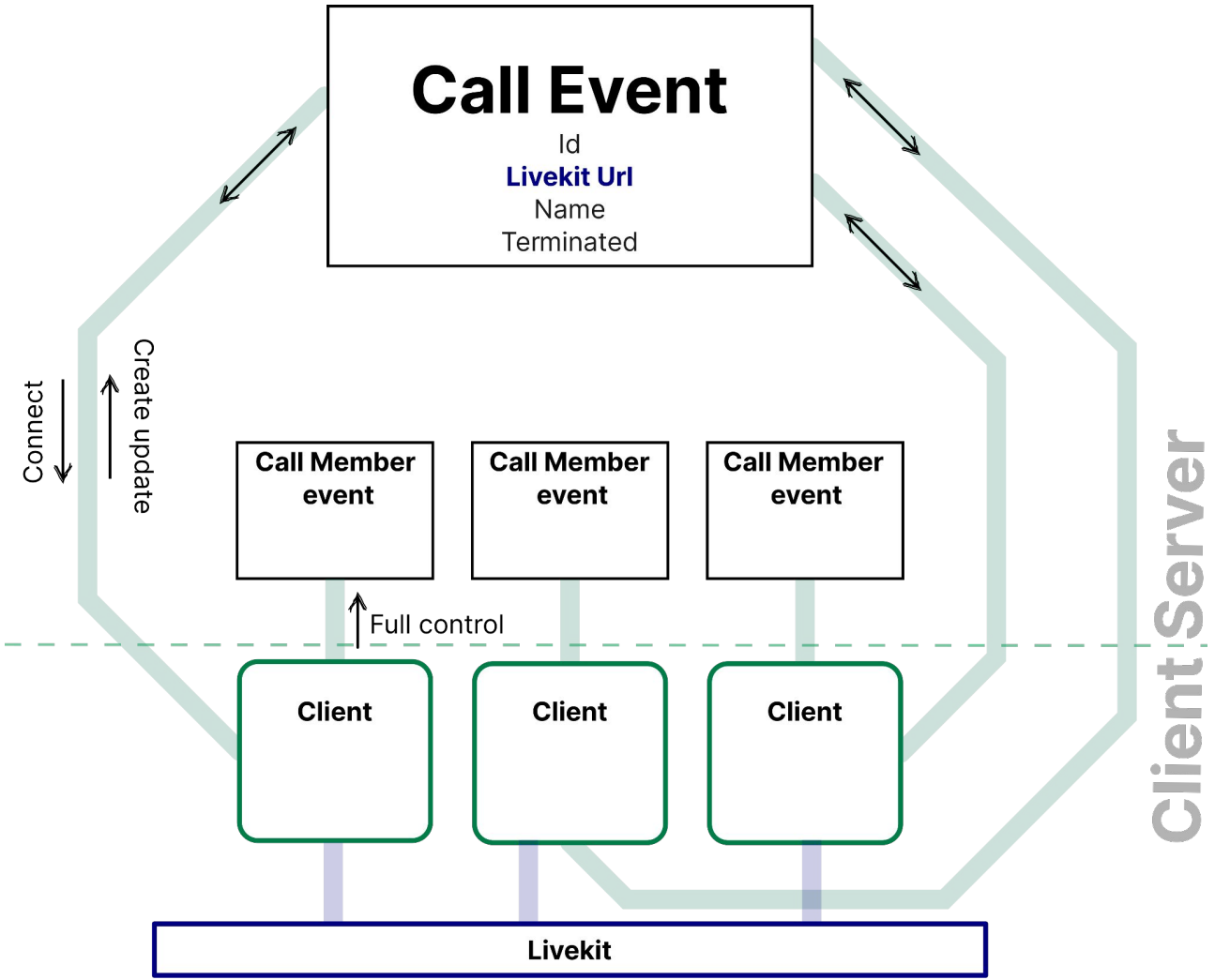
Room State

High level signalling


Livekit SFU
Infrastructure (WebRTC)
Livekit SFU
Homeserver A
Infrastructure
Homeserver B
Room State
Matrix Room
State
CallMember, @userA
call_id = 1...
CallMember, @userA
call_id = 1...
Change State Event
"Join call"
Change State Event
"leave call"
Timeline
Timeline Event
...
Element Web
Element X Android/iOS
Encryption
(ToDevice through CS API)
@userA
@userC

Initial (Intuitive) approach

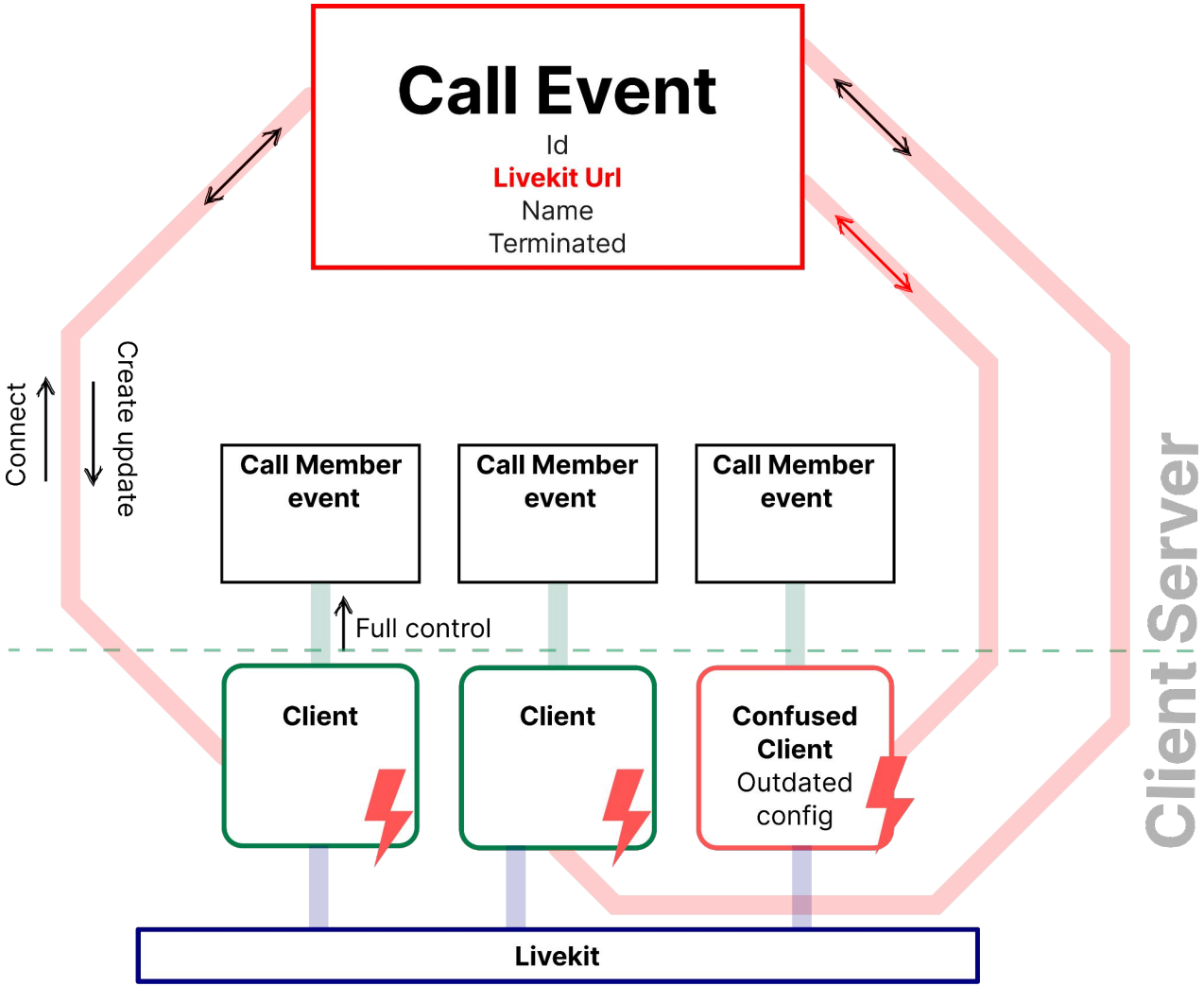
- A call is a state event
- It advocates the Infrastructure it will use.



Initial (Intuitive) approach

- A call is a state event
- It advocates the Infrastructure it will use.

- **One broken/malicious client can break it for everyone**

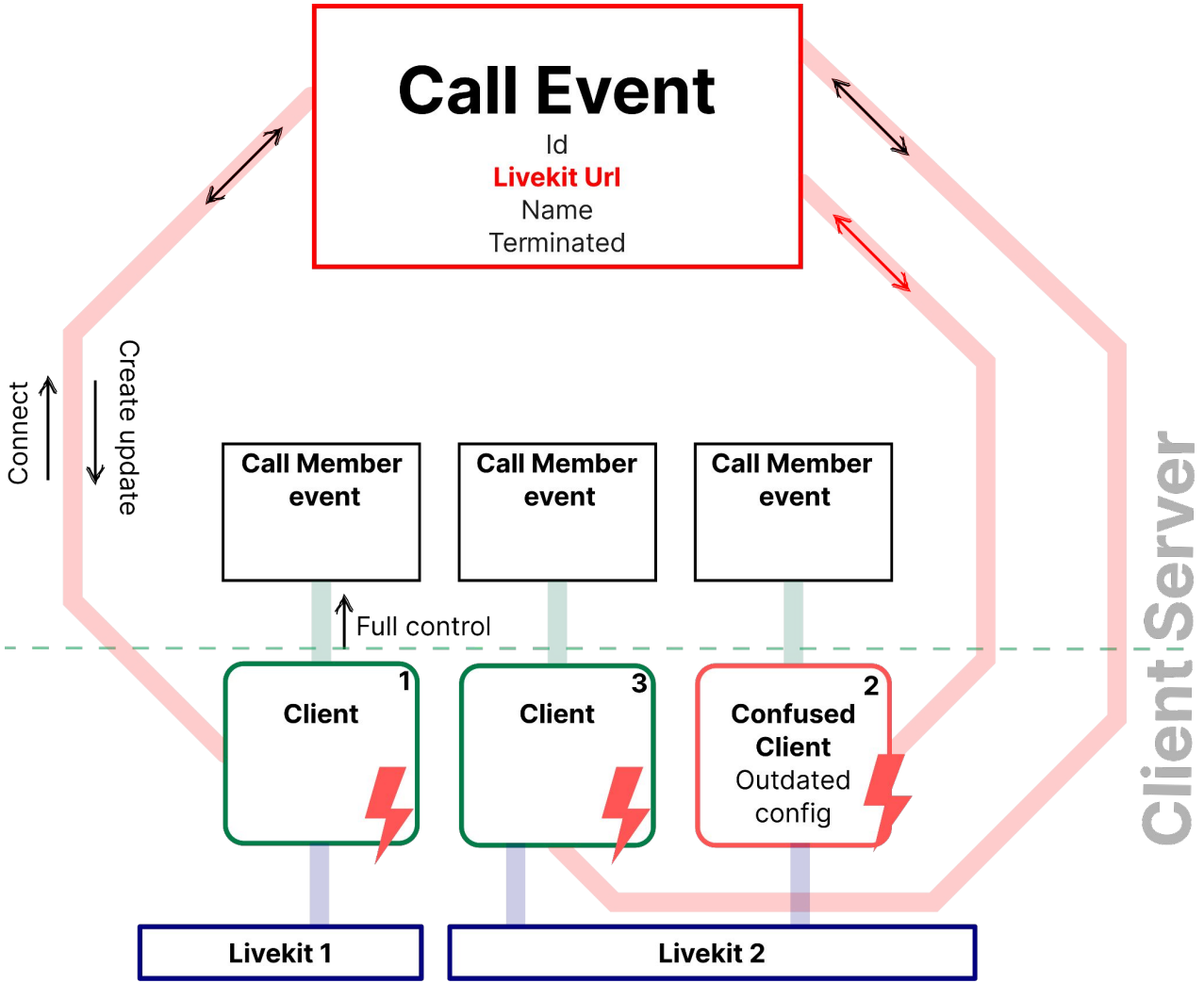


Initial (Intuitive) approach

- A call is a state event
- It advocates the Infrastructure it will use.

- **One broken/ malicious client can break it for everyone**

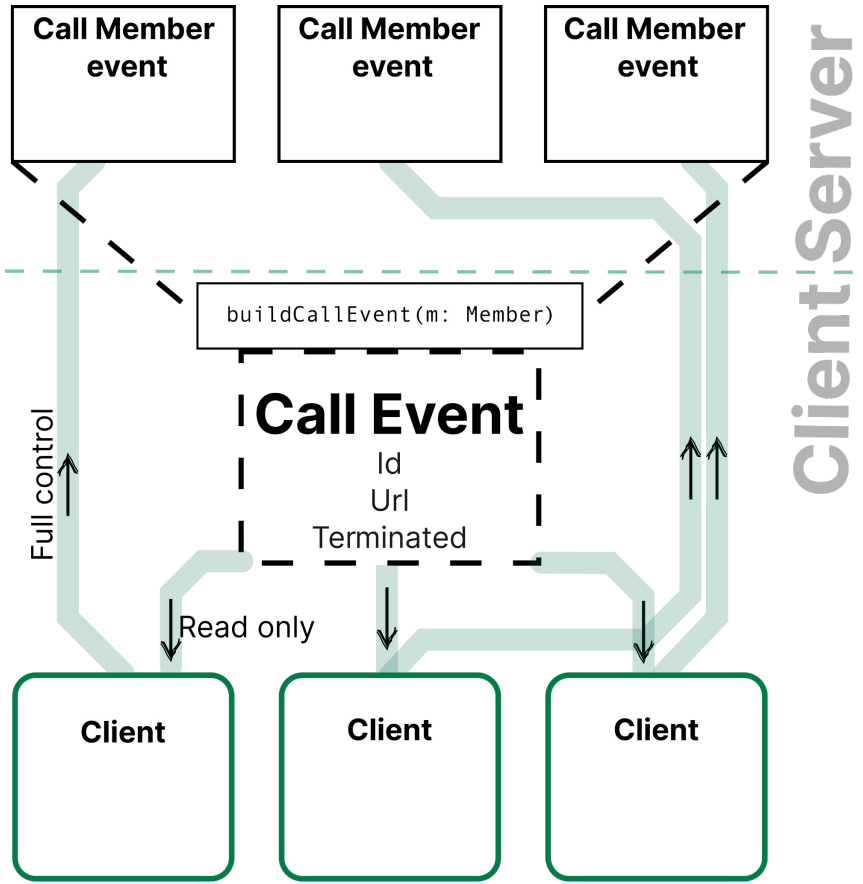
- If not listening to url changes → splitbrain



Solution: Calls without a Call Event

Only use Call Member events!

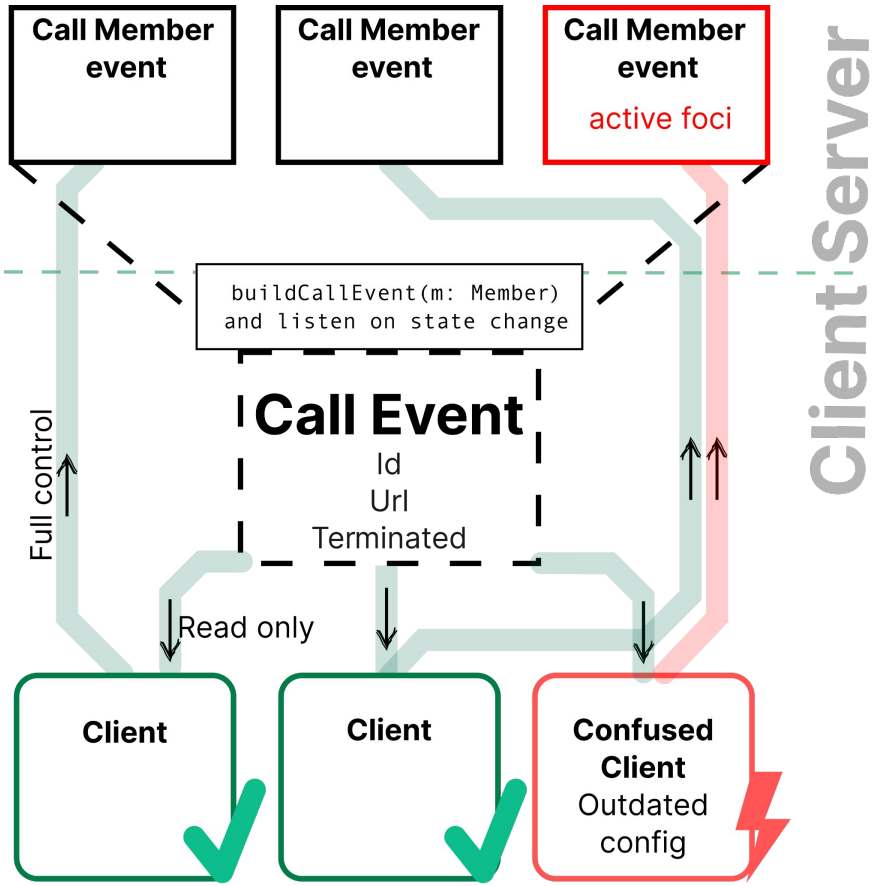
- Clear ownership model
  - No overriding
- Computed CallEvent
  - The Call Event is not necessary
  - Situation dependent transparent glaring. (democratic, admin, ...)



Solution: Calls without a Call Event

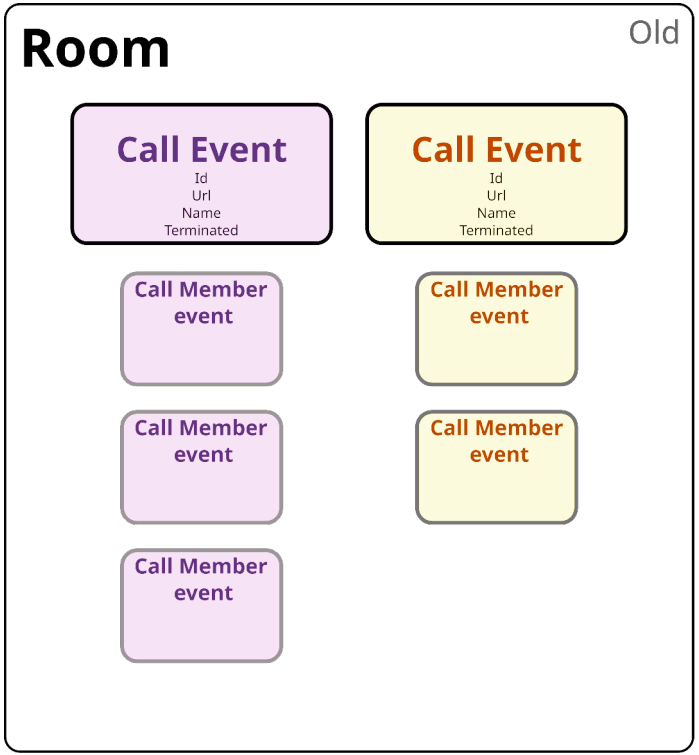
Only use Call Member events!

- Clear ownership model
  - No overriding
- Computed CallEvent
  - The Call Event is not necessary
  - Situation dependent transparent glaring. (democratic, admin, ...)
- **One broken/malicious client does not have the power to break the experience for the others!**

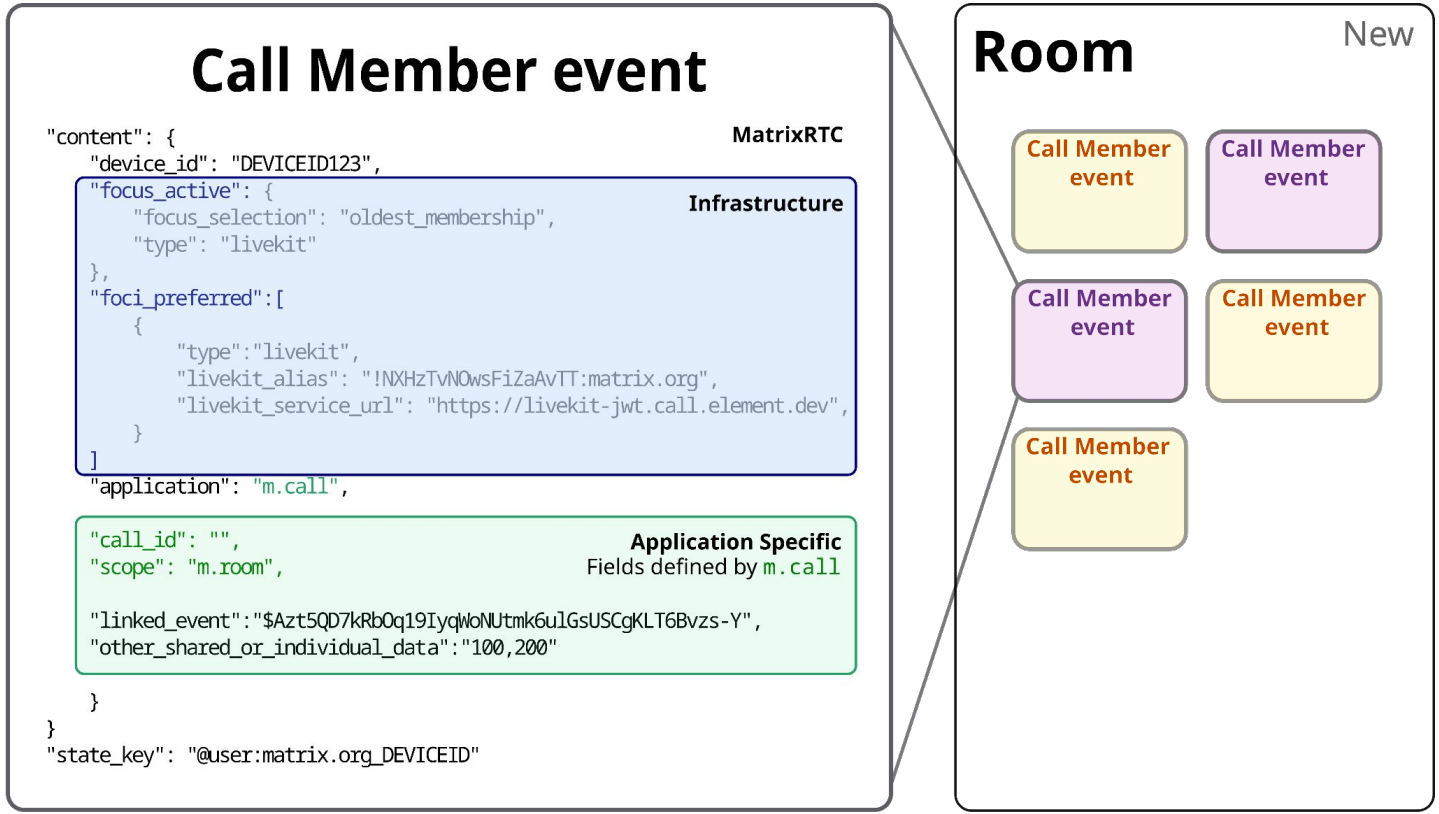


MatrixRTC is now only a collection of members.

- Every room can always be used as a MatrixRTC session
- Everything else would be application specific



Let’s have a closer look at the call member Event



MatrixRTC / App Specific / Infrastructure

MatrixRTC
- Defines Sessions

Infrastructure
- Is replaceable.
- Specced separately.
- Allow adapting emerging technologies.
- Independent communication system (websocket, matrix ToDevice, ...)

App Specific
- App specific state fields
- Can have multiple implementations

Call Member event

MatrixRTC

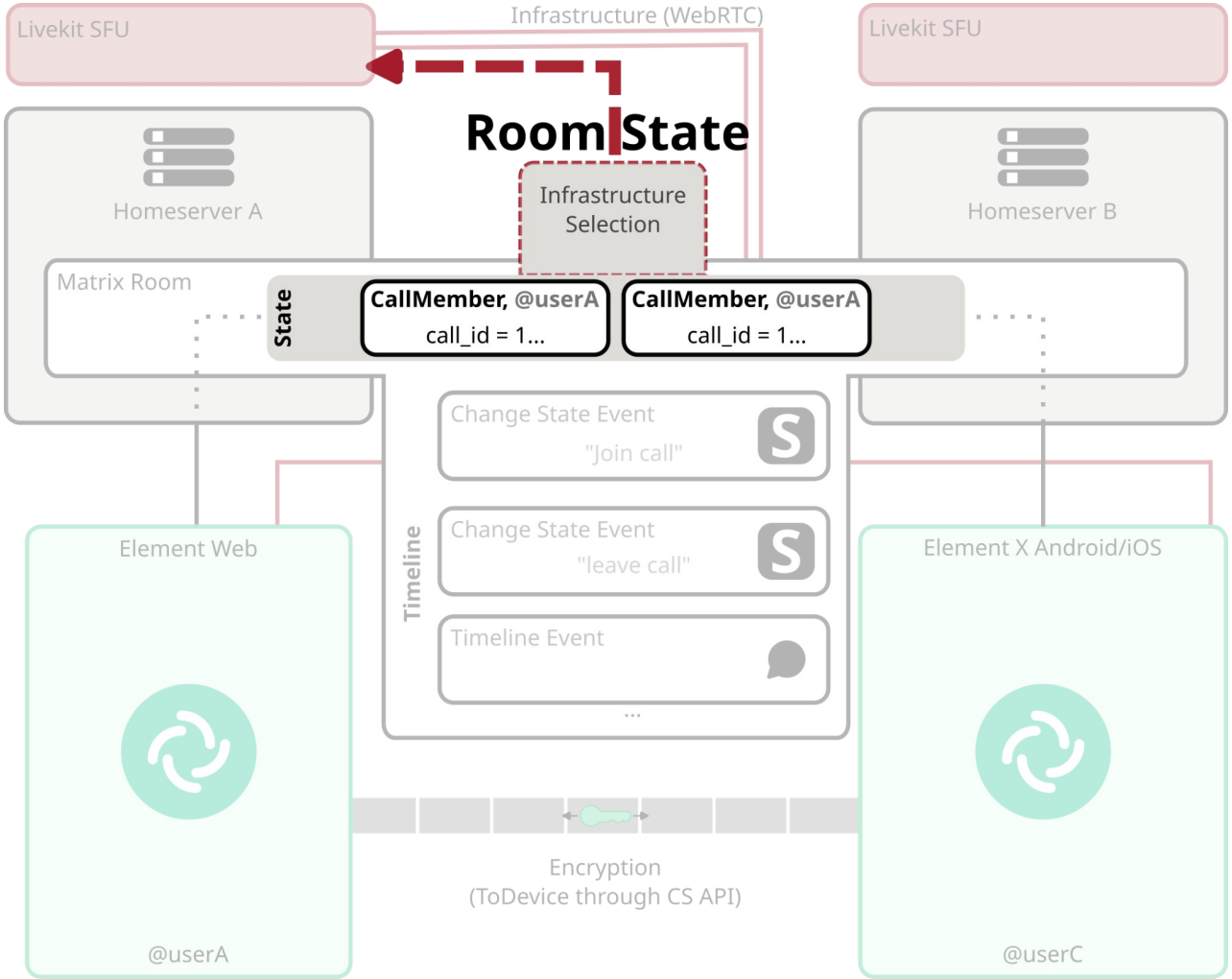
```
"content": {
    "device_id": "DEVICEID123",
    "focus_active": {
        "focus_selection": "oldest_membership",
        "type": "livekit"
    },
    "foci_preferred":[
        {
            "type":"livekit",
            "livekit_alias": "!NXHzTvNOwsFiZaAvTT:matrix.org",
            "livekit_service_url": "https://livekit-jwt.call.element.dev",
        }
    ]
    "application": "m.call",
    "call_id": "",
    "scope": "m.room",

    "linked_event":"$Azt5QD7kRbOq19IyqWoNUtmk6ulGsUSCgKLT6Bvzs-Y",
    "other_shared_or_individual_data":"100,200"

    }
}
"state_key": "@user:matrix.org_DEVICEID"
```

Infrastructure

Application Specific
Fields defined by `m.call`

Room State

Focus Selection


Livekit SFU
Infrastructure (WebRTC)
Livekit SFU
RoomState
Infrastructure Selection
Homeserver A
Homeserver B
Matrix Room
State
CallMember, @userA
call_id = 1...
CallMember, @userA
call_id = 1...
Change State Event
"Join call"
Change State Event
"leave call"
Timeline
Timeline Event
...
Element Web
Element X Android/iOS
Encryption
(ToDevice through CS API)
@userA
@userC

Focus Selection

foci_active:
“The algorithm that defines how to choose a focus for this member”

foci_preferred:
“The input data for this algorithm contributed by this member”

Note: The **focus_active** needs to be designed so that all participants converge to the same SFU/focus.

Call Member event

MatrixRTC

Infrastructure

Application Specific
Fields defined by `m.call`

```
"content": {
    "device_id": "DEVICEID123",
    "focus_active": {
        "focus_selection": "oldest_membership",
        "type": "livekit"
    },
    "foci_preferred":[
        {
            "type":"livekit",
            "livekit_alias": "!NXHzTvNOwsFiZaAvTT:matrix.org",
            "livekit_service_url": "https://livekit-jwt.call.element.dev",
        }
    ]
    "application": "m.call",

    "call_id": "",
    "scope": "m.room",

    "linked_event":"$Azt5QD7kRbOq19IyqWoNUtmk6ulGsUSCgKLT6Bvzs-Y",
    "other_shared_or_individual_data":"100,200"

    }
}
"state_key": "@user:matrix.org_DEVICEID"
```

Revival of Full-Mesh

“Active foci is the algorithm/method to connect to a member.”

Full mesh can work

- Another focus type

Future:
A focus type that starts with full_mesh and scales:

- full_mesh_into_livekit

Call Member event

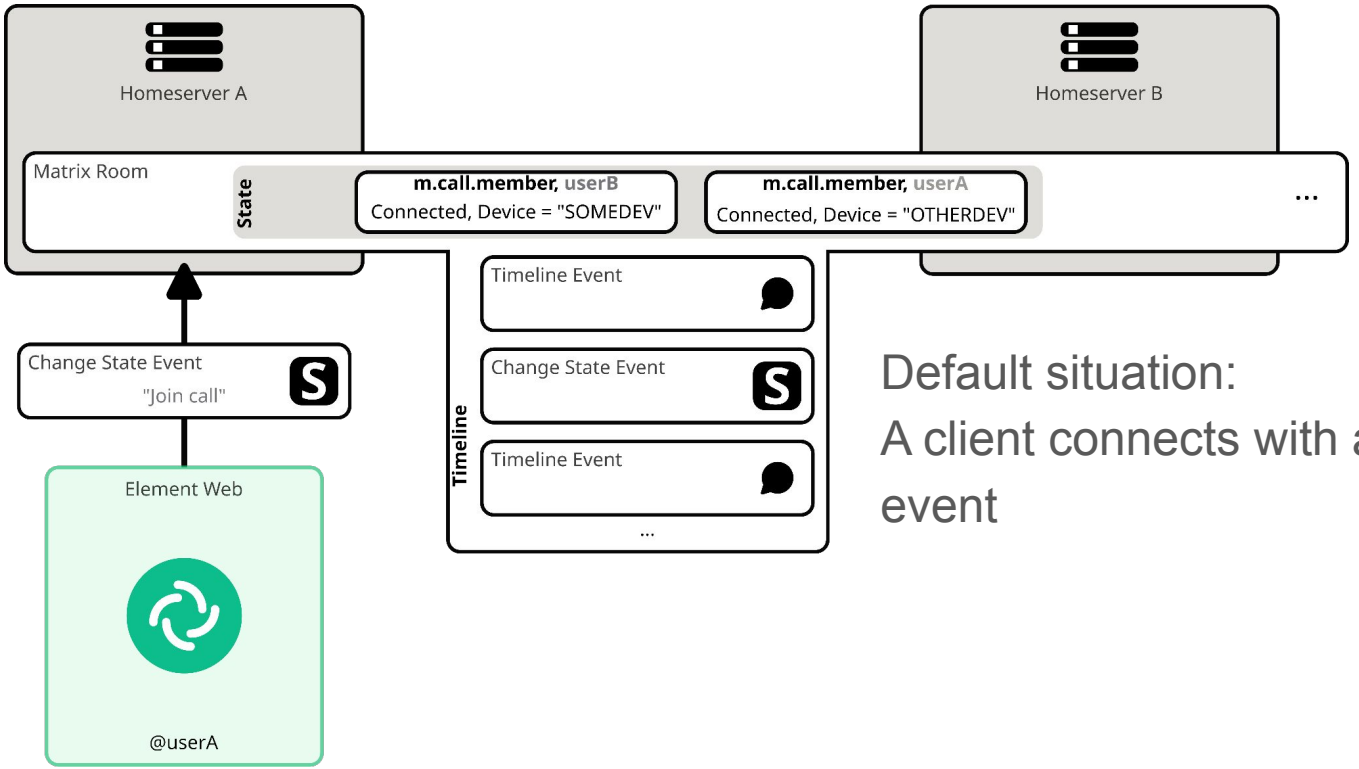
MatrixRTC

```
"content": {
    "device_id": "DEVICEID123",
    "focus_active": {                          Infrastructure
        "type": "full_mesh"
    },
    "foci_preferred":[]
    "application": "m.call",

    "call_id": "",                             Application Specific
    "scope": "m.room",                         Fields defined by m.call

    "linked_event":"$Azt5QD7kRbOq19IyqWoNUtmk6ulGsUSCgKLT6Bvzs-Y",
    "other_shared_or_individual_data":"100,200"

    }
}
"state_key": "@user:matrix.org_DEVICEID"
```

Room State

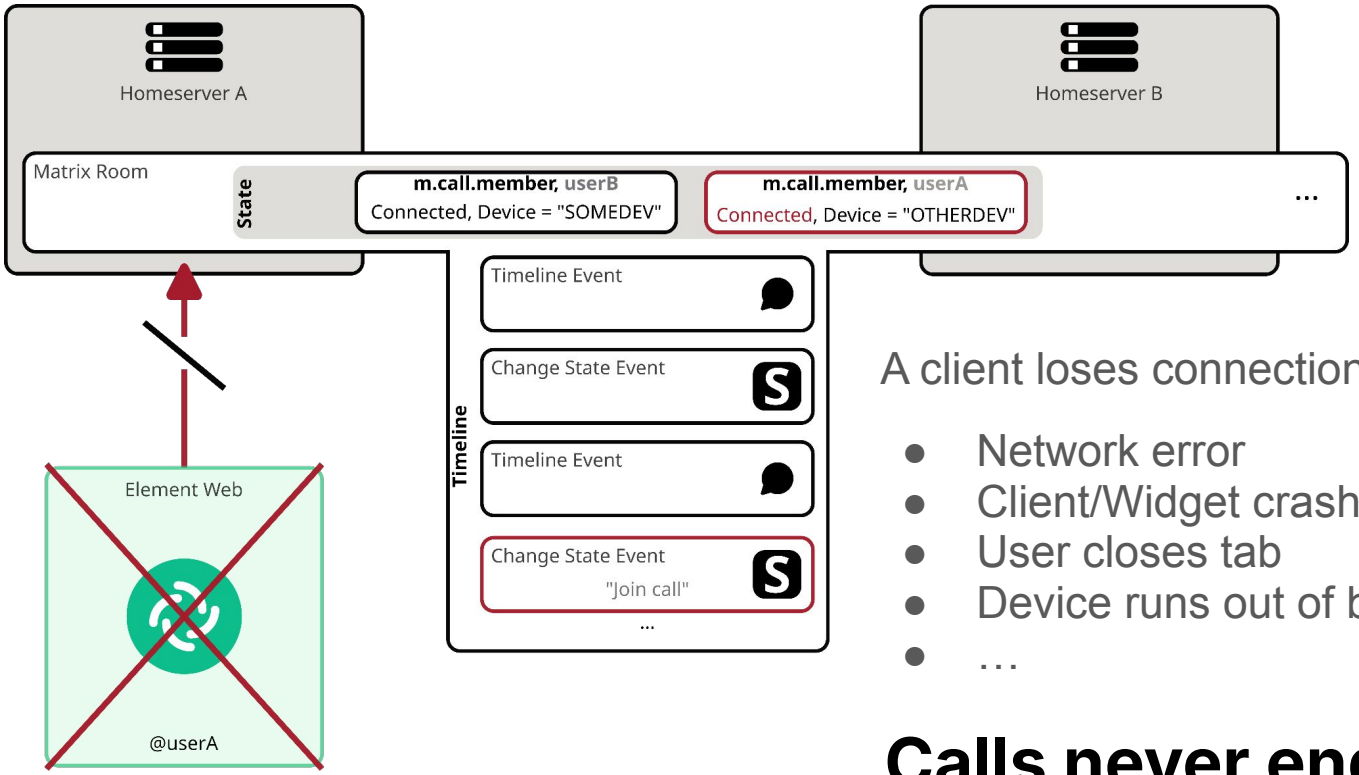
Reliable memberships

Problem



Default situation:
A client connects with a join event

Problem

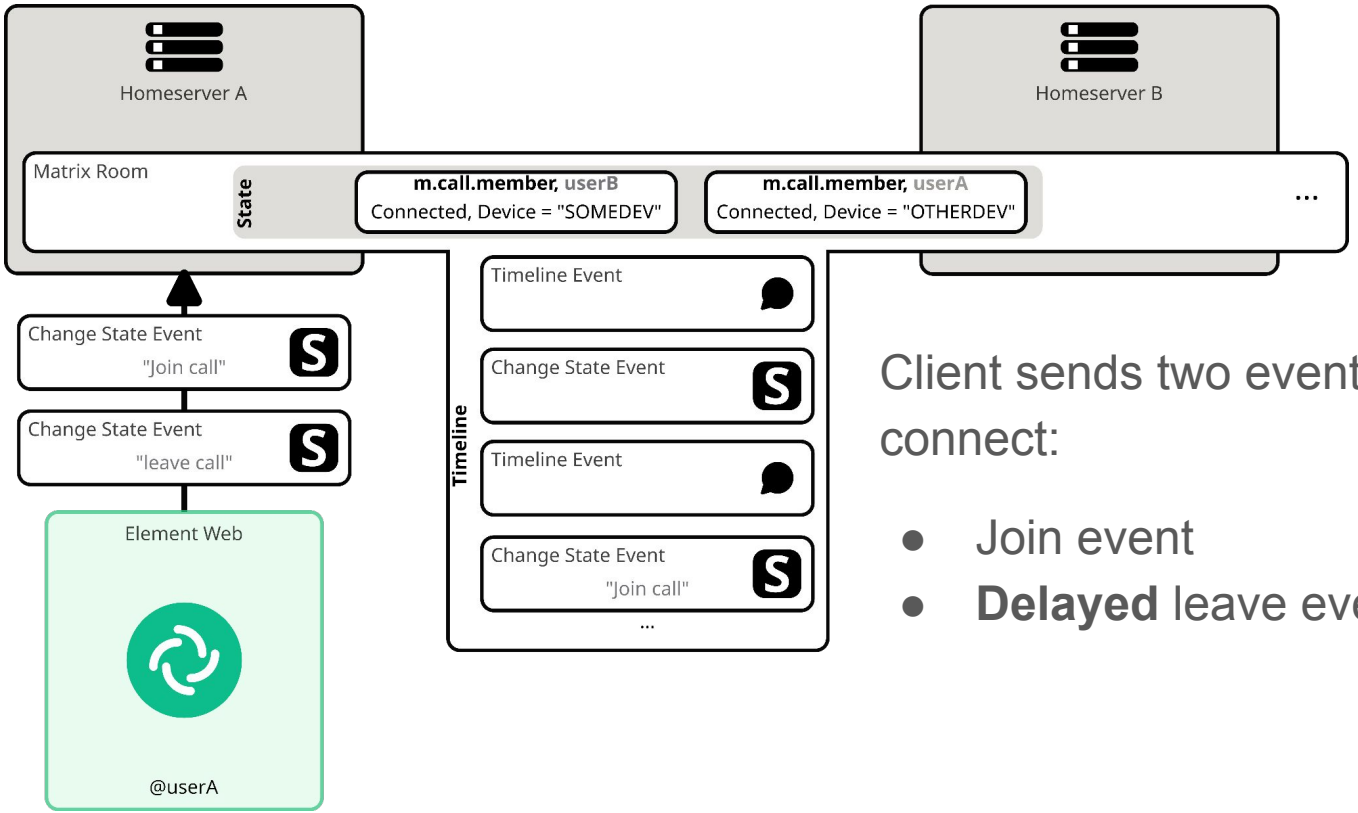


A client loses connection:

- Network error
- Client/Widget crash
- User closes tab
- Device runs out of battery
- …

Calls never end!

Solution “Delayed events”



Client sends two events on connect:

- Join event
- **Delayed** leave event

Solution “Delayed events”

Queue an event on the homeserver.

The queue timeout can be restarted with a new endpoint.

"Last will" via Delayed Events

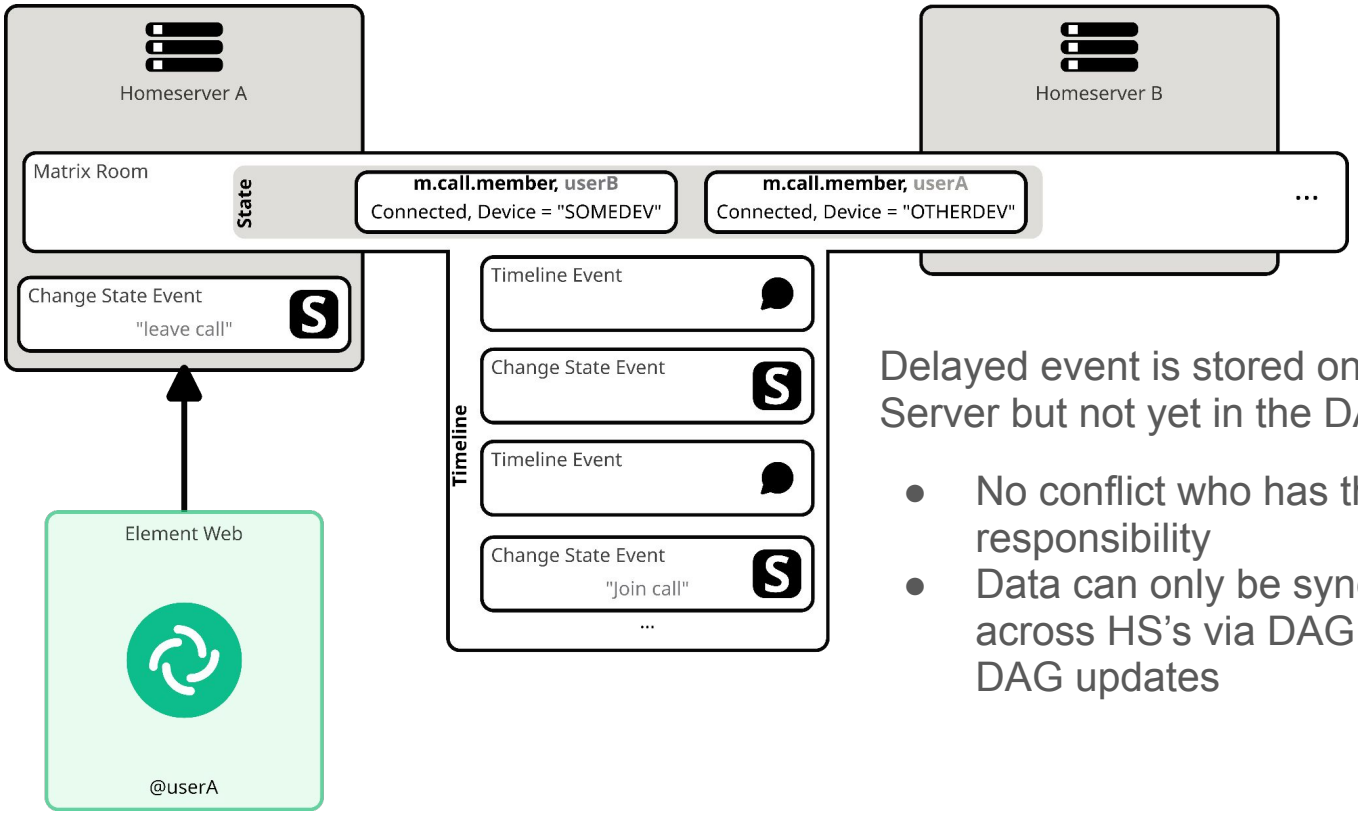
```
PUT /_matrix/client/v3/rooms/{roomId}/state/{eventType}/{stateKey}?delay=5000

{
  "leave_reason": "lost_connection"
}
```

Schedule Delayed Event in 5s

```
POST /_matrix/client/v1/delayed_events/{delay_id}
Content-Type: application/json

{
  "action": "restart"
}
```

Restart the timer called every 3s

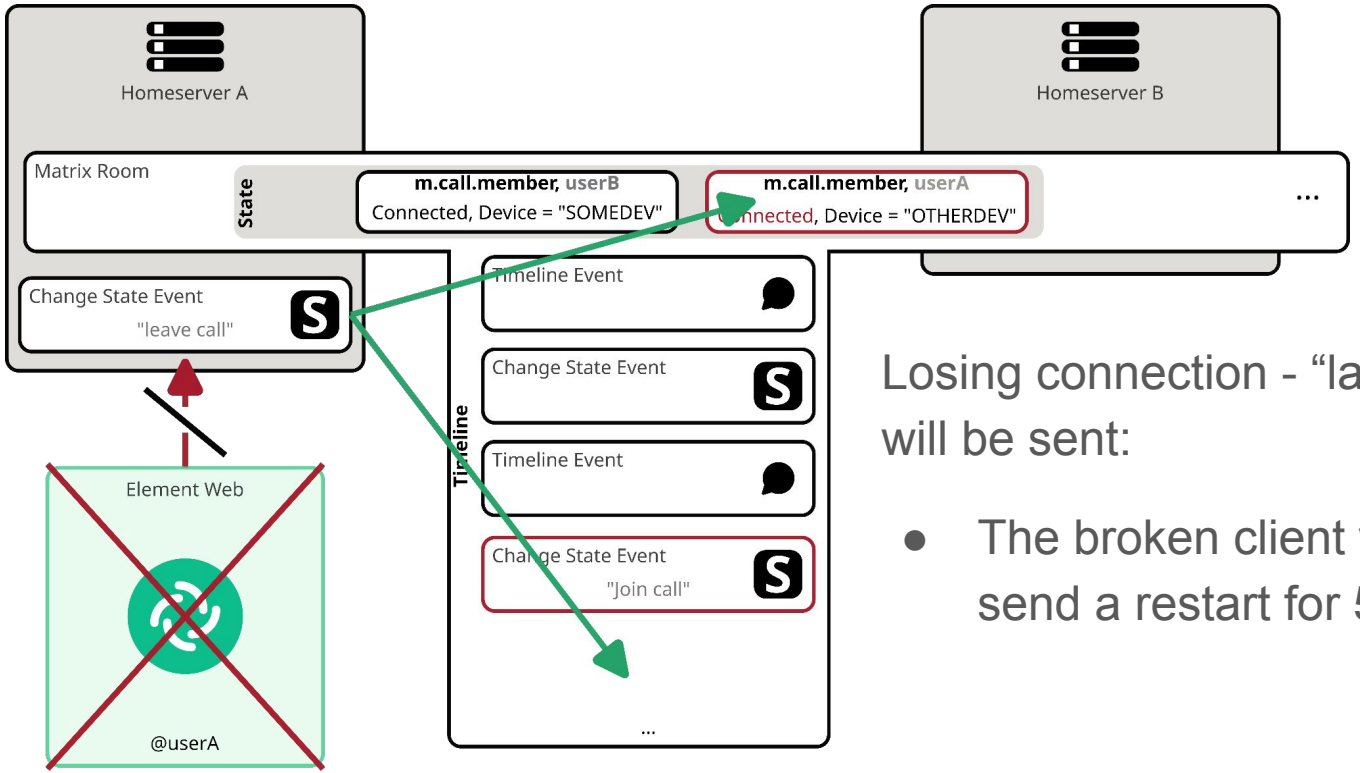
Solution “Delayed events”



Delayed event is stored on the Home Server but not yet in the DAG:

- No conflict who has the timeout responsibility
- Data can only be synced across HS’s via DAG - lots of DAG updates

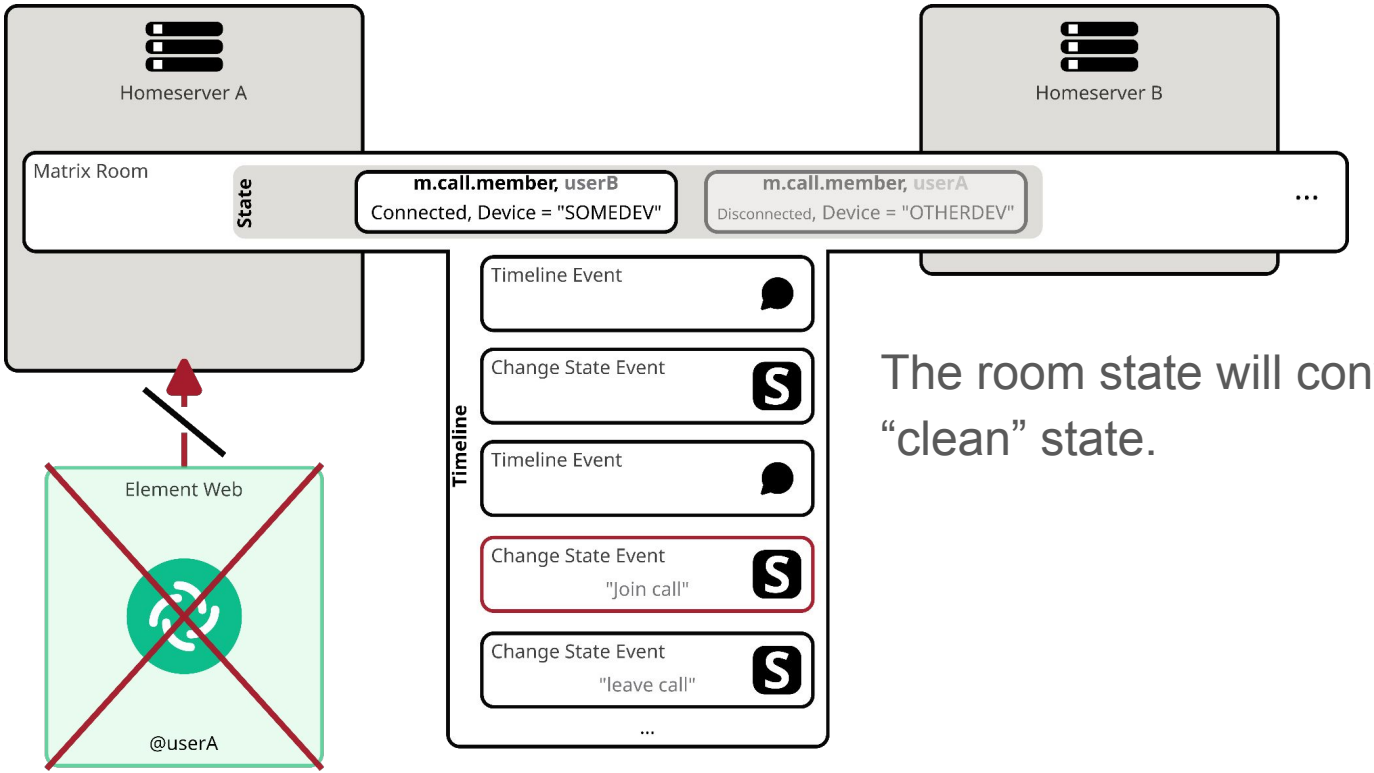
Solution “Delayed events”



Losing connection - “last will” will be sent:

- The broken client will not send a restart for 5s

Solution “Delayed events”



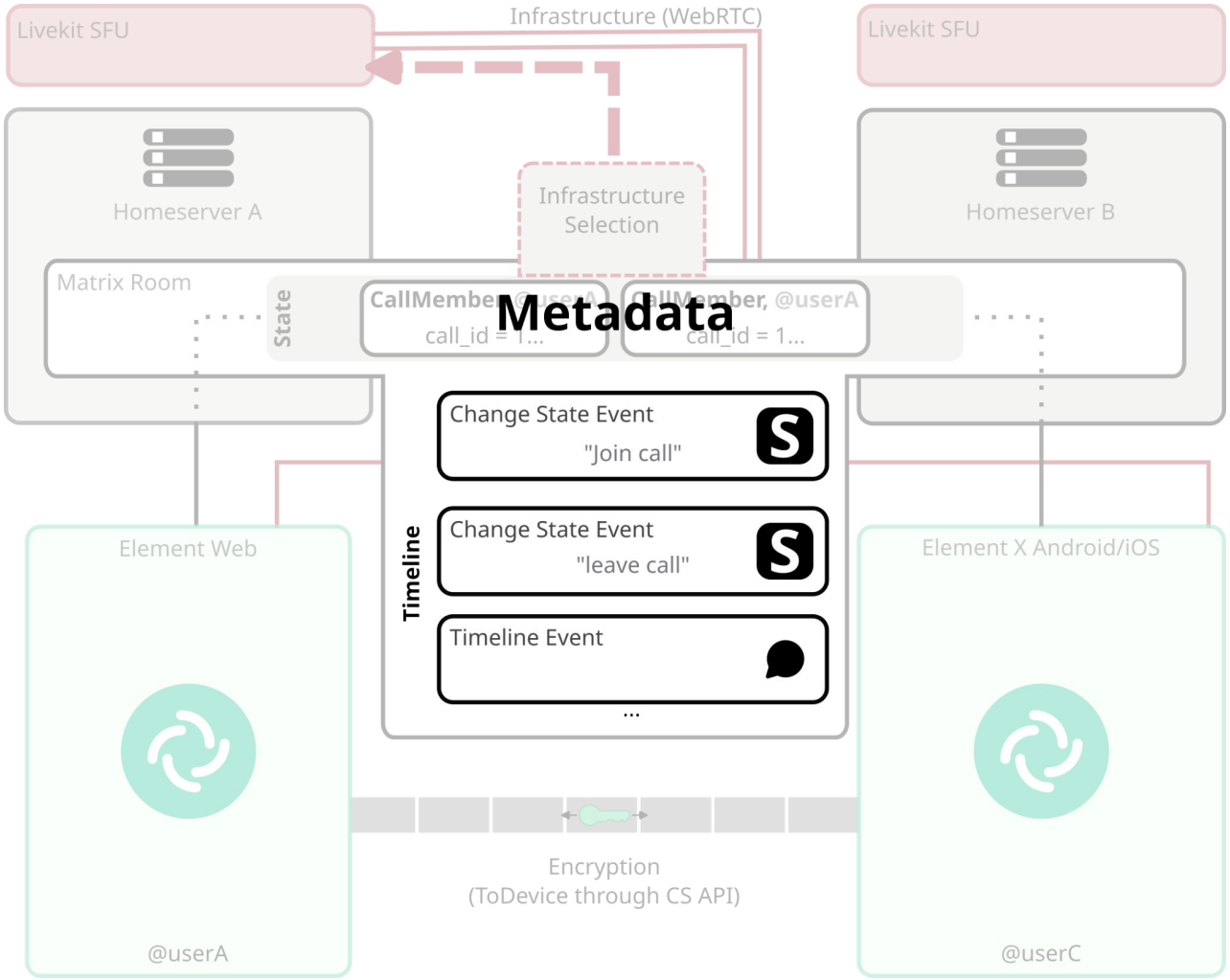
The room state will convert to a “clean” state.

Other possibilities

- Self destructing messages
- Scheduled events
- Tea pot timer :-)
  - m.call.notify
- Temporal room permissions
- …

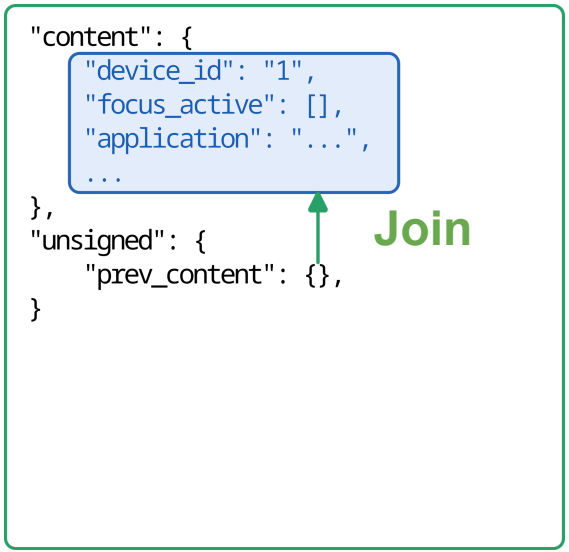
Metadata

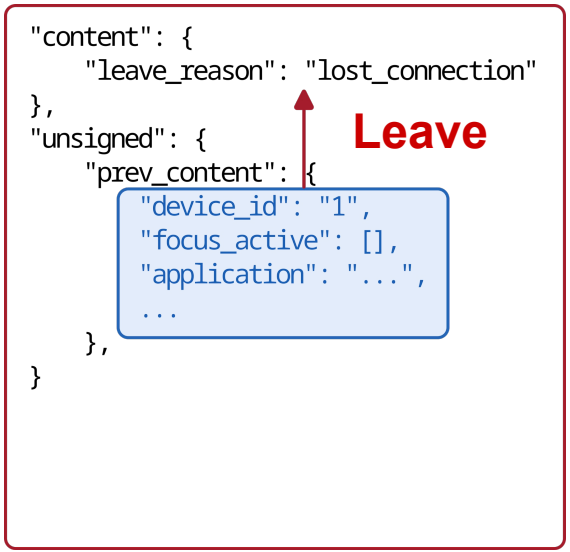
History


Livekit SFU
Infrastructure (WebRTC)
Livekit SFU
Homeserver A
Infrastructure Selection
Homeserver B
Matrix Room
State
CallMember, @userA
call_id = 1...
CallMember, @userA
call_id = 1...
Metadata
Change State Event
"Join call"
Timeline
Change State Event
"leave call"
Timeline Event
...
Element Web
Element X Android/iOS
Encryption
(ToDevice through CS API)
@userA
@userC

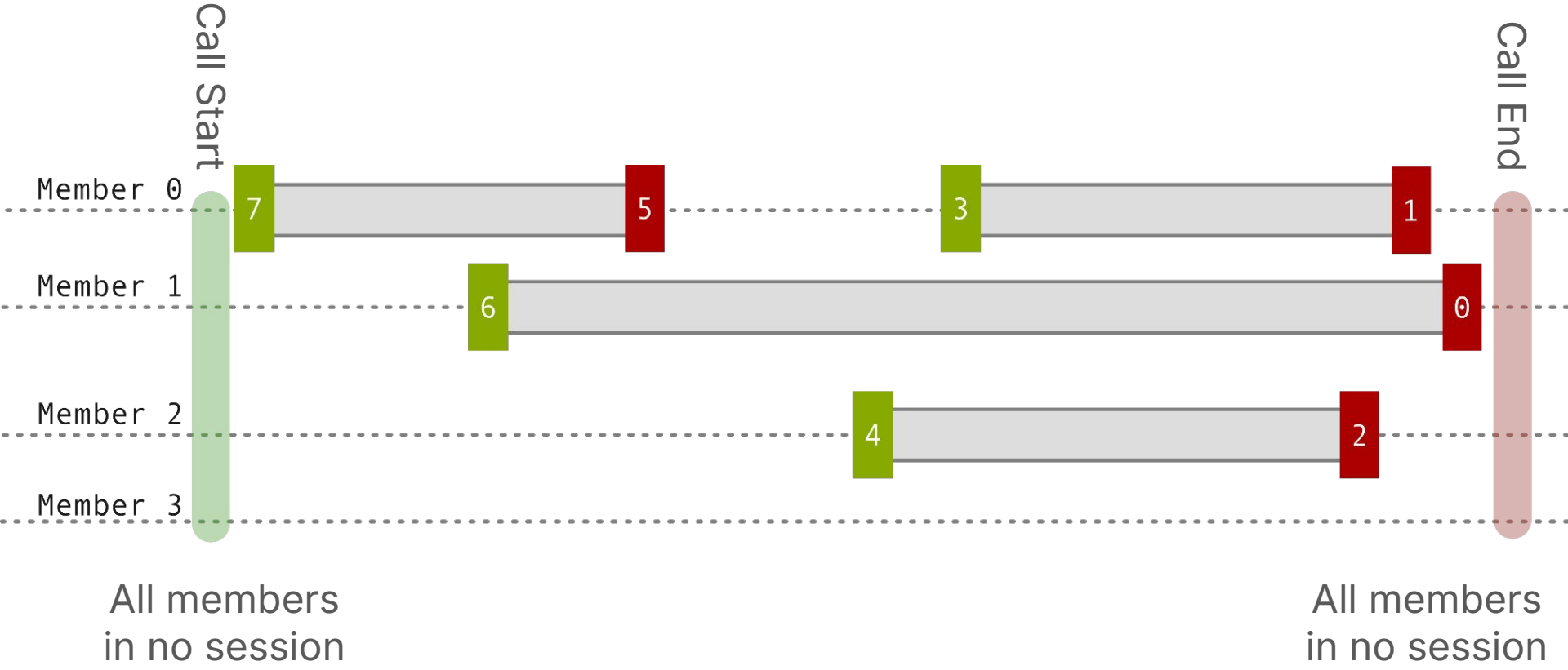
Call History

The m.call.member events can be parsed as **Join** and **Leave** events:





Call History



Metadata

Ringing

Ringing

- Use existing m.mentions
- Can be a simple room event entirely application specific

Call Notify Event

```
"content": {
    "application": "m.call",
    "call_id": "!SomeRoomId:matrix.org",
    "m.mentions": {
        "room": true
    },
    "notify_type": "ring"|"notify"
},
"type": "org.matrix.msc4075.call.notify"
```

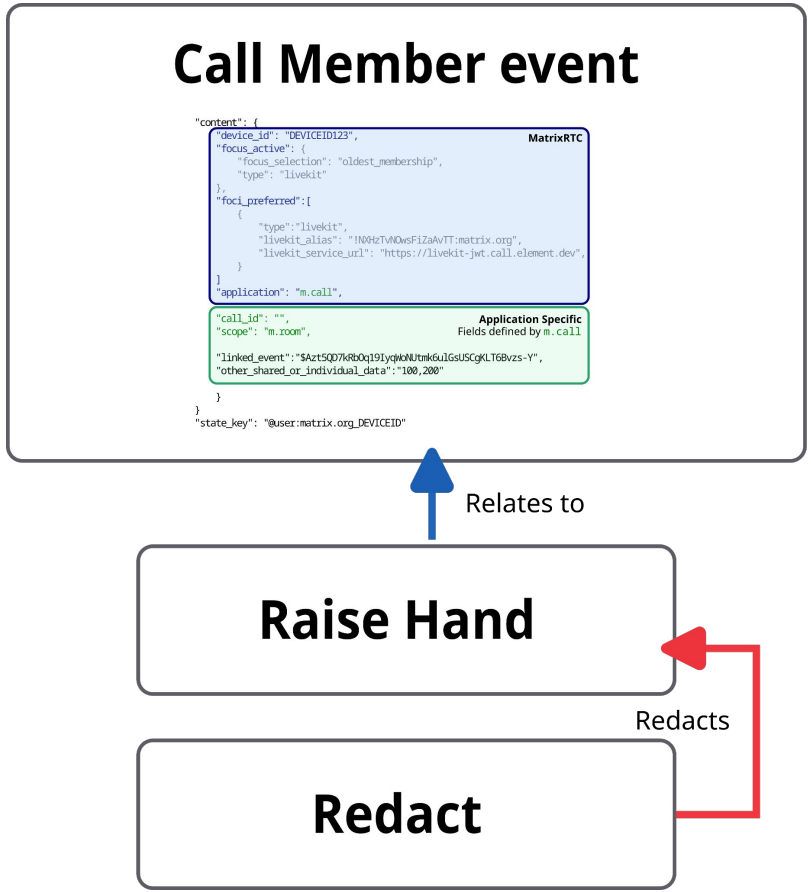
Intentional Mentions

Raise Hand / Emojis

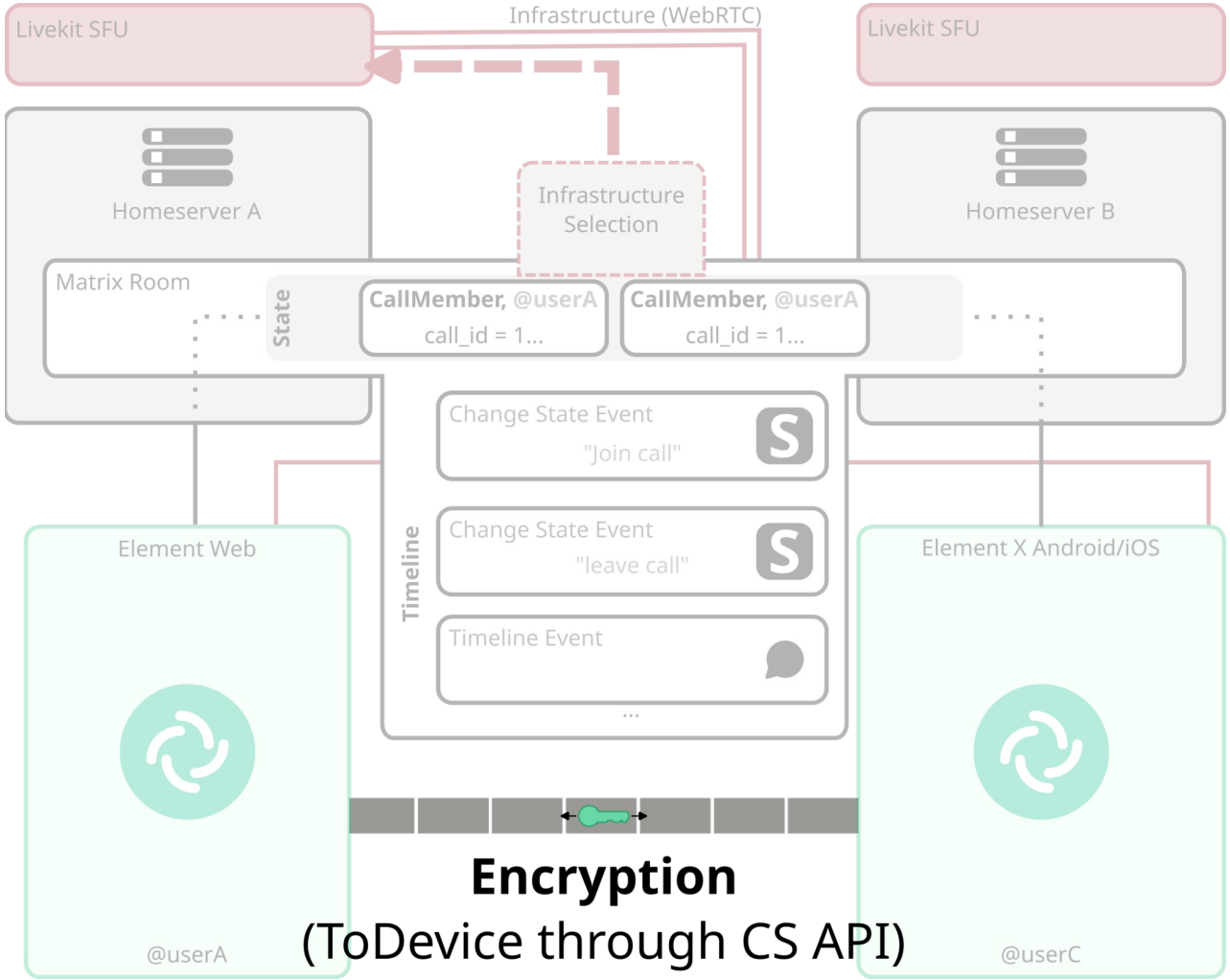
Use Matrix primitives

Load “raise hand state” on join

- relations



Encryption


Livekit SFU
Infrastructure (WebRTC)
Livekit SFU
Homeserver A
Infrastructure Selection
Homeserver B
Matrix Room
State
CallMember, @userA
call_id = 1...
CallMember, @userA
call_id = 1...
Change State Event
"Join call"
Change State Event
"leave call"
Timeline
Timeline Event
...
Element Web
Element X Android/iOS
@userA
@userC
Encryption
(ToDevice through CS API)

What we already have in Matrix

- Megolm
- Device verification
- Secure channels to all participants (Olm, encrypted ToDevice)

Does it fit for MatrixRTC

- Sub-group of a room aka call participants

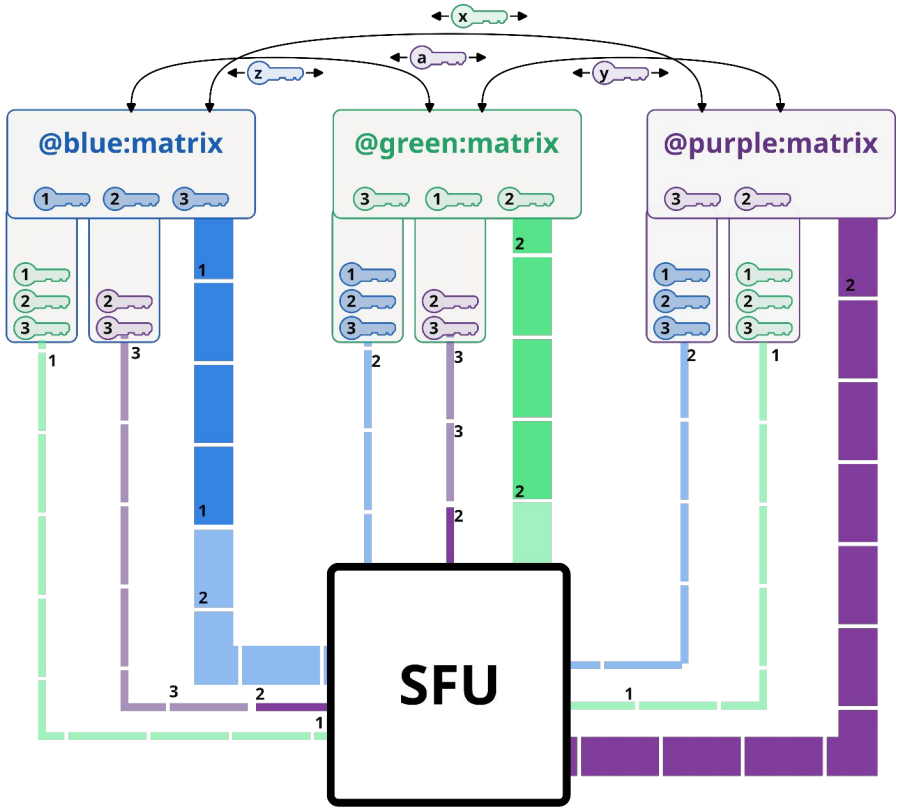
Encryption for WebRTC (SFrame)

- Frame Trailer
- Symmetric keys
- Timing Tradeoff
  - Distributing keys
  - Switching sender keys

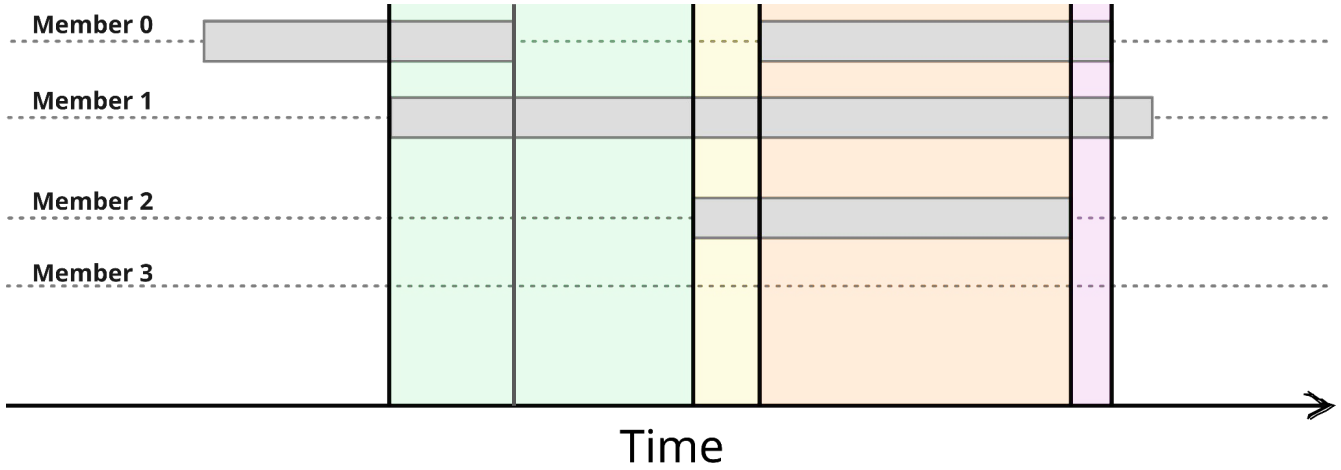
Exchanging Ahead of time

Storing

Using

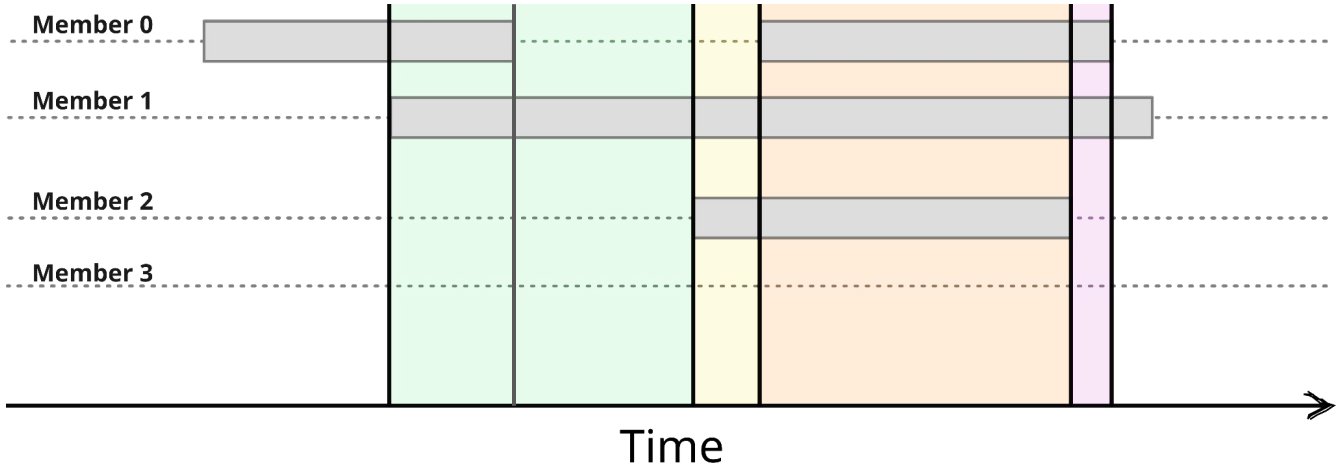


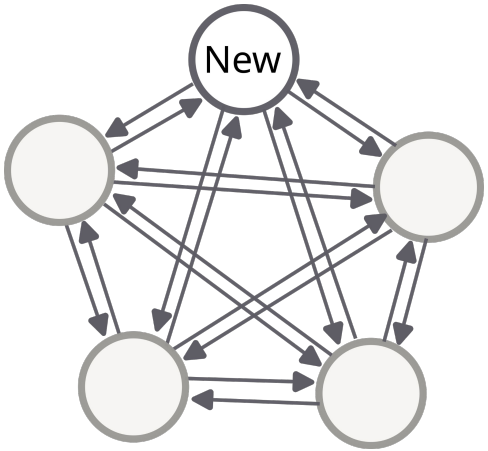
Encryption Join/Leave



- Forward secrecy
- Post compromising

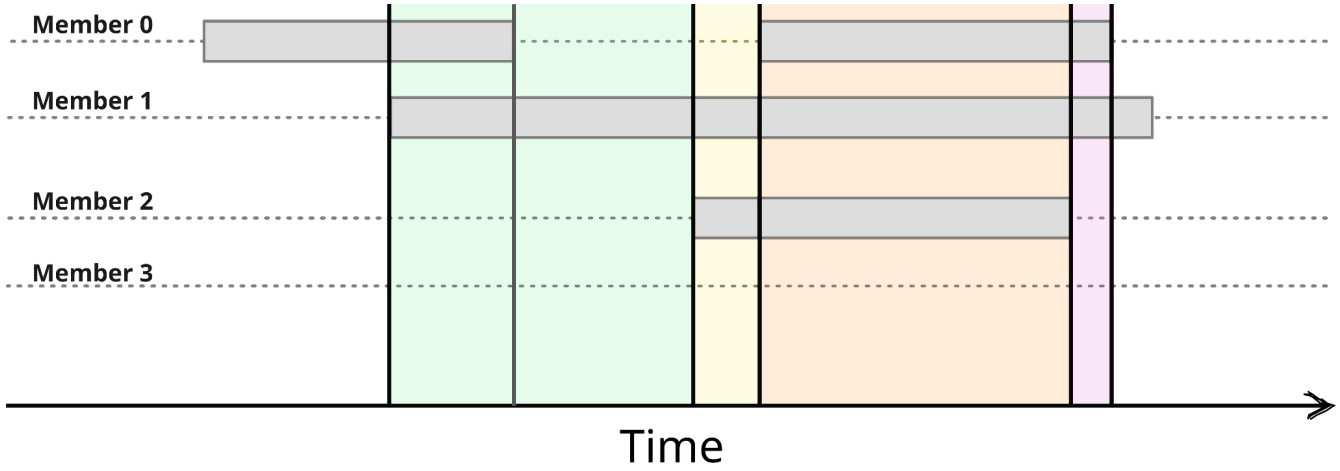
Encryption Full Mesh O(N^2)

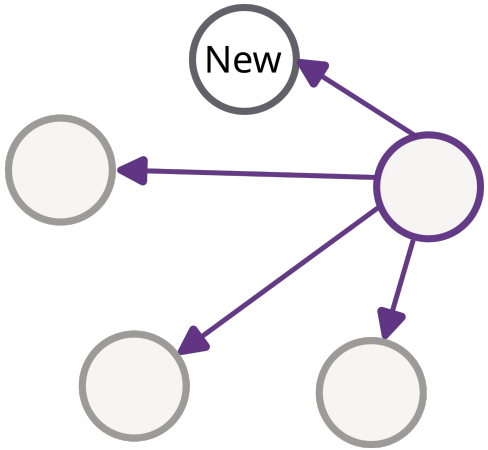




- Forward secrecy
- Post compromising

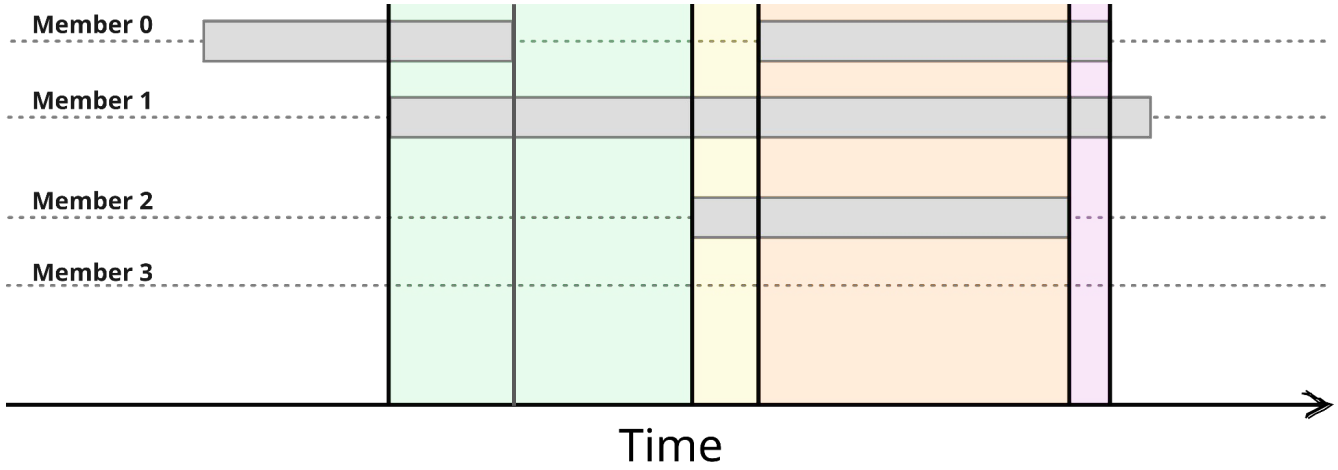
Encryption Shared Key

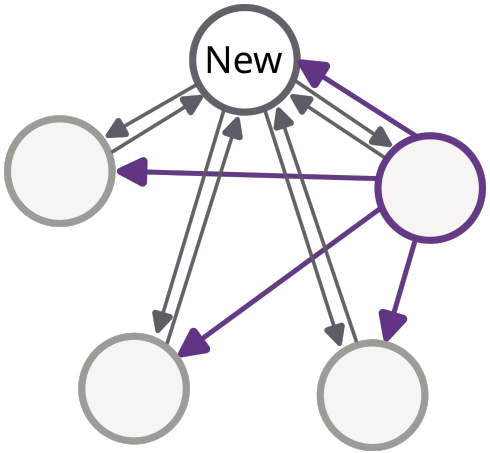




- Forward secrecy
- Post compromising

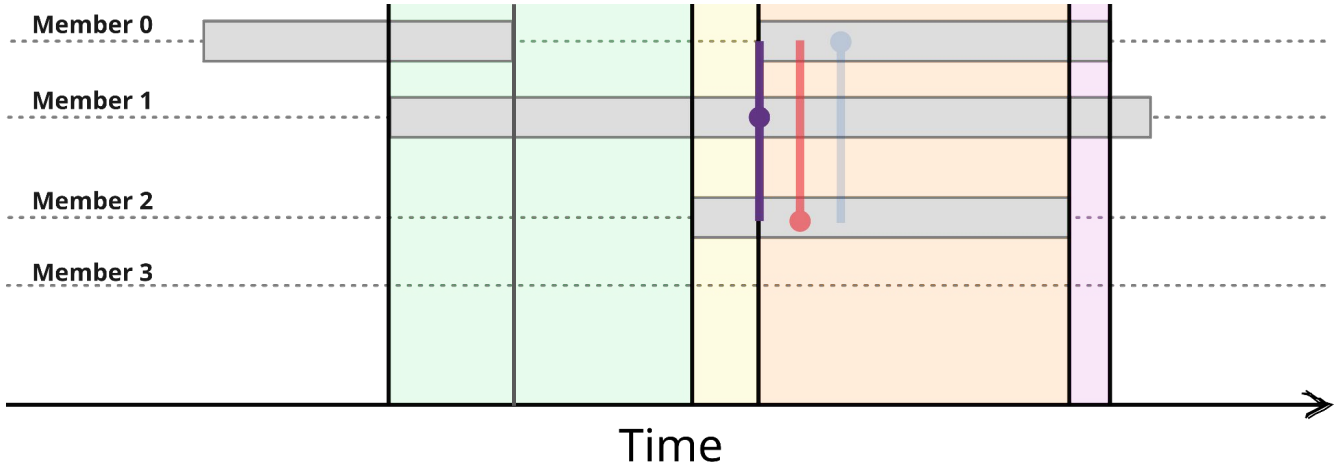
Encryption *per-sender* key + Shared Salt

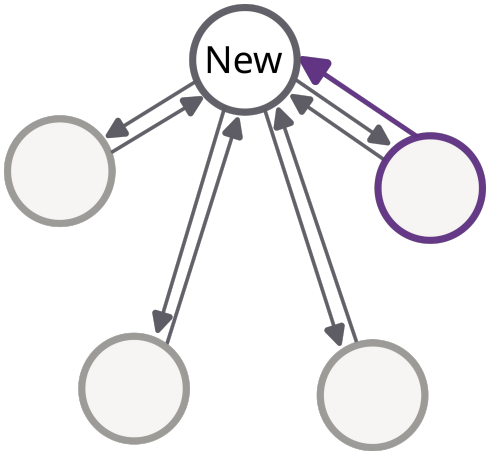




Only share a *per-sender* key on **join**
Update the key using a shared salt

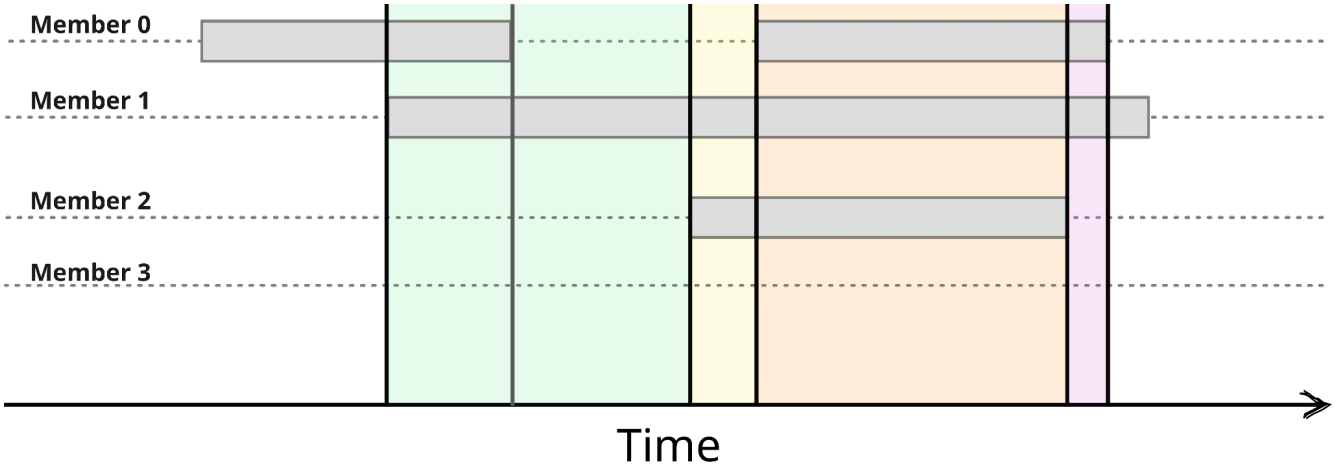
Ownership complexities!





Who sends the shared salt
No-one (broken client, malicious user) **should be able to break the call for everyone**

Encryption Ratcheting? Megolm Subsession? MLS?



- Ratcheting as a possible join (only) optimization
- Existing systems: MLS, Megolm Subsession
- Conditions are great: Leverage on the existing matrix trust and encryption systems

Key takeaways

- **Interchangeable** RTC backend (Livekit, full mesh, ...)
  - Interchangeable algorithms how to find/converge to a backend
- Matrix helps a lot with **encryption**
- Support MatrixRTC is **simple**
  - Client needs to be aware about only one event type: m.rtc.member
    - Show type and user count in a session
  - Supporting (join) a specific application is harder (widgets can help a lot)
- **Extensible** like the rest of Matrix
  - New and different MatrixRTC apps are part of the design

This allows MatrixRTC to be ready now and grow into the best it can be organically

Thank you for listening!

timok@element.io
@toger5:matrix.org