Sliding Sync.next

Stabilising an experimental new sync mechanism

Ivan Enderlin — Matrix Conference 2024

At the beginning...

- Matrix is about communication
- Communication happens inside rooms
- Rooms are listed in a **room list**
- Each room contains a **timeline**, composed of **events**

- We need to **sync** the rooms with their new events, including **e2ee** events
- We want to sync as **fast** as possible


Join me
(alt. 1530m)

Driven by room list features

Filtering and searching

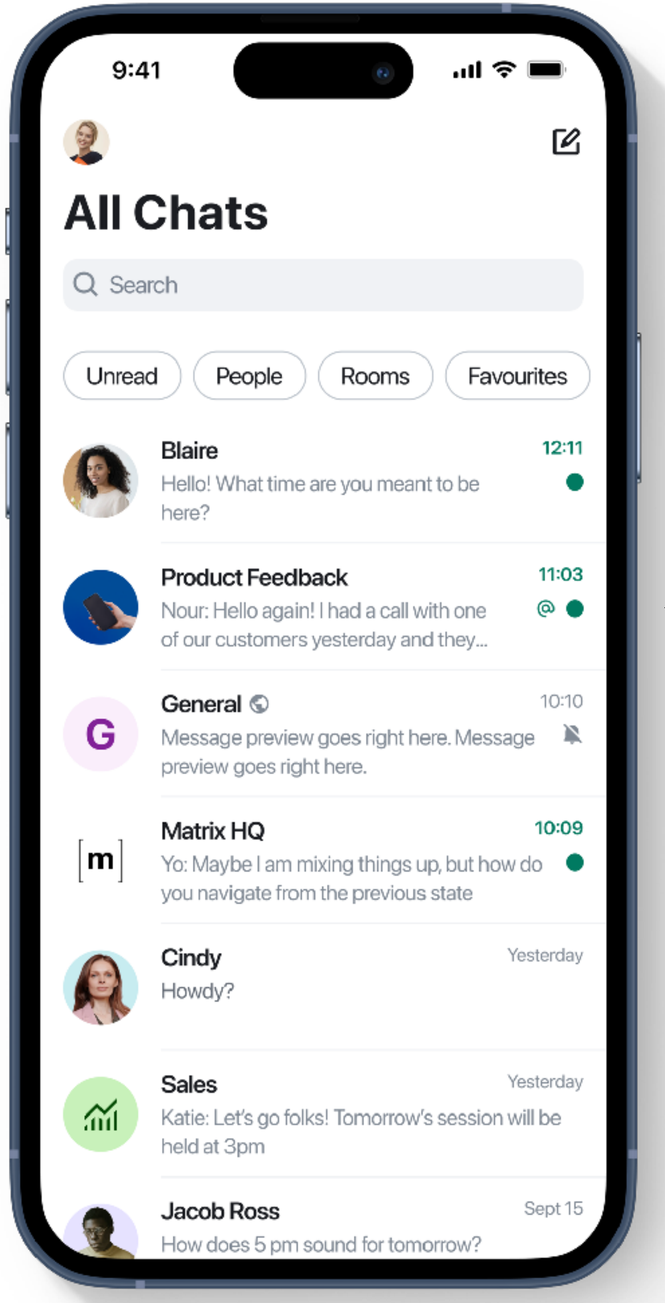
- live, works while syncing

Sorting

- live, works while syncing
- how to sort encrypted rooms?

Room

- preview (may be encrypted)
- name with heroes
- room avatar
- what if the event is not a `m.room.message`?



← Markers

Sliding sync in a nutshell

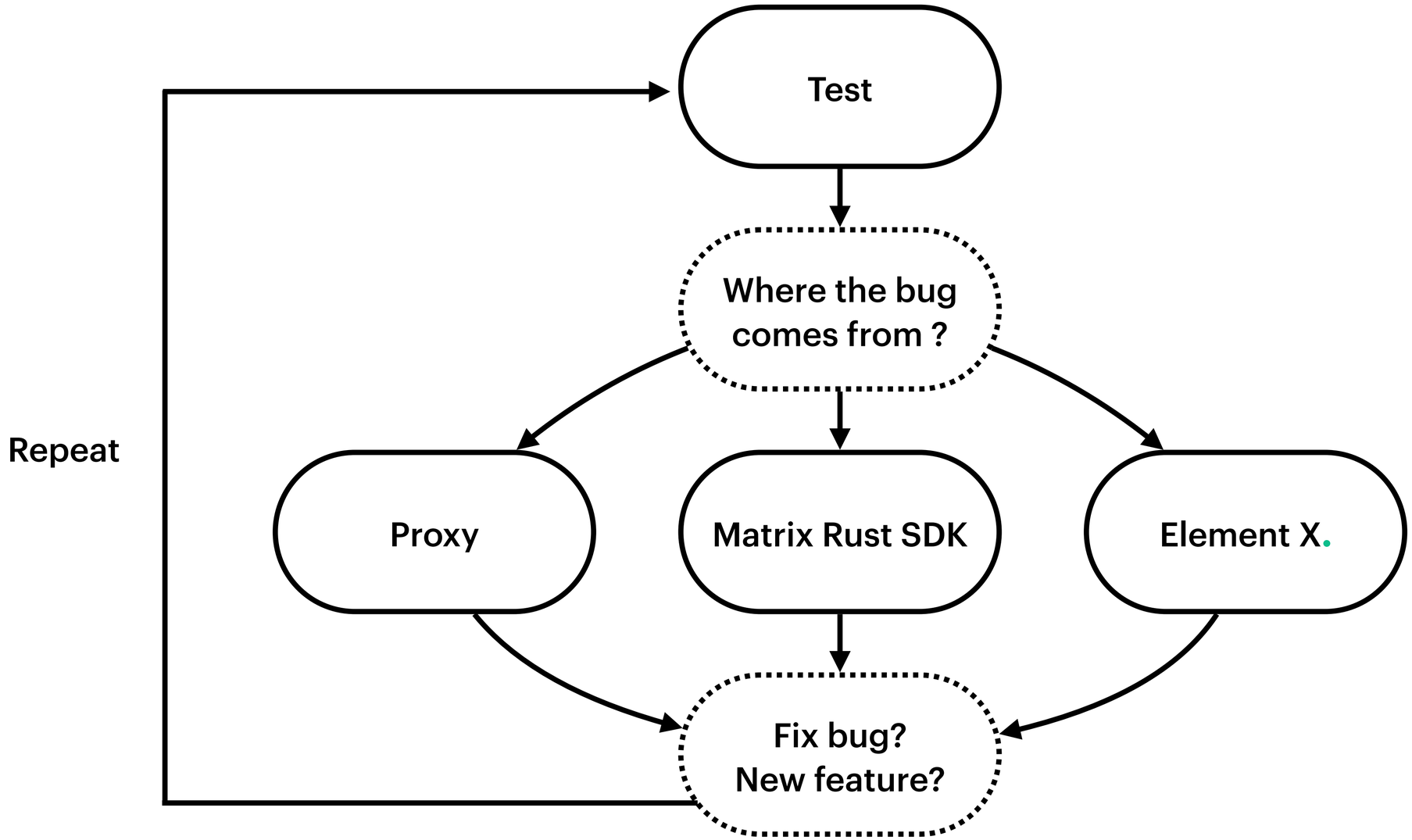
- **Iterative** sync
- Load the room list **page** by **page**
  - Rooms are sorted
  - How to sort rooms if they are encrypted?
- Pre-populate a timeline on-demand
  - How many events do we want?
    - Load more events for rooms that are likely to be opened by the user
  - Can we decrypt the latest event for the room preview?
- Query other data, like e2ee events, account data, read receipts, members...

Enter MSC3575

Sliding sync

- December 20th, 2021, first draft of MSC3575
- First server implementation as a proxy, running on top of Synapse
- First client implementation inside the Matrix Rust SDK
- First app to use it is Element X.
- Let the fun begins
  - One new *server*. One new client. Two new apps. One new sync mechanism.
  - What can go wrong?

The debugging cycle (don't do that)

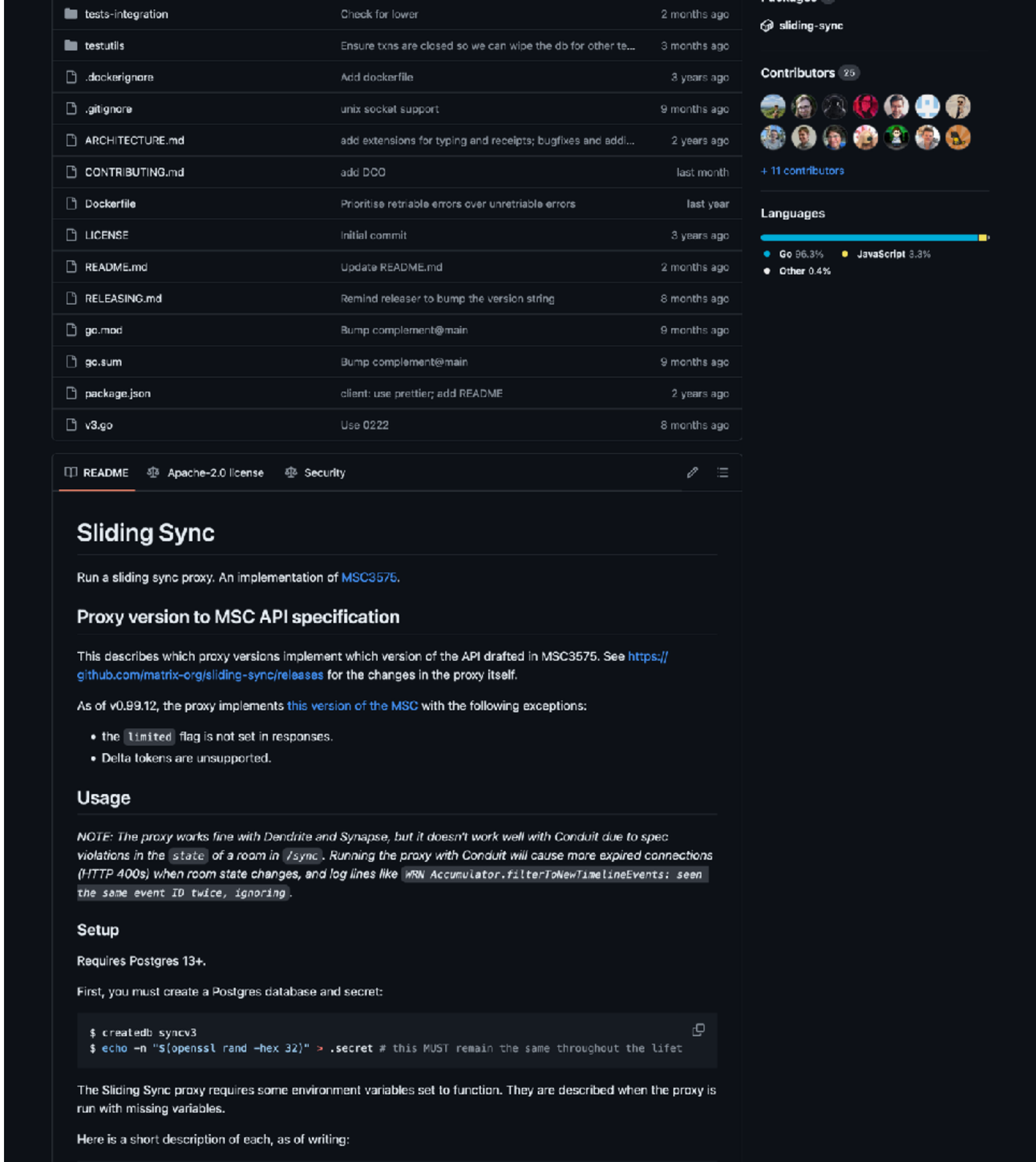


What did we test?

- Accuracy
  - Do we have all the **correct** data we need?
  - Can we select more data when we need to?
- Network performance
  - Bandwidth, request and response sizes
  - How fast is it to recover from an error, or an inactive session?
  - How fast is it to render the room list or a room?
- Does it handle all the e2ee scenarios?

The sliding sync proxy

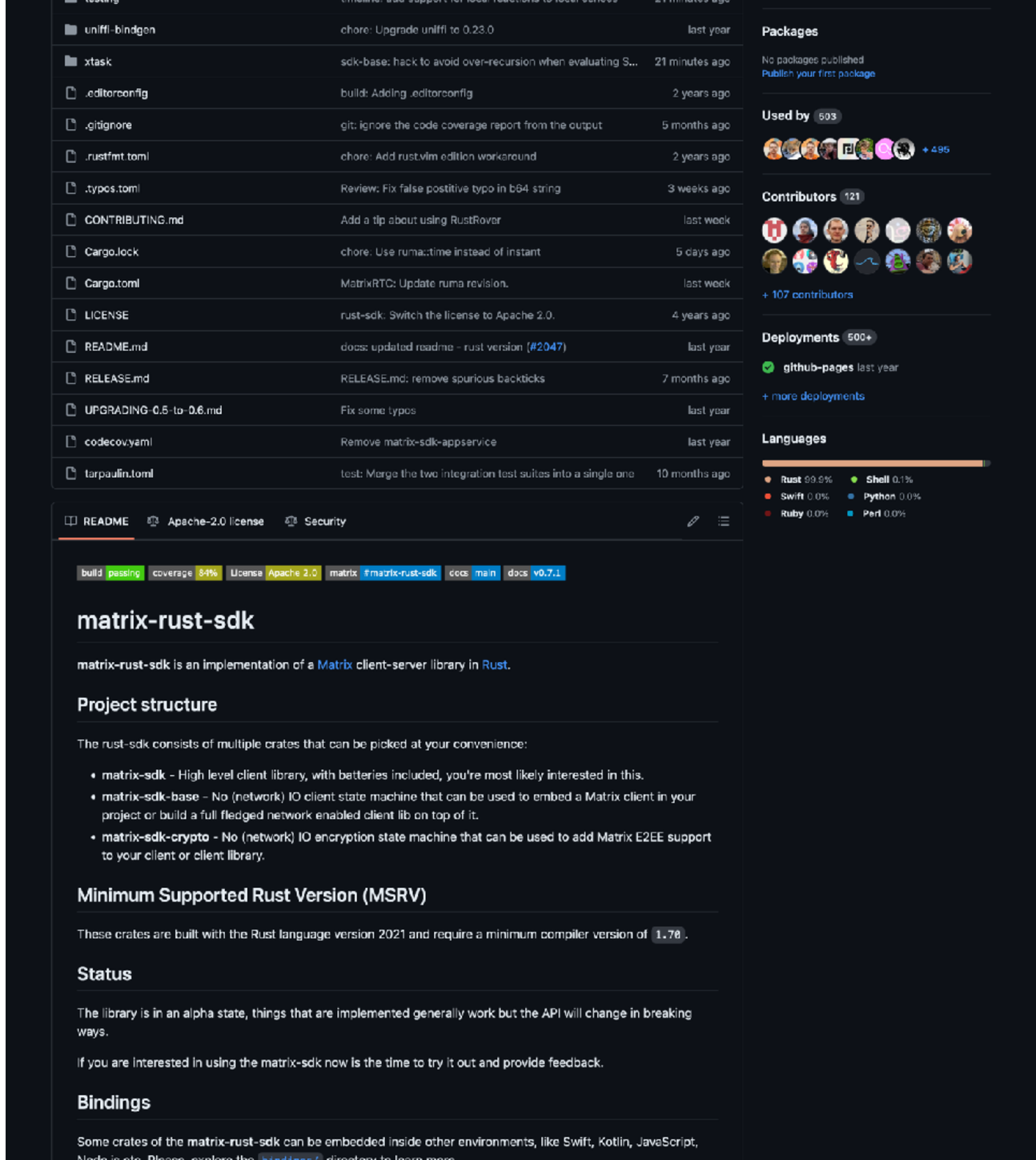
- 1864 commits
- 204 files, 55’349 lines
- Released to the community



tests-integration	Check for lower	2 months ago
testutils	Ensure txns are closed so we can wipe the db for other te...	3 months ago
.dockerignore	Add dockerfile	3 years ago
.gitignore	unix socket support	9 months ago
ARCHITECTURE.md	add extensions for typing and receipts; bugfixes and addi...	2 years ago
CONTRIBUTING.md	add DCO	last month
Dockerfile	Prioritise retriable errors over unretriable errors	last year
LICENSE	Initial commit	3 years ago
README.md	Update README.md	2 months ago
RELEASING.md	Remind releaser to bump the version string	8 months ago
go.mod	Bump complement@main	9 months ago
go.sum	Bump complement@main	9 months ago
package.json	client: use prettier; add README	2 years ago
v3.go	Use 0222	8 months ago

The Matrix Rust SDK

- Introduce a new `SlidingSync` API
- Introduce new APIs built on top of sliding sync
  - `RoomListService`
  - `EncryptionSync`
  - `NotificationClient`
- API have been carefully designed to avoid any sliding sync aspects, i.e. it feels agnostic of any sync mechanisms



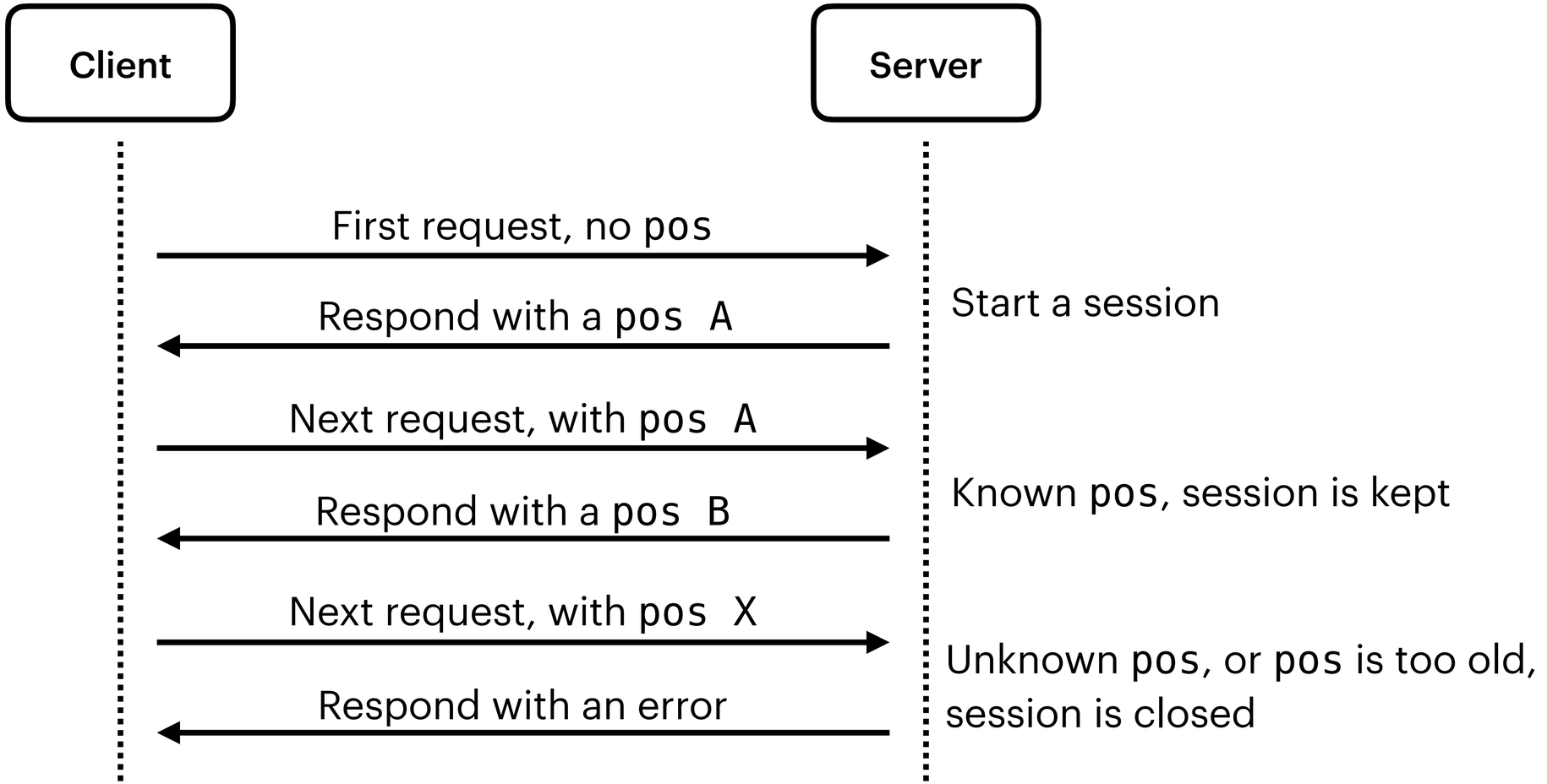
Element X.

- A new Matrix client
- Built entirely on top the Matrix Rust SDK
- Requiring sliding sync

Sliding sync principles

- The server maintains **one or more lists of rooms**, based on the request
- A list is sorted and filtered
- A list is updated by some events
  - Rooms move in a list if one of their event triggers a “bump” (a new sort)
- Each room of a list has a maximum number of events
  - It pre-populates the **timeline**
- The client requests a slice of each list, based on **a range**

Sliding sync sessions



pos is a totally opaque value for the client

HTTP: the request

```
POST /_matrix/client/unstable/org.matrix.msc3575/sync?pos=7

{
    "conn_id": "…",
    "txn_id": "…",
    "lists": { . . . },
    "room_subscriptions": { … },
    "unsubscribe_rooms": { … },
    "extensions": { … }
}
```

HTTP: the request

"lists"

```
"lists": {
    "my_list_name": {
        "ranges": [ [0, 9] ],
        "sort": [ "by_recency", "by_name" ],
        "bump_event_types": [ "m.room.message", … ],
        "timeline_limit": 1,
        "include_heroes": true,
        "filters": [
            "is_tombstoned": false,
            "not_room_types": [ "m.space" ]
        ],
        "required_state": [ [ "m.room.encryption", "" ], … ]
    },
    …
}
```

HTTP: the response

```
{
    "pos": "…",
    "txn_id": "…",
    "initial": …,
    "lists": { . . . },
    "rooms": { . . . },
    "extensions": { … }
}
```

HTTP: the response

"lists"

```
"lists": {
    "my_list_name": {
        "count": 42,
        "ops": [
            {
                "op": "SYNC",
                "range": [0, 3],
                "room_ids": [
                    "!foo:bar",
                    …
                ]
            }
        ]
    }
}
```

HTTP: the response

"rooms"

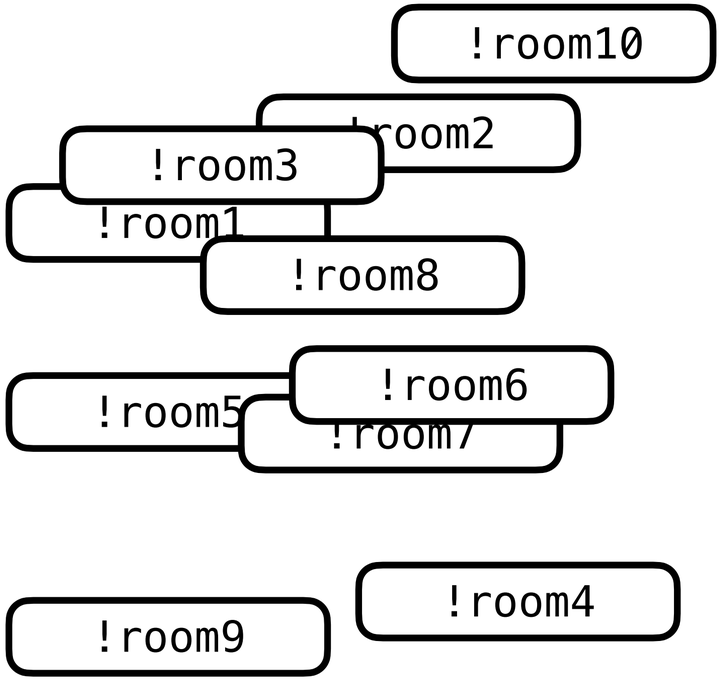
```
"rooms": {
    "!foo:bar": {
        "name": "Alice and Bob",
        "avatar": "mxc://...",
        "initial": true,
        "required_state": [ … ],
        "timeline": [
            {
                "sender": "@alice:qux",
                "type": "m.room.message",
                "content": { "body": "Hello 👋" }
            }
        ],
        "prev_batch": "t111_222_333",
        "joined_count": 1,
        "invited_count": 0,
        "notification_count": 1,
        "highlight_count": 0
    },
    …
}
```

Sliding sync in “action”

Client

Server

Federation



Sliding sync in “action”

Client

request #1:

- `pos = None`
- `range = 0..=4`

Server

`!room1`
`!room2`
`!room3`
`!room4`
`!room5`
`!room6`
`!room7`
`!room8`
`!room9`
`!room10`

Federation

Sliding sync in “action”

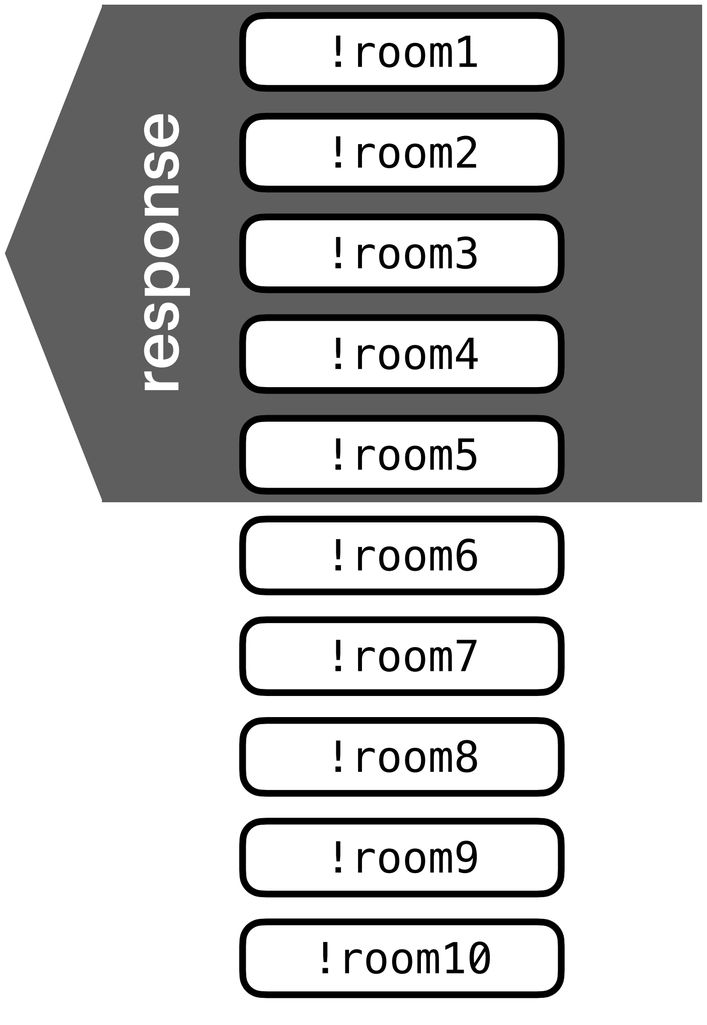
Client

Server

Federation

request #1:

- `pos = None`
- `range = 0..=4`

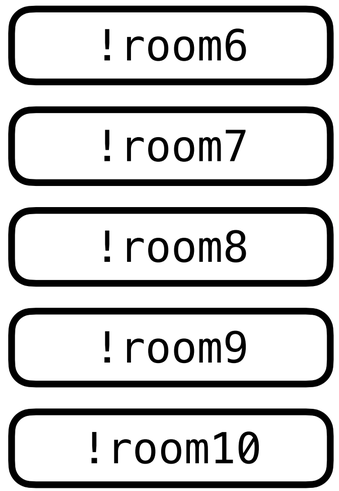


Sliding sync in “action”

Client | Server | Federation

request #1:

- `pos = None`
- `range = 0..=4`

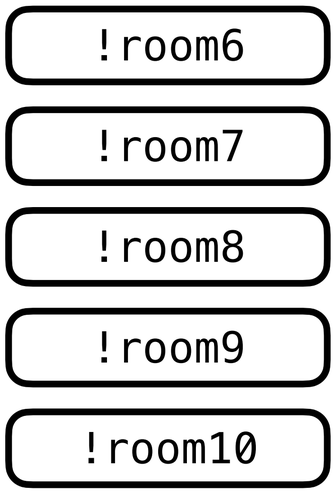


Sliding sync in “action”

Client | Server | Federation

request #2:

- `pos = A`
- `range = 0..=4`



Sliding sync in “action”

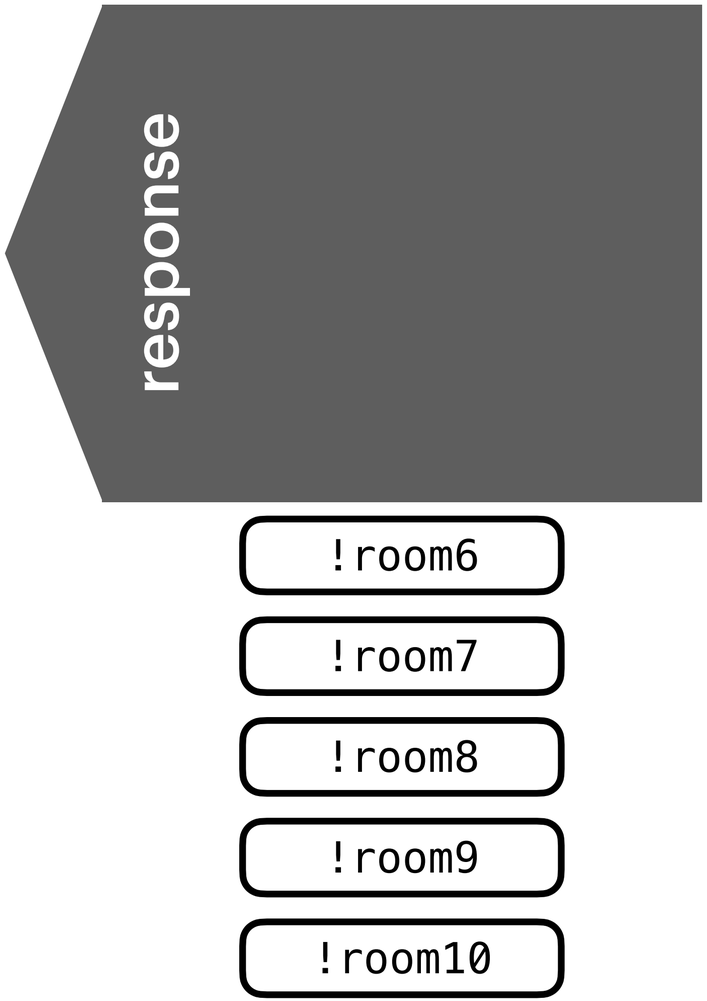
Client

Server

Federation

request #2:

- `pos = A`
- `range = 0..=4`

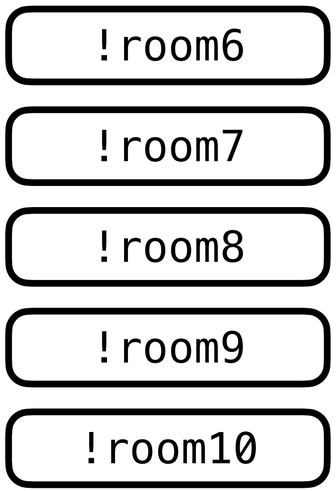


Sliding sync in “action”

Client | Server | Federation

request #2:

- `pos = A`
- `range = 0..=4`

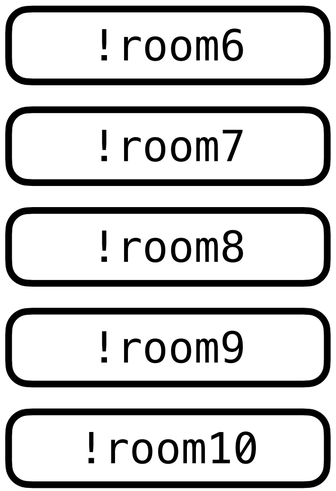


Sliding sync in “action”

Client | Server | Federation

request #3:

- `pos = B`
- `range = 0..=9`



Sliding sync in “action”

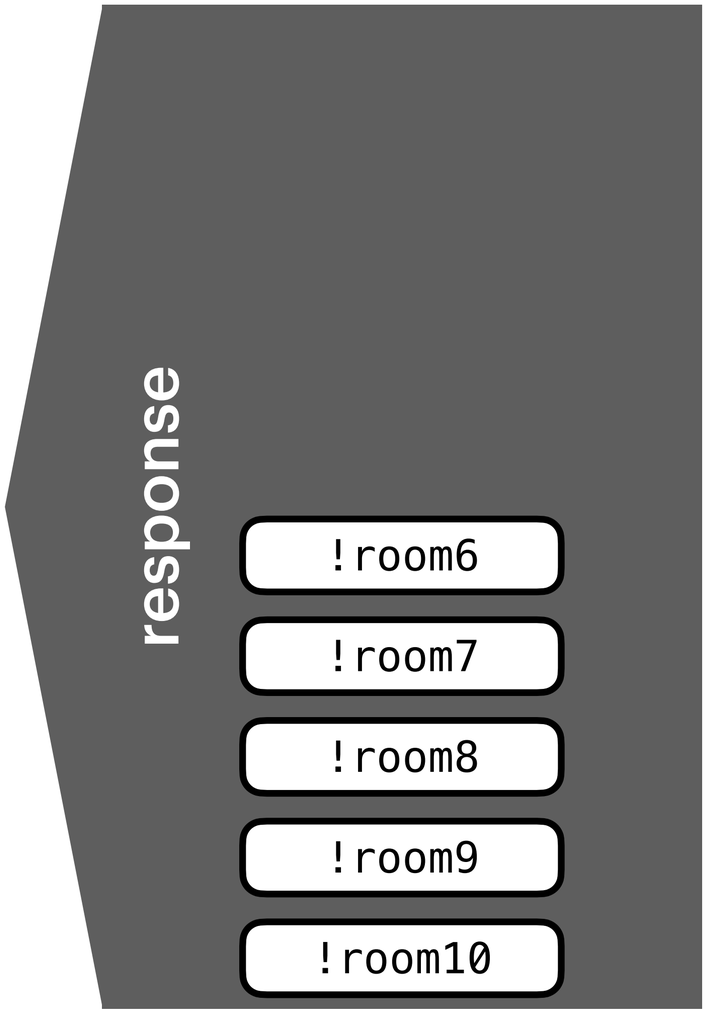
Client

Server

Federation

request #3:

- `pos = B`
- `range = 0..=9`



Sliding sync in “action”

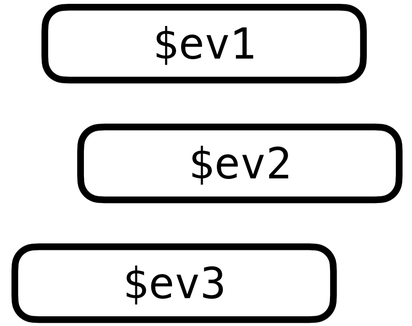
Client

Server

Federation

request #3:

- `pos = B`
- `range = 0..=9`

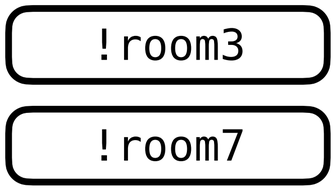


Sliding sync in “action”

Client

Server

Federation



Sliding sync in “action”

Client

request #4:

- `pos = C`
- `range = 0..=9`

Server

`!room3`

`!room7`

Federation

Sliding sync in “action”

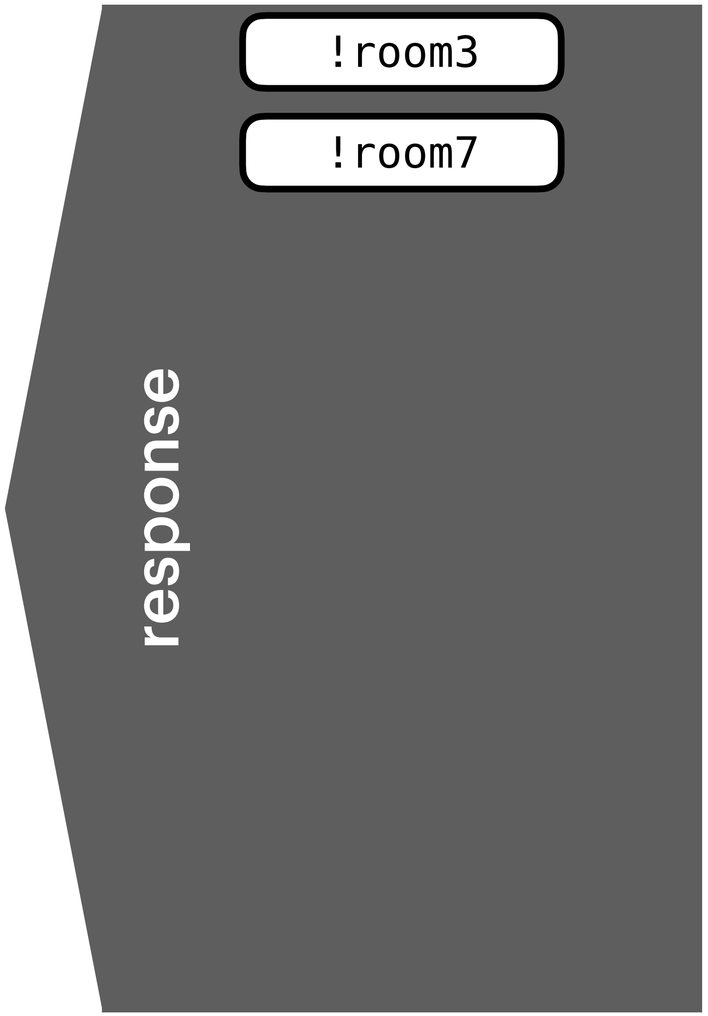
Client

Server

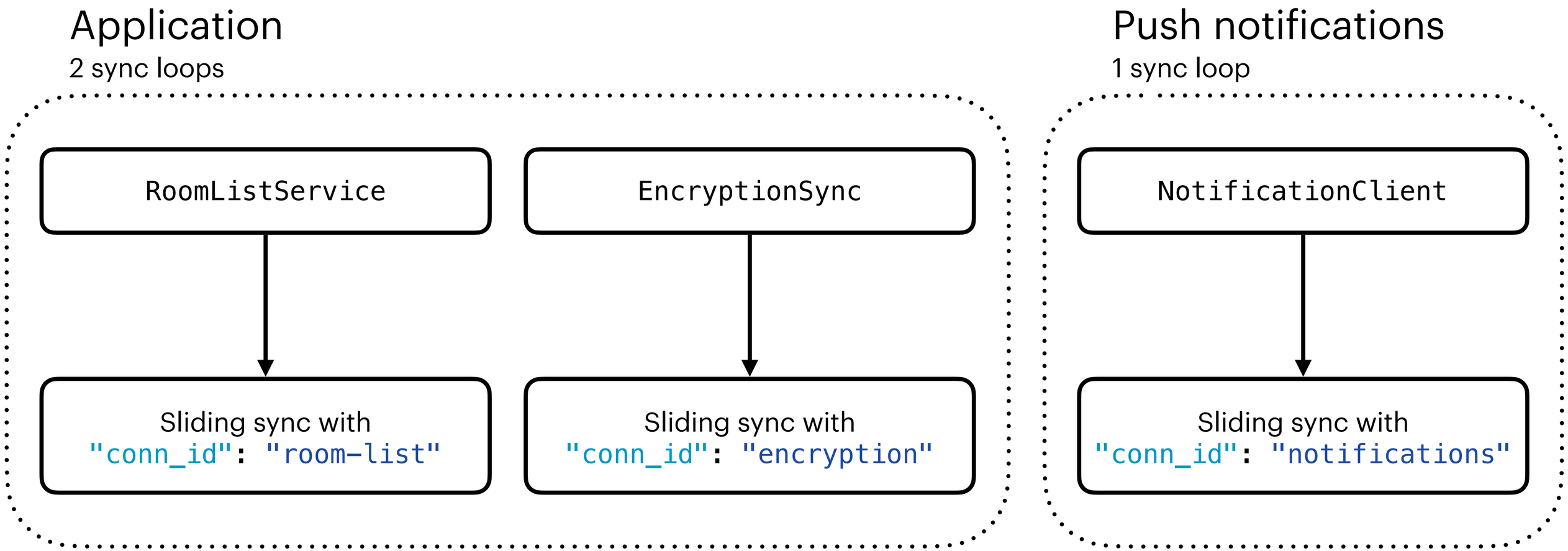
Federation

request #4:

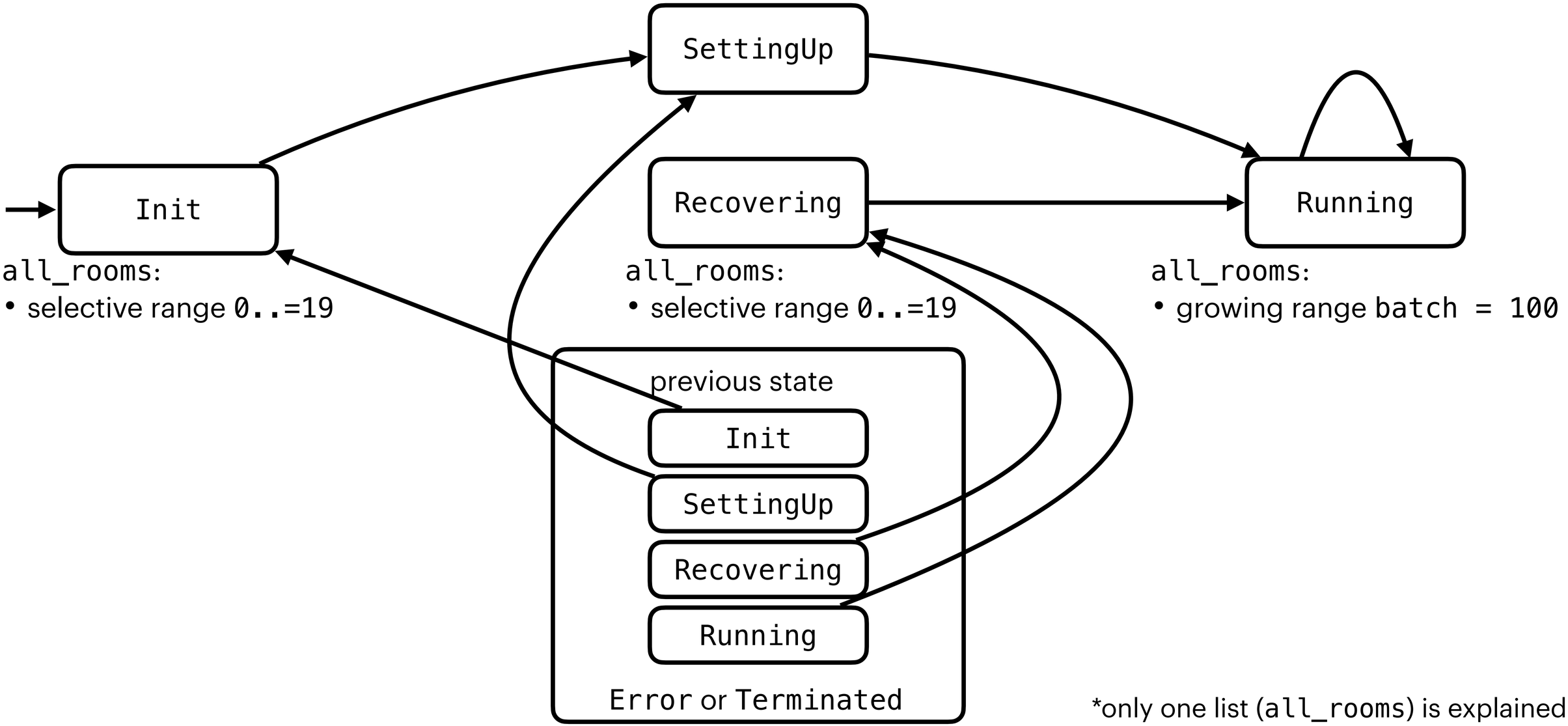
- `pos = C`
- `range = 0..=9`



Concurrent sliding syncs inside the client



The syncing room list state machine



*only one list (`all_rooms`) is explained

Client ranges API

Selective range:

- always the same range

Example with `size = 20`:

- request #1: `0..=19`
- request #2: `0..=19`
- request #3: `0..=19`

Growing range:

- starts from `0`
- adds `batch_size` every time

Example with `batch_size = 100`:

- request #1: `0..=99`
- request #2: `0..=199`
- request #3: `0..=299`


Taking a break
(alt. 2740m)

Is sliding sync working?

- After almost 2 years of iteration, an efficient pattern is emerging
- It’s working
- It’s fast
- It’s really fast (within 3 order of magnitude compared to sync v2)
- But the proxy is too costly to run
- And there is bugs… like… complex bugs
- Bugs hide in features we *may* not need
- Now we know what to do: might we **simplify sliding sync**?


The objective

Removing filters

- Filtering the room list must be instantaneous
- **Filtering moves on client-side**
- Remove most filters from the request:

DEL `is_dm`

DEL `spaces`

DEL `is_encrypted`

- `is_invite`

DEL `is_tombstoned`

DEL `room_types`

- `not_room_types`

DEL `room_name_like`

DEL `tags`

DEL `not_tags`

DEL others

```
let (stream, controller) = room_list.entries_with_dynamic_adapters(…);
controller.set_filter(Box::new(new_filter_all(vec![
    Box::new(new_filter_unread()),
    Box::new(new_filter_favourite()),
    Box::new(new_filter_fuzzy_match_room_name("matco")),
])));
```

Removing sorting operations

- Sorting happened on the server-side, via `ops`
  - The server was maintaining the lists and was sending the “diff”s
    - `SYNC` or `INVALIDATE`: Insert or remove a range of rooms
    - `INSERT` or `DELETE`: insert or delete a single room
  - Super complex, incorrect by nature, and costly for the server = error-prone and inefficient
- Remove `ops` entirely
- **Sorting moves client-side**

```
stream.sort_by(new_sorter_lexicographic(vec![
    Box::new(new_sorter_recency()),
    Box::new(new_sorter_name())
]))
```

Other removals

A detailed list for the implementors

- DEL `bump_event_types` is now hard-coded on the server-side
- DEL `delta_token` was never implemented
- DEL `slow_get_all_rooms` can be replaced by a growing range
- DEL `include_old_rooms` is hard-coded on the server-side
- DEL `unsubscribe_rooms` is no more possible (can only subscribe)

End of the experiment

- The proxy aimed at iterating and hacking quickly for the experiment
- At this point, we know exactly what we want
  - Sliding sync is more stable
  - Plus, we don’t want to keep the proxy on top of Synapse
- **Let’s deprecate the proxy**

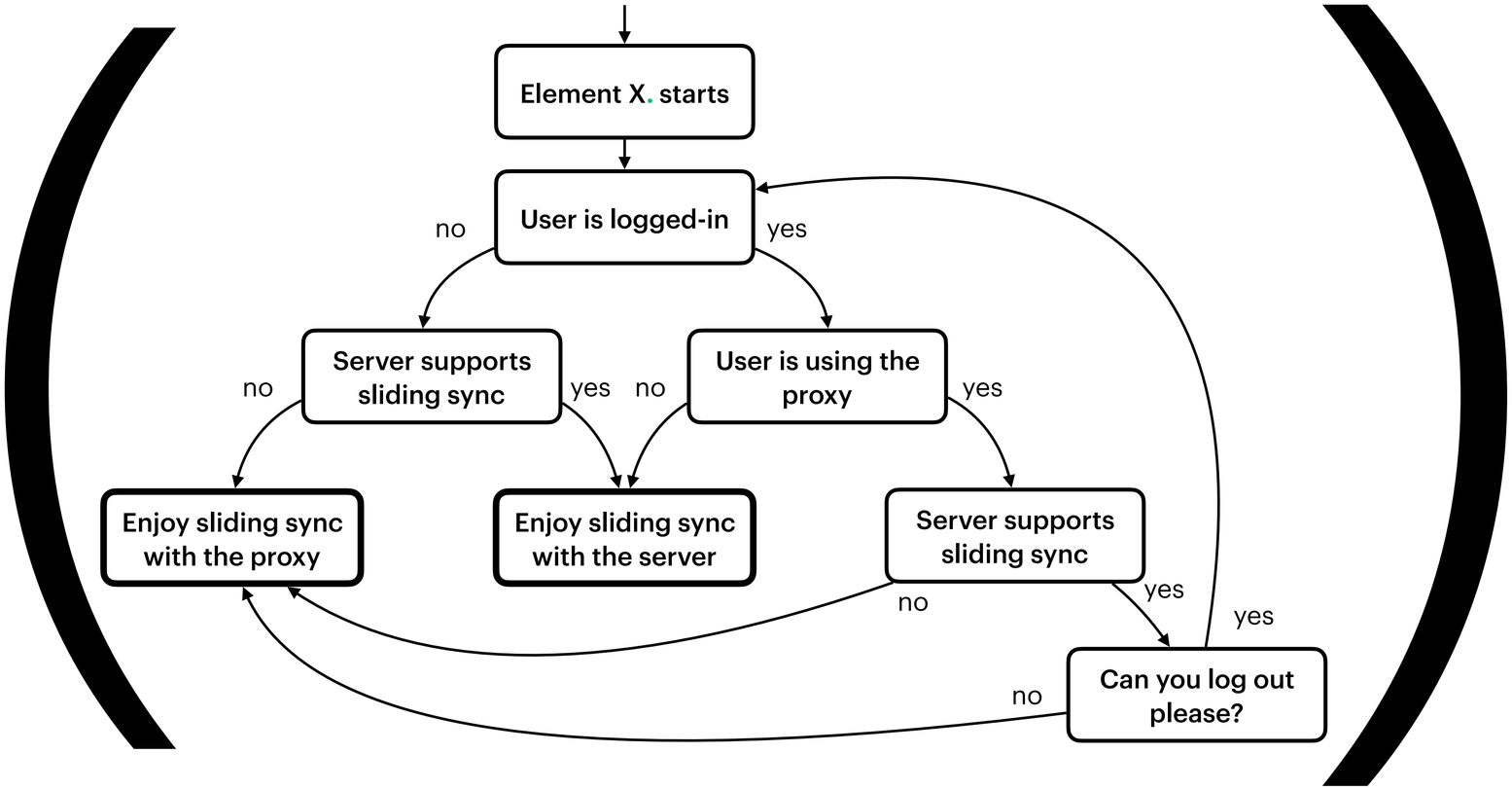
Enter MSC4186

Simplified sliding sync

- August 31st, 2024, first draft of MSC4186
- **`POST`** `/_matrix/client/unstable/org.matrix.simplified_msc3575/sync`
- **Simplified sliding sync is implemented inside Synapse!**
- The proxy doesn't and won't implement simplified sliding sync
- We are migrating from the proxy to Synapse
- We want to sunset the proxy as soon as possible
  - Farewell playmate
- The Matrix Rust SDK speaks both MSC3575 and MSC4186

Migration

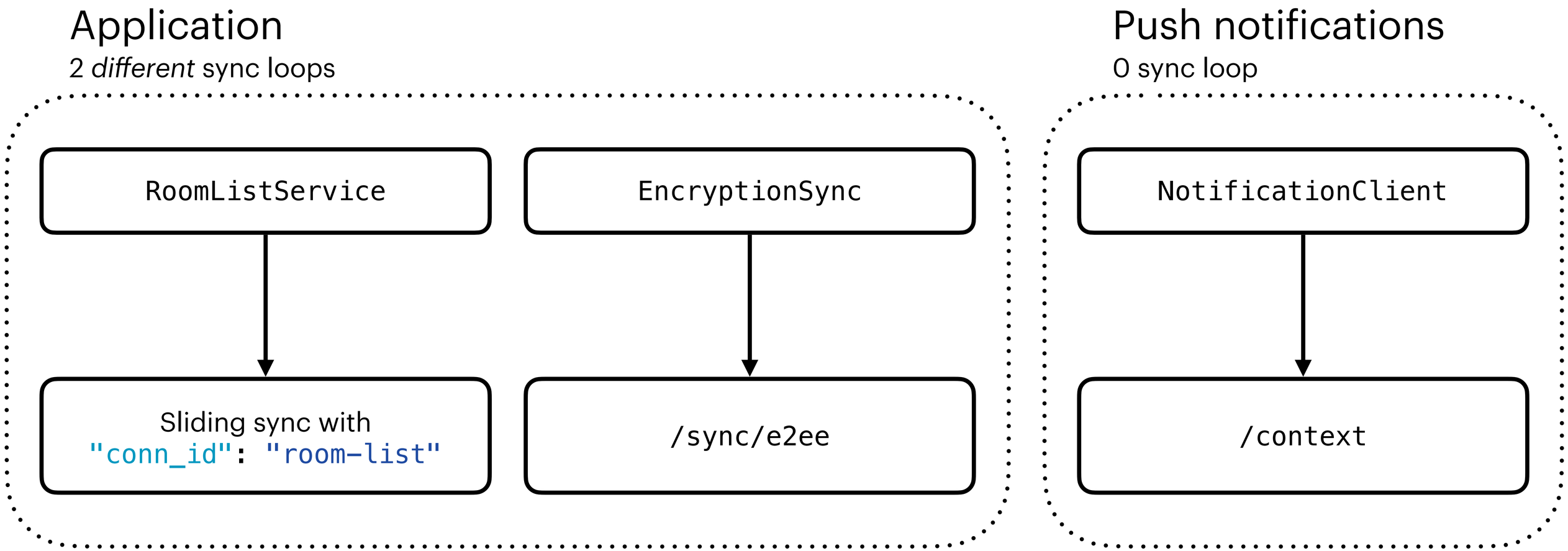
- Element X. automagically detects sliding sync on the homeserver and switches the user
- Homeserver maintainers have little to do
  1. Update the homeserver
  2. Wait for all users to migrate from the proxy
  3. Uninstall the proxy



Moooar simplifications

- Sliding sync has a non-negligible cost on the server-side
- Based on our experience, we see patterns that can be isolated
- Extract some usage of sliding sync into their own **endpoint**
  - New `/sync/e2ee`
  - Enhanced `/context`
  - Reduce loads and complexity for the server (may remove `conn_id`)

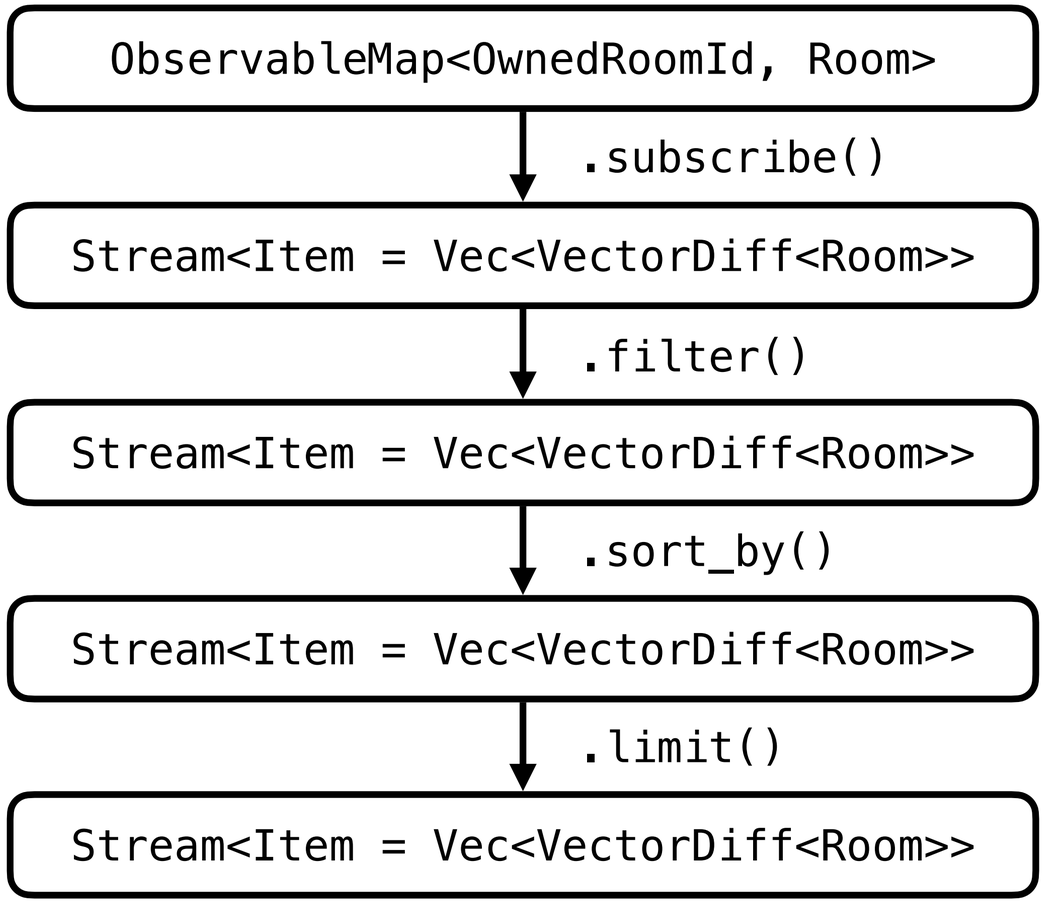
In a short future




Summit
(alt. 3180m)
Cabane du Trient, CH

Reactive programming

The beauty of higher-order `Stream`



- Single source of truth
- Ultra flexible, tiny memory footprint (`VectorDiff<Room>` is 72 bytes only), async, lazy, performant
- Traverse bindings to Swift and Kotlin easily and cheaply
- Learn more at: https://mnt.io/series/reactive-programming-in-rust/ ✨

Thanks!

- MSC3575 was an experimental proposal
  - Implemented inside a proxy
- **MSC4186** is the final proposal
  - Implemented inside Synapse
- Matrix Rust SDK supports both
- All applications based on the SDK support both:
  - Element X., Fractal…

Fun patches

- SDK:
  - (In `eyeball`) Implement the SortBy stream adapter (#43) +1,029 −23
  - Client-side sorting in RoomList (#3585) +1,516 −1,779
  - Remove RoomListEntry and ops (#3664) +167 −1,506
  - Migrate from Sliding Sync to Simplified Sliding Sync (#3676) +600 −695
  - Total +24,903 −15,262 over 525 pull requests
- Synapse:
  - Add Sliding Sync /sync endpoint (#17187) +2,302 −15