matrix

RTC

Introducing a new concept to the Matrix-specification

timok@element.io
@toger5:matrix.org

Why are we here today?

matrix

We do not want a calling solution limited for 1:1 calls.

We want more than VoIP.
(Historically: ThirdRoom, Nowadays: shared documents (neoboard))

A flexible real time setup is extremely useful:
We want to get it right

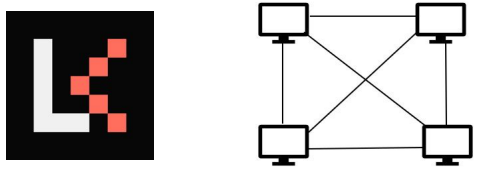
Matrix gets put to the test if it provides the required primitives to power **MatrixRTC**

We want to get the proposal trough spec review → The proposal needs to be really good.

Lets see what we improved!

The components of MatrixRTC

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

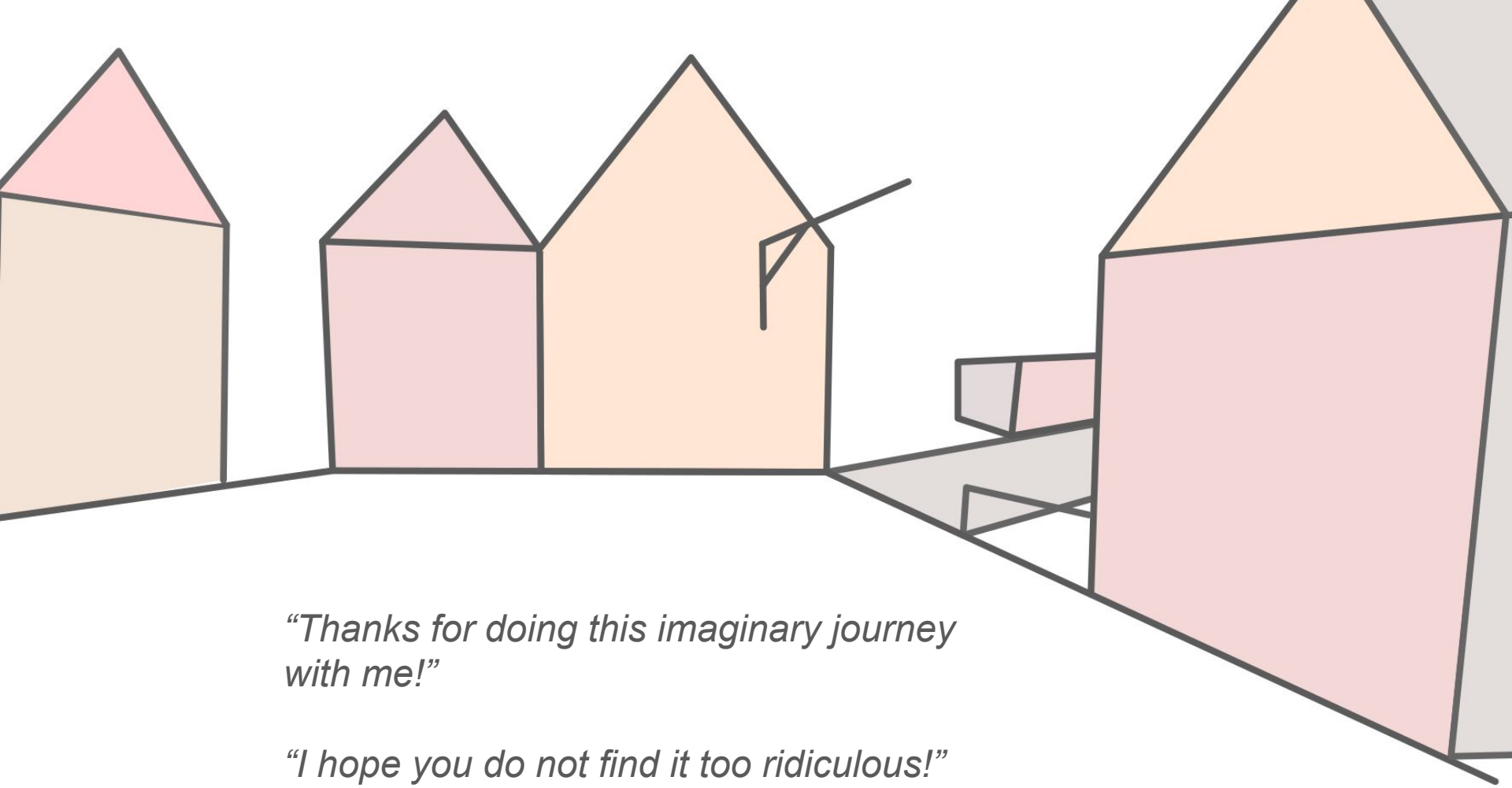


What is great about the RTC proposal

matrix

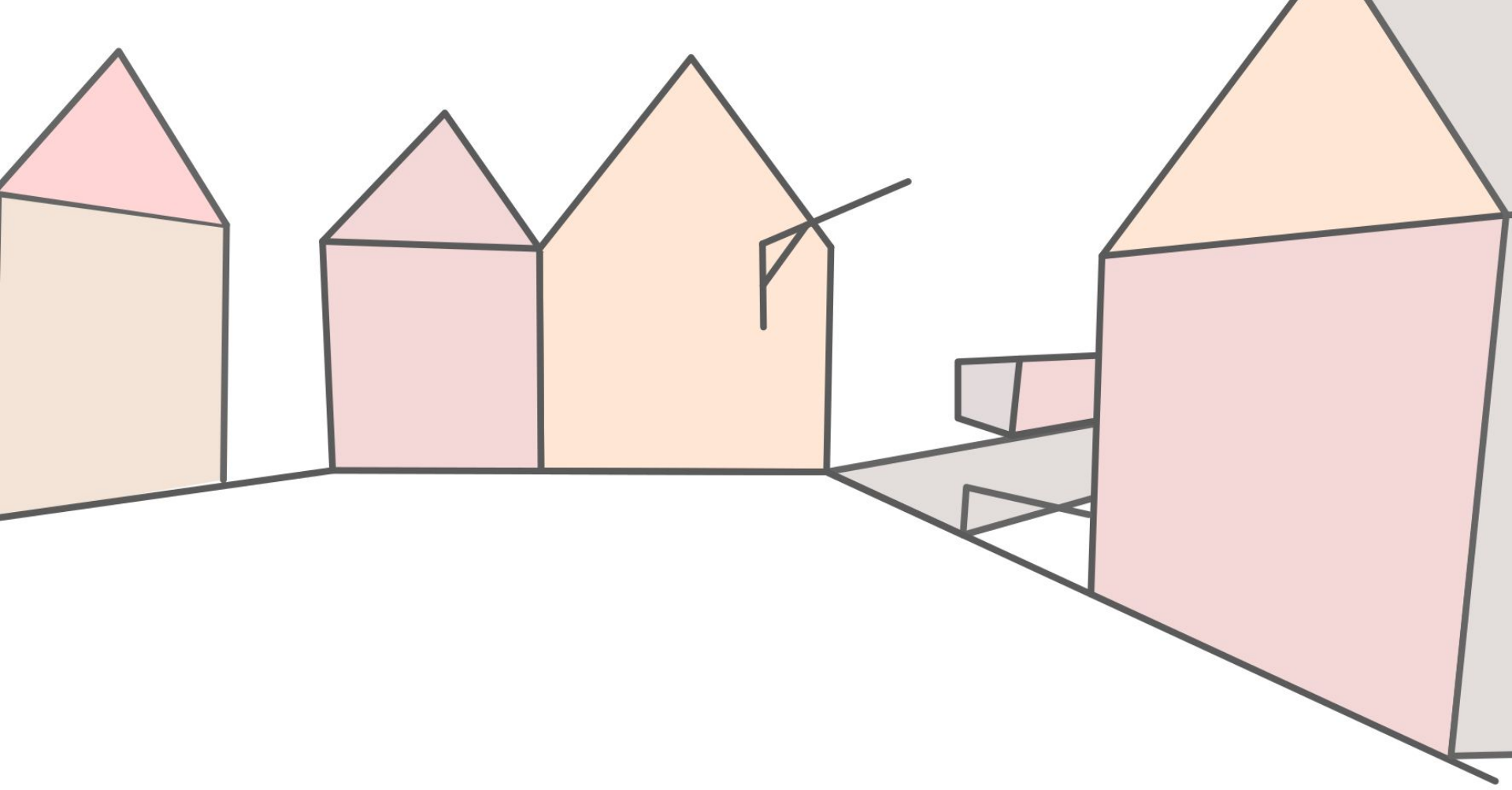
The **MatrixRTC** approach checks more boxes for VoIP than ever before

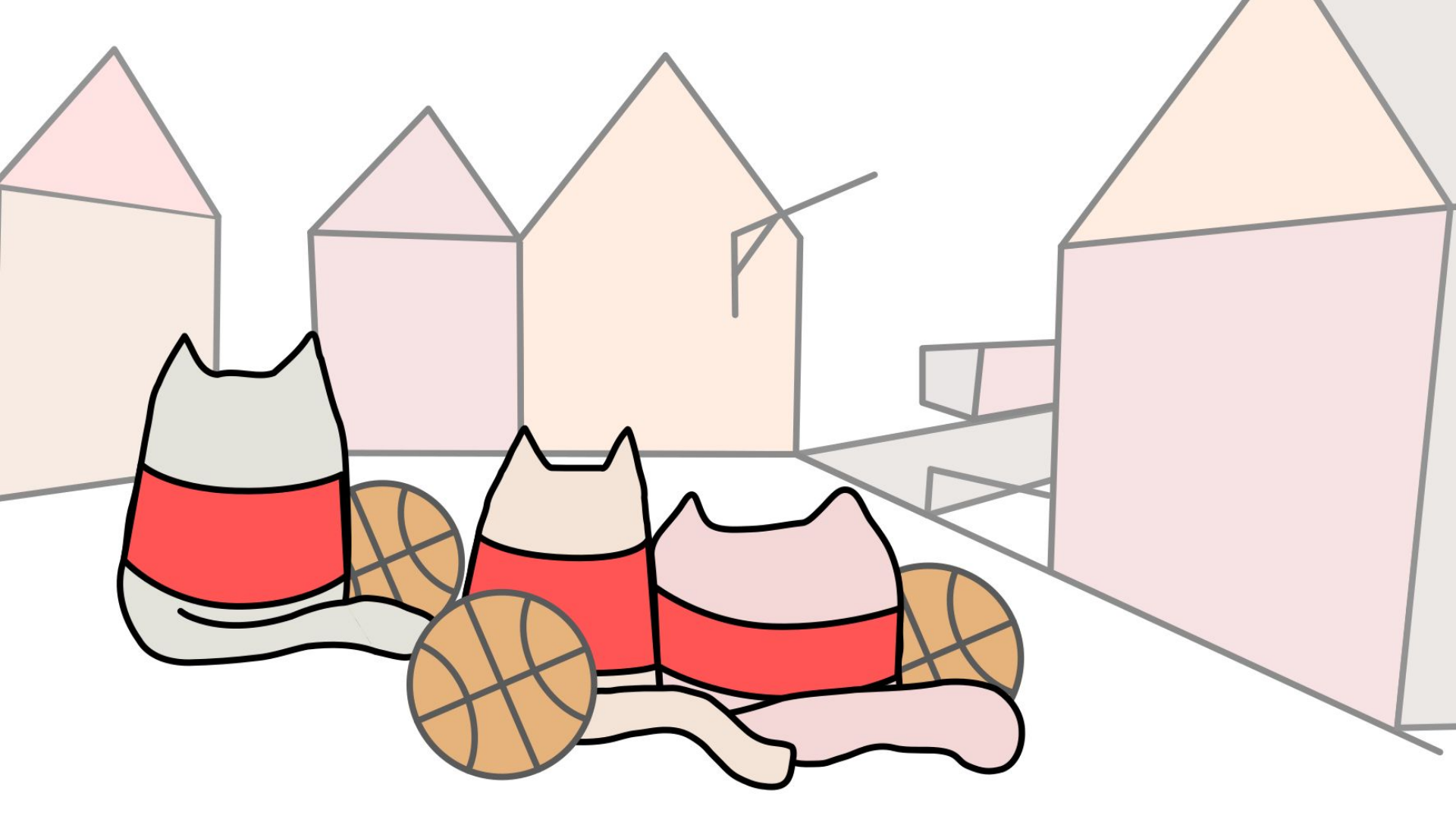
- ✅ Scalable
- ✅ Interchangeable components
- ✅ Very flexible! (can be used for much more than VoIP)
- ✅ Secure, Federated (even more now), verified identities ...
- ✅ Moderatable by design (new)

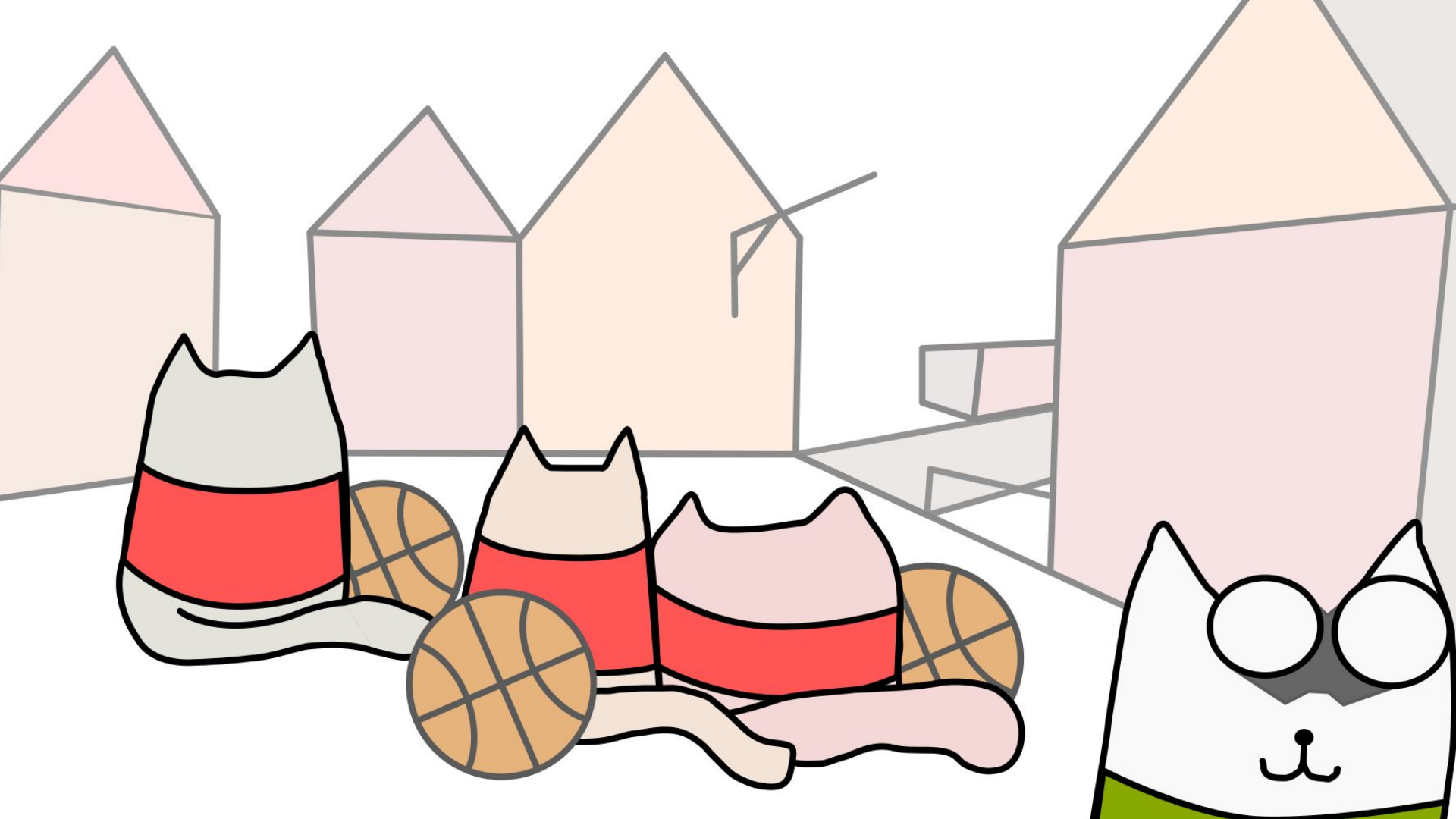


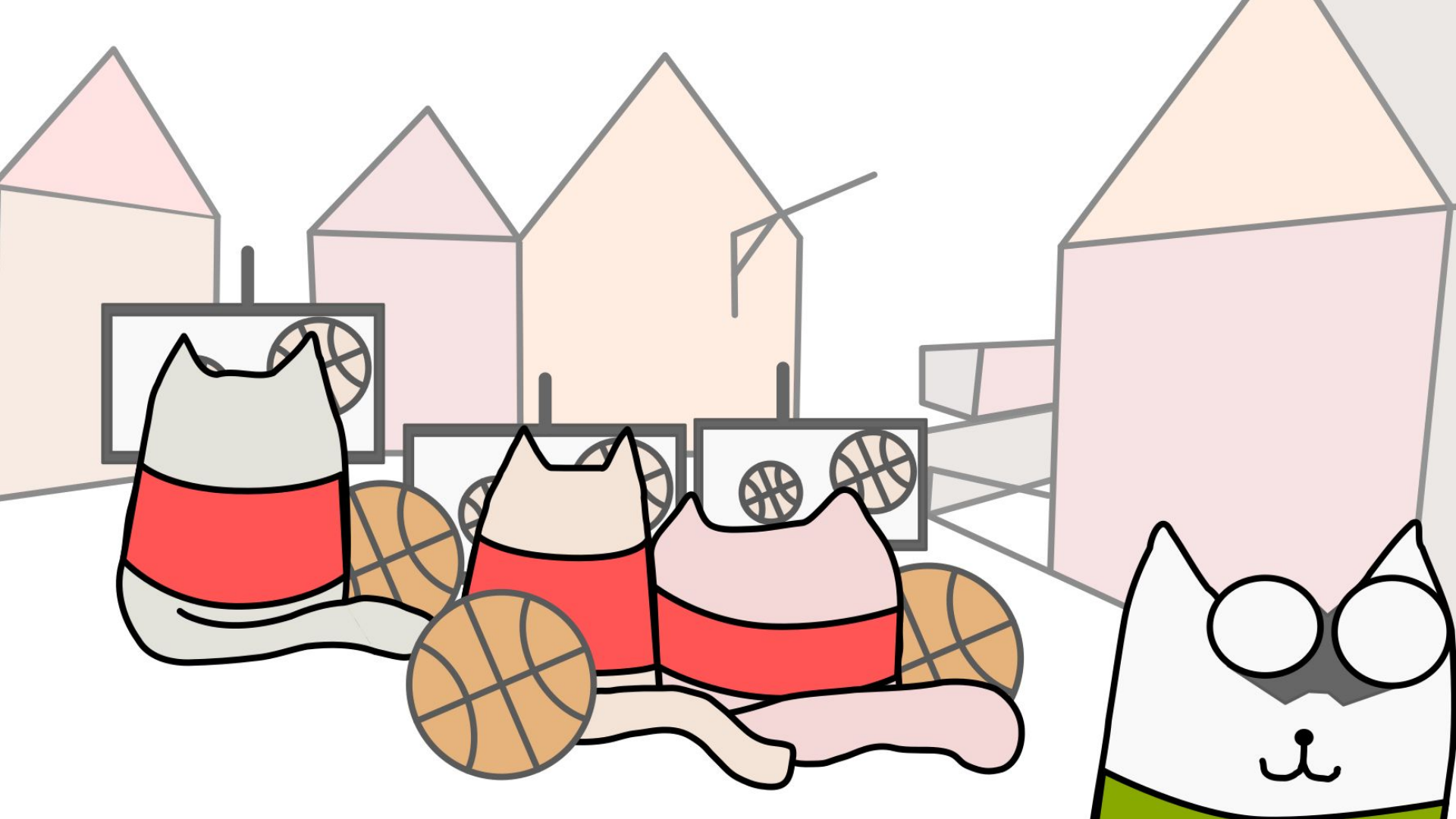
“Thanks for doing this imaginary journey with me!”

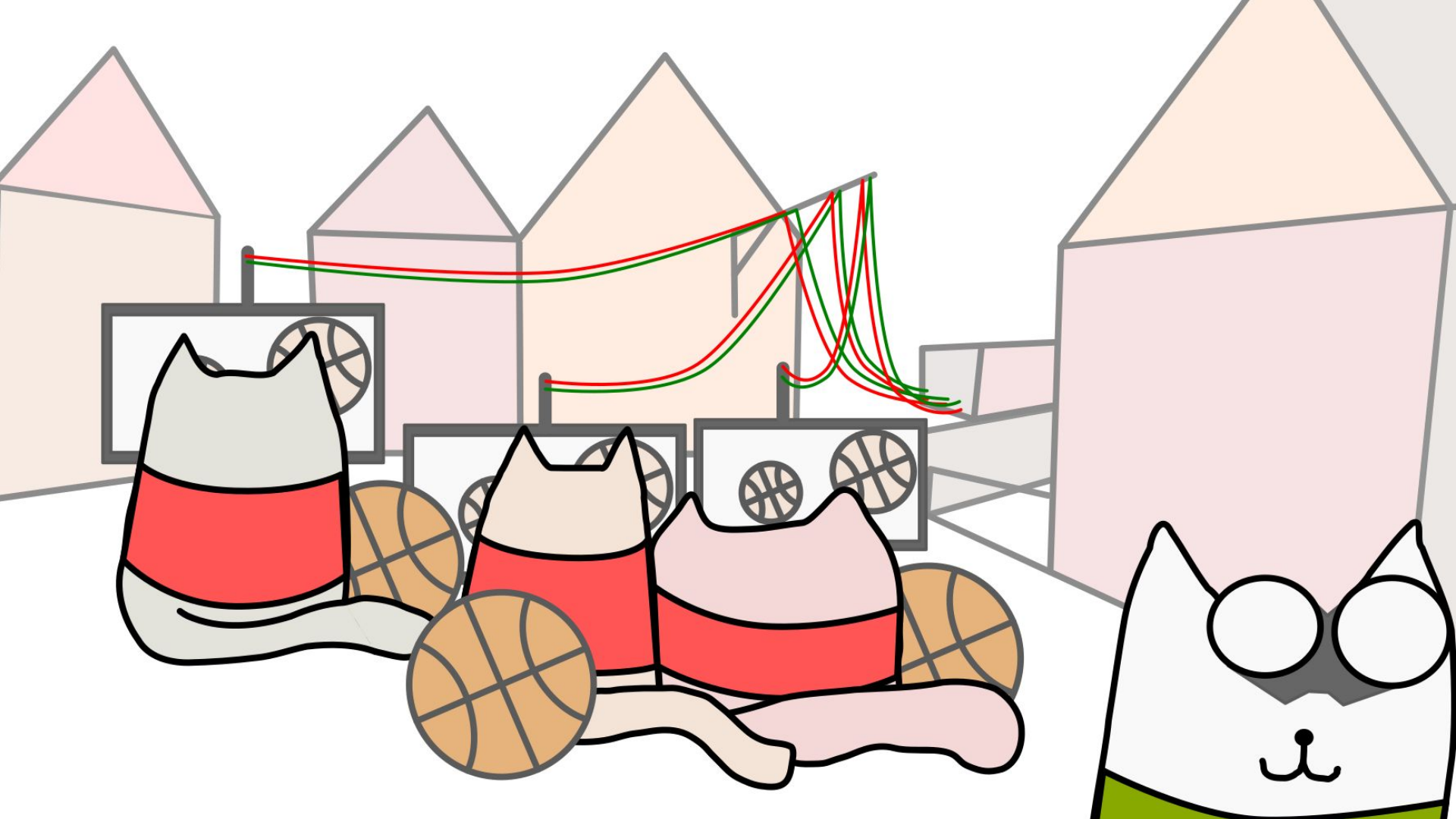
“I hope you do not find it too ridiculous!”

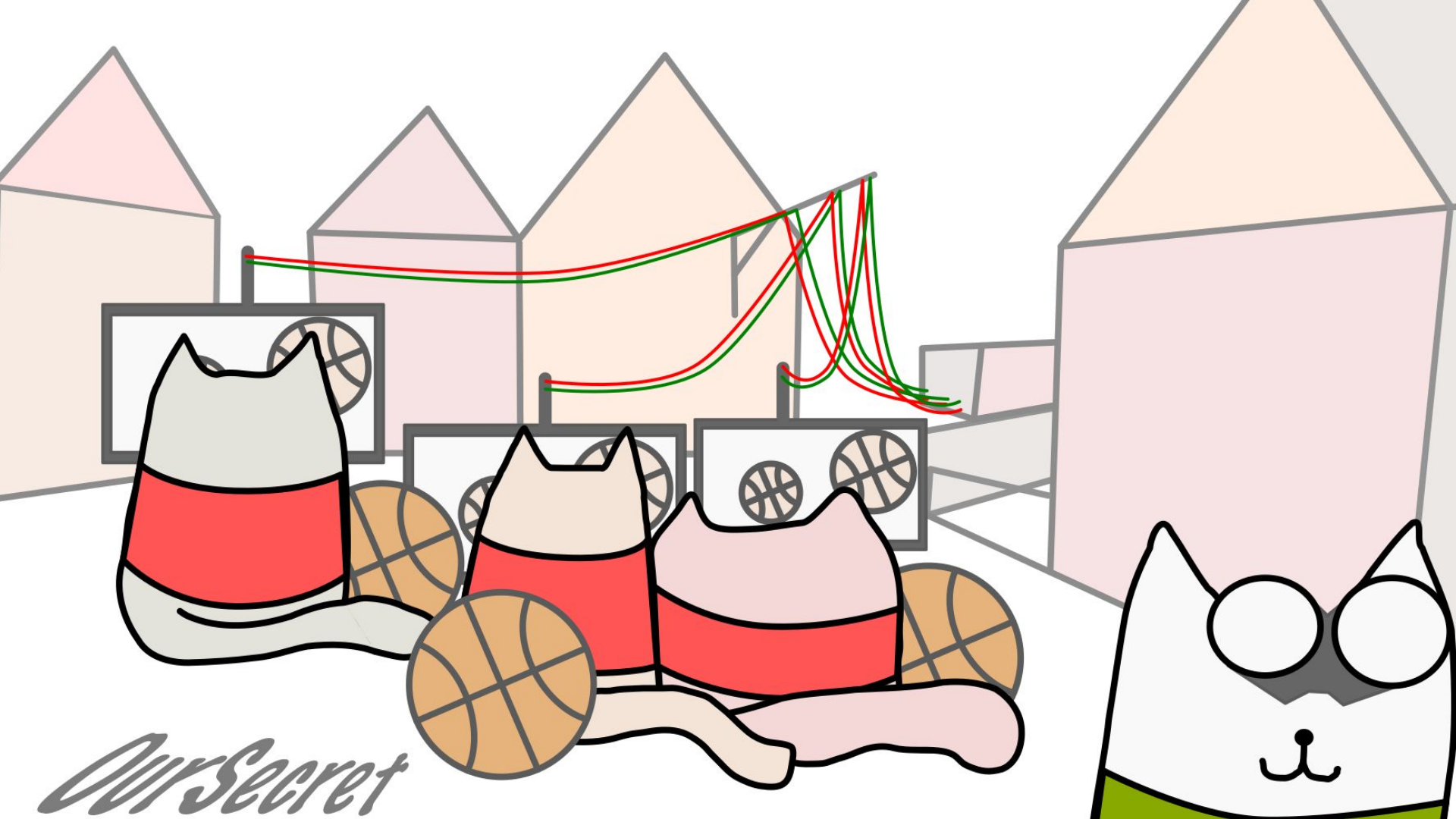











OurSecret

Improvements

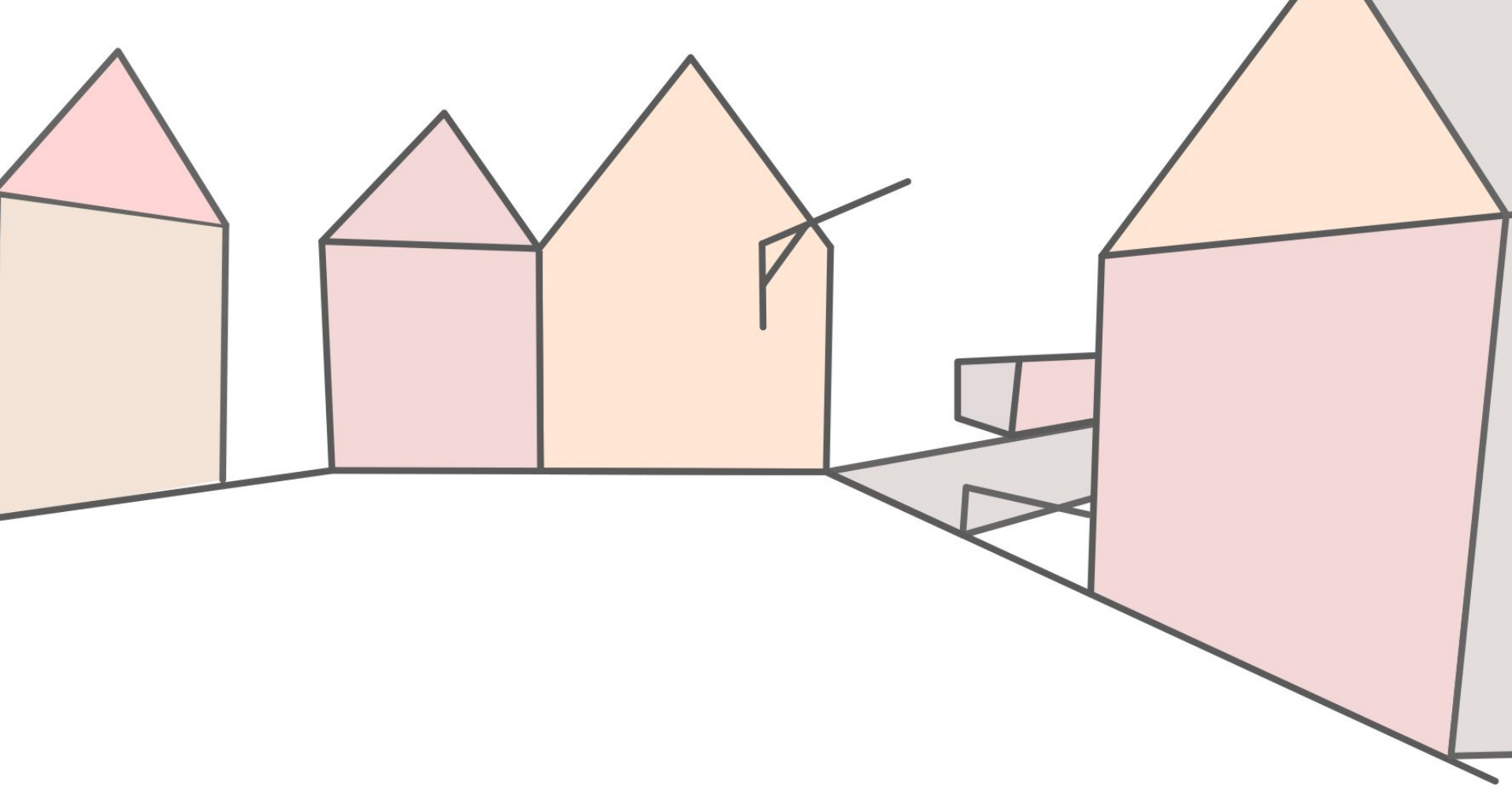
matrix

What they did not like:

- All other could see encryption password
- Connection procedure
- Public server metadata

What the cats parents did not like

- That they played in the middle of the night…



Improvements for the situation

matrix

Toms parents problem: **That the cats played in the middle of the night…**

Solution:

- remove the batteries
- add playstation docking station **slots** on the yard
- restrict slots based on t-shirt color and game

“Those docking station slots quickly become the places where the kids meet.”

Improvements for the situation

matrix

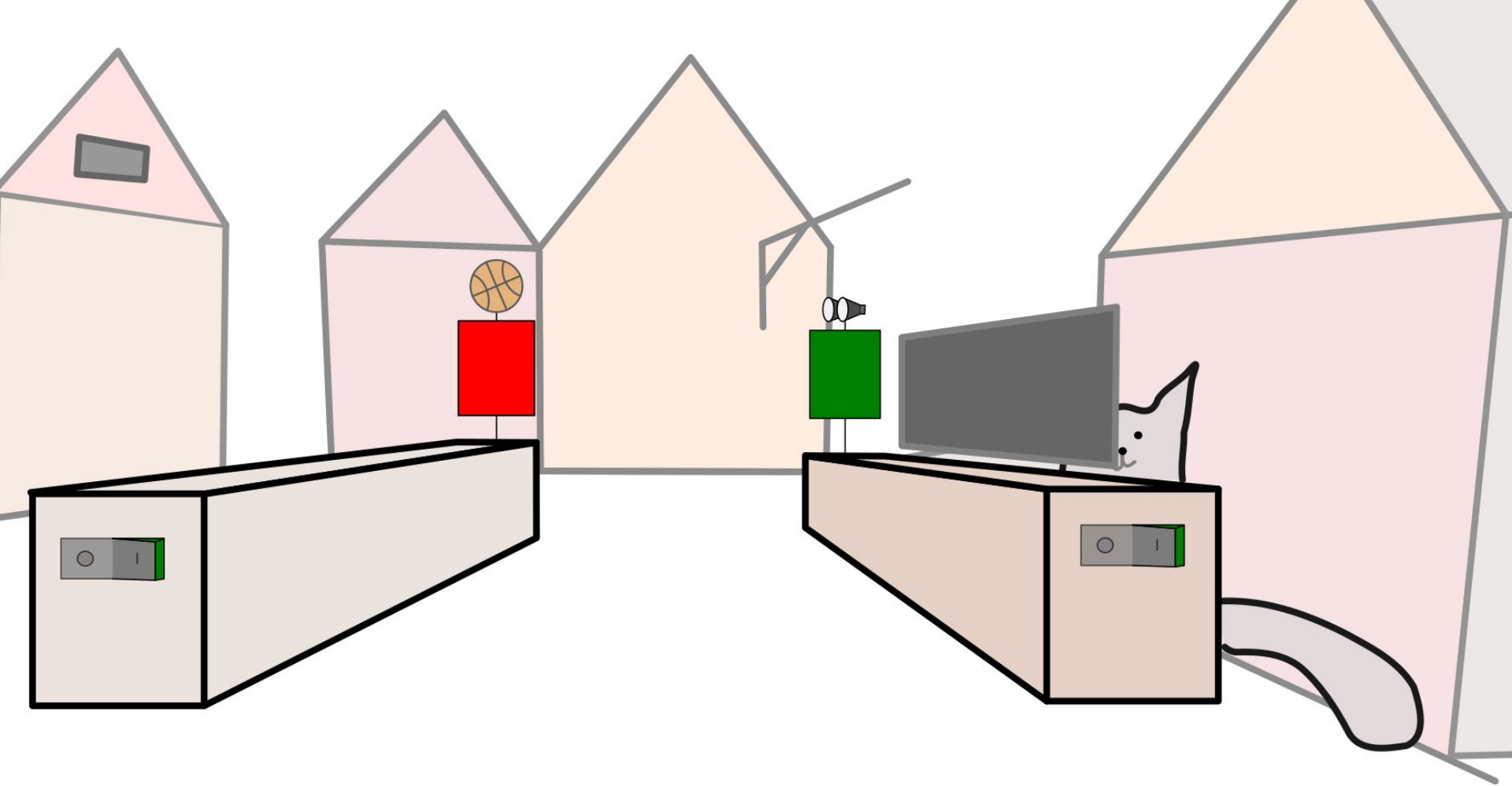
Toms parents problem: **That they played in the middle of the night…**

Solution:

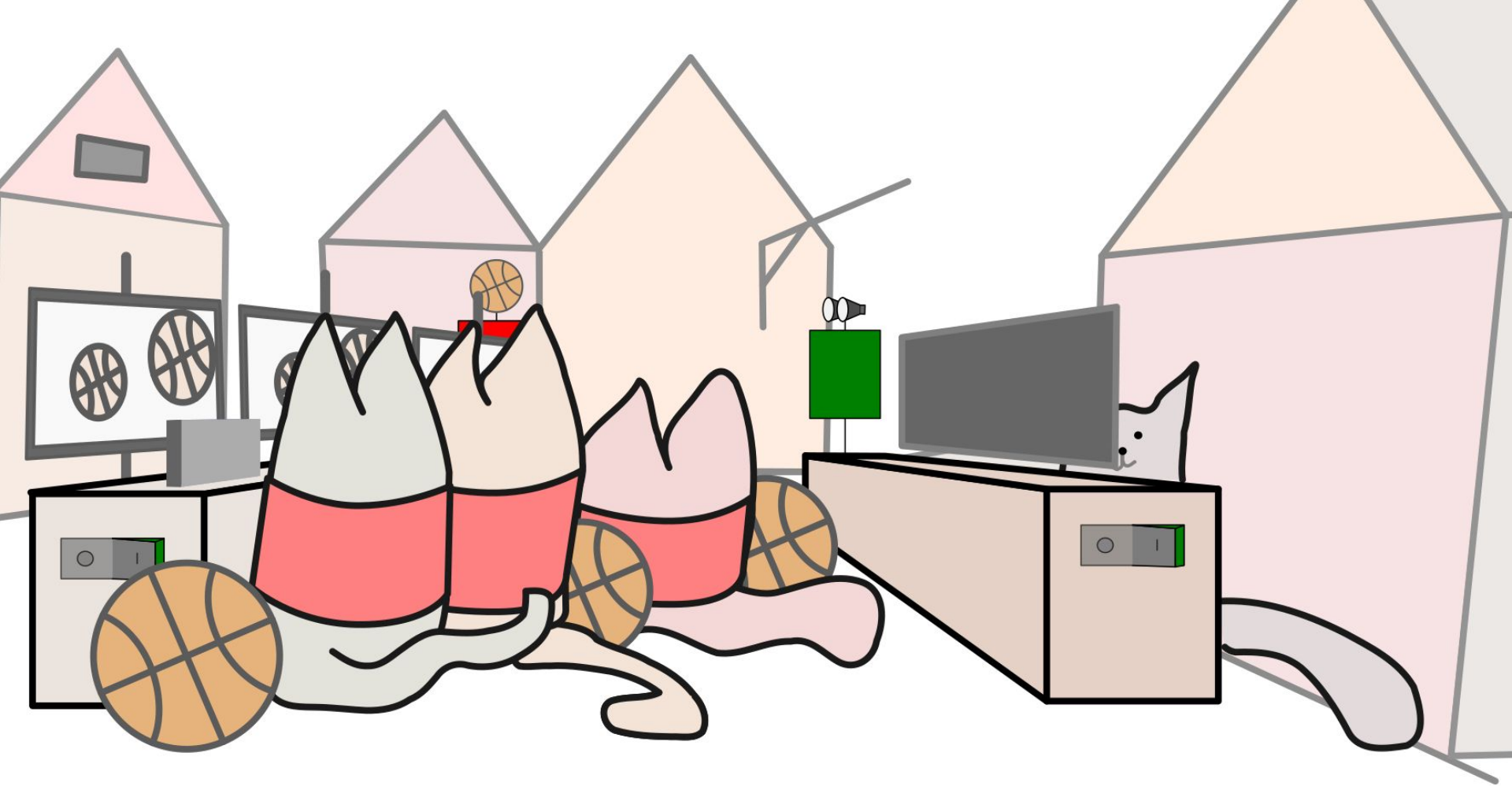
- remove the batteries
- add playstation docking station **slots** on the yard
- restrict slots based on t-shirt color and game

“Those docking station slots quickly become the places where the kids meet.”

Slot Moderation State Event



Slot Moderation State Event



Slot Moderation State Event

Improvements for the situation

Problem: **All other kids in the yard could also see their encryption password**

Solution:

- Small paper notes
- Update key whenever someone leave or joins

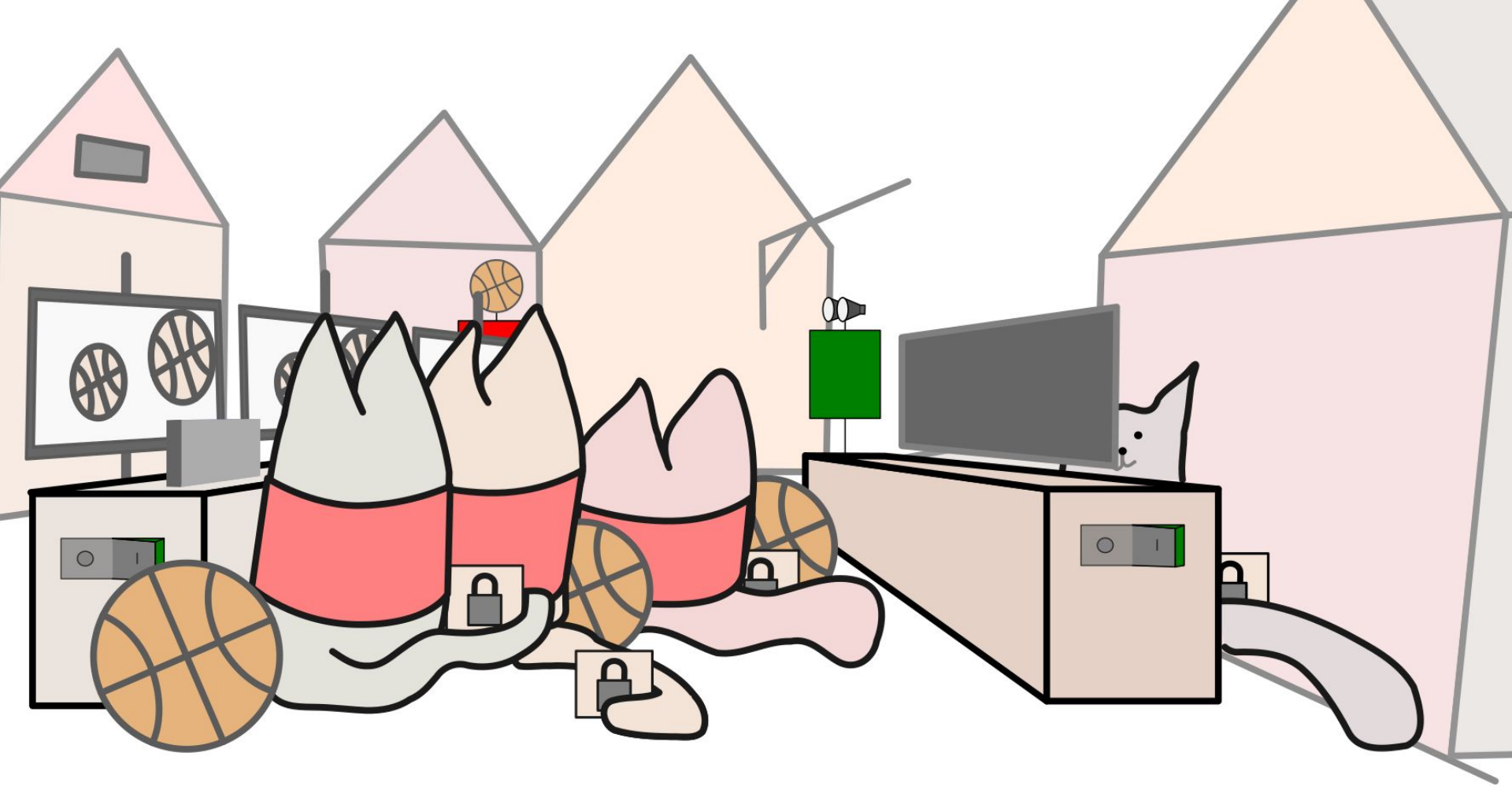
Improvements for the situation

Problem: **All other kids in the yard could also see their encryption password**

Solution:

- Small paper notes
- Update key whenever someone leave or joins

To-Device key sharing



Slot Moderation State Event

To-Device key sharing

Improvements for the situation

Problem: **Connection procedure: Discussing where to connect (when Tom eventually leaves → reconnect)**

Solution:

- “Publish where you want”
- Same amount of upload connection
- Subscribing to each publication
- No additional data consumption
- Can be the same outcome as consensus (all from the same HS)

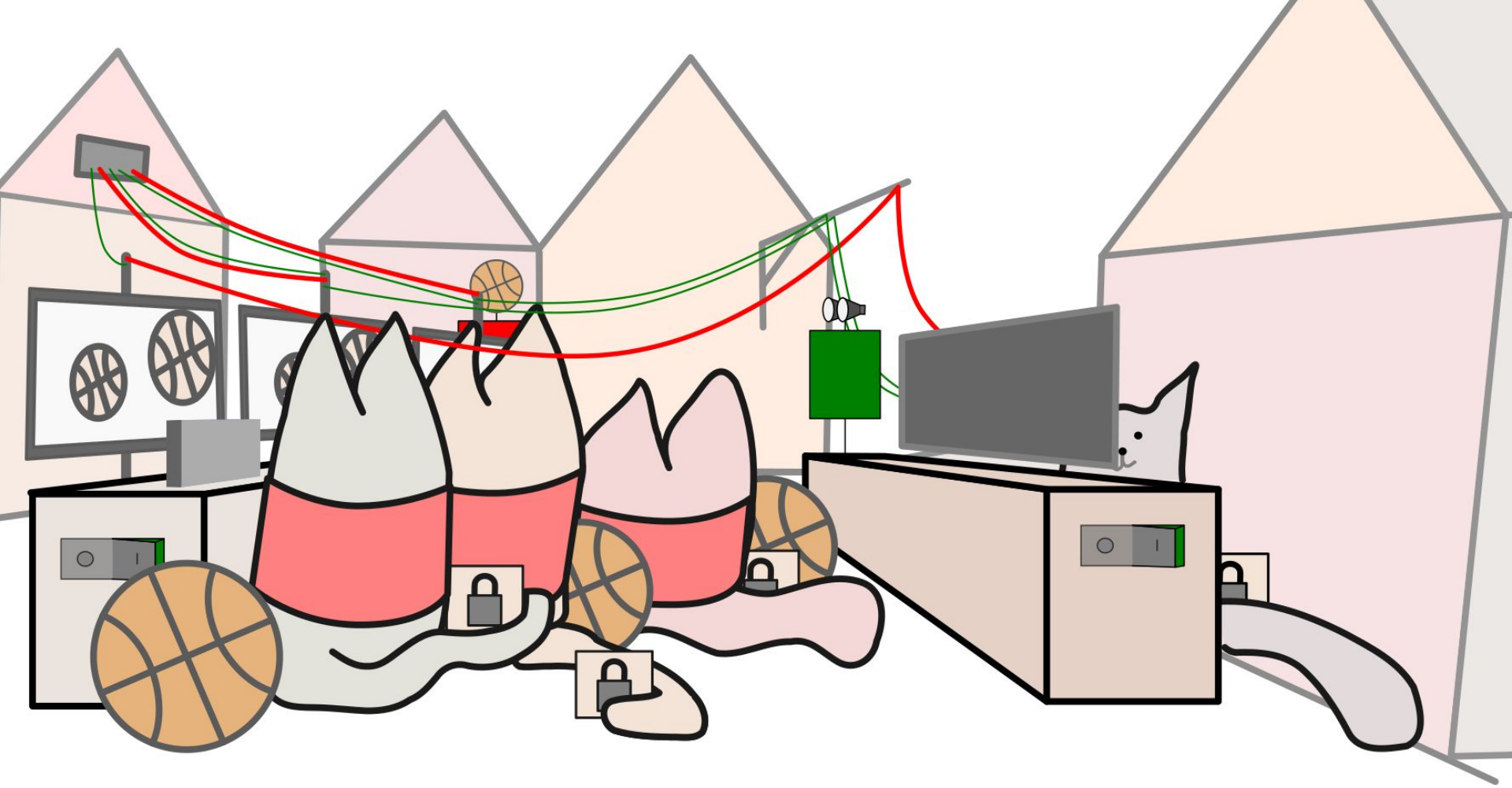
Improvements for the situation

Problem: **Connection procedure: Discussing where to connect (when Tom eventually leaves → reconnect)**

Solution:

- “Publish where you want”
- Same amount of upload connection
- Subscribing to each publication
- No additional data consumption
- Can be the same outcome as consensus (all from the same HS)

Multi-SFU



Slot Moderation State Event

To-Device key sharing

Multi-SFU

Improvements for the situation

matrix

Problem: **Public server metadata** (media server knows identity)

Solution:

- “don’t write your real name on the ethernet wire!”
- Label ethernet wires with pseudonymous identities

Improvements for the situation

matrix

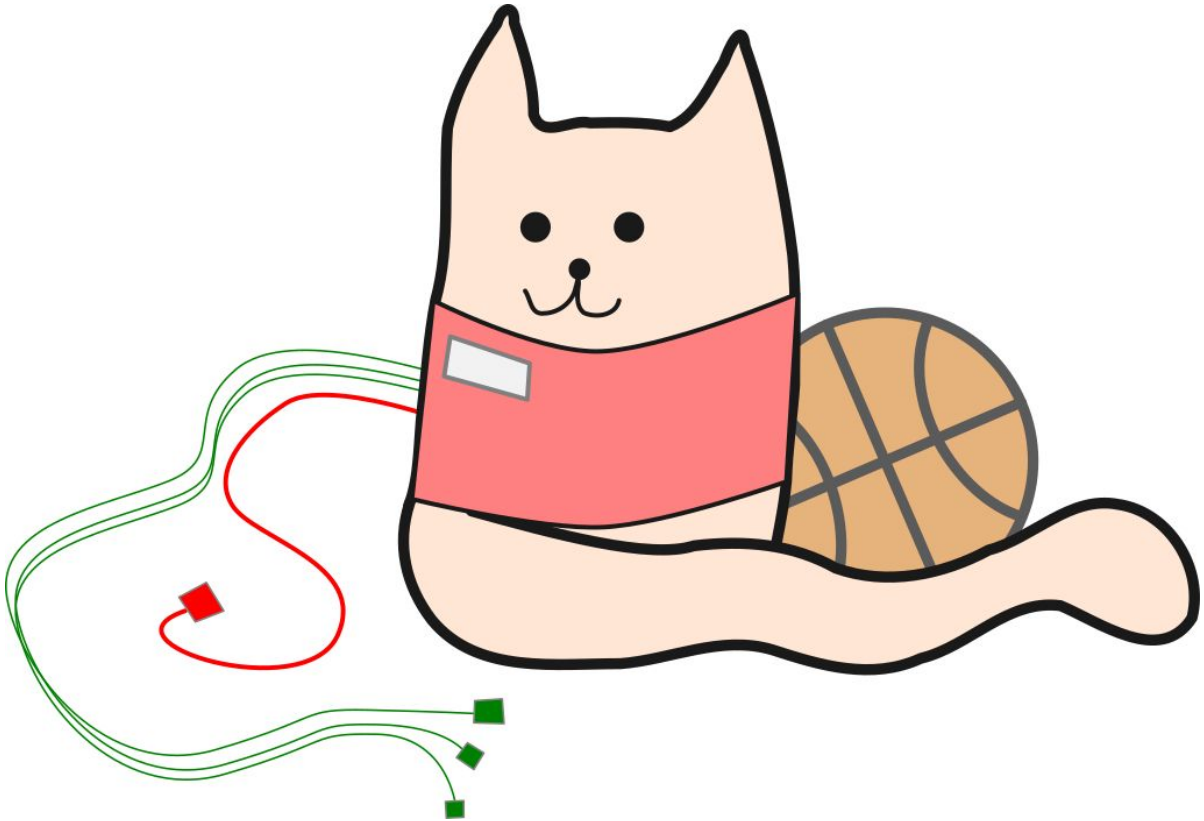
Problem: **Public server metadata** (media server knows identity)

Solution:

- “don’t write your real name on the ethernet wire!”
- Label ethernet wires with pseudonymous identities

Pseudonymous Member.Id

matrix



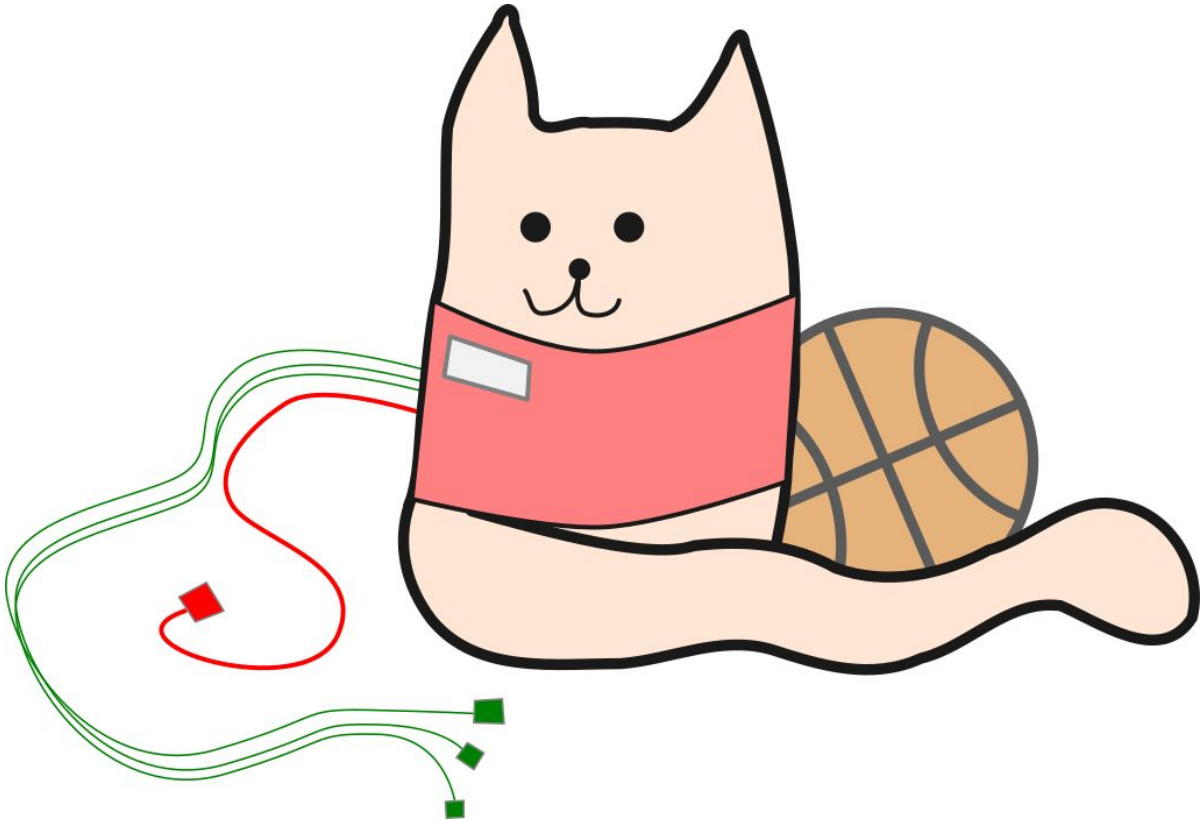
Slot Moderation State Event

To-Device key sharing

Multi-SFU

Pseudonymous Member.Id

matrix



Those are the new changes we will look into today!

Slot Moderation State Event **To-Device key sharing** **Multi-SFU** **Pseudonymous Member.Id**

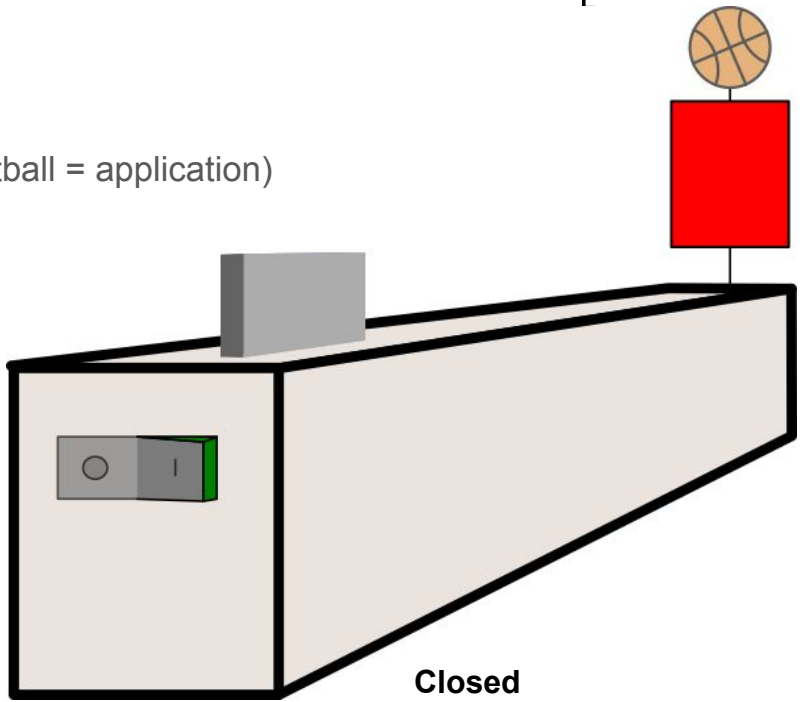
Agenda

- Slot Moderation State Event
- Member event - Limit Metadata
  - Pseudonymous member id
  - Is there something better than state?
  - Sticky events
- Key sharing
- Multi-SFU

Slot Moderation Event

Moderate RTC activity:

- State key: defines activity (t-shirt color = id, basketball = application)
- Only moderators can alter
- Shared state: (video vs voice call …)
- `{}` = closed, contains `application` = open



Open

```
// event type: "m.rtc.member"
{
  "application": {
    "type": "play.basketball", // Basketball
    // further fields for the application (optional)
    “play.basketball.match_duration”: 100
    “play.basketball.id”:”friday_night_match_1”
  },
},
// state_key: "play.basketball#RED", // Group identity: Item (Basketball) + T-shirt color
```

Closed

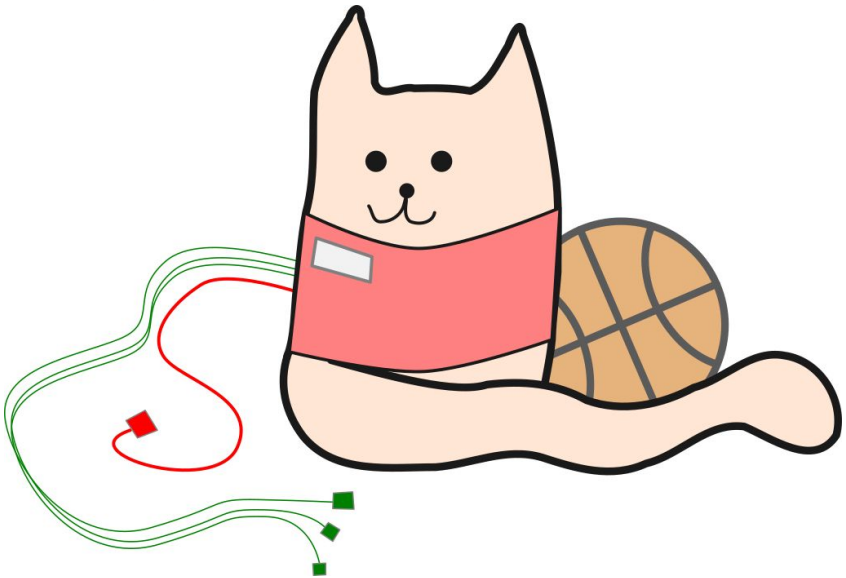
```
// event type: "m.rtc.member"
{},
// state_key: "play.basketball#RED"
```

m.rtc.member

Instead of going out in the yard we share our current **state** with the room.

Connect

```
// event type: "m.rtc.member"
{
  "slot_id": "play.basketball#RED", // Group identity: Item (Basketball) + T-shirt color
  "application": {
    "type": "play.basketball", // Basketball
    // further fields for the application (optional)
    "play.basketball.player_role": “center"
  },
  "member": {
    "id": "xyzABCDEF0123" // Name Tag (Pseudonymous)
    "claimed_device_id": "DEVICEID"
    "claimed_user_id": "@user:matrix.domain"
  },
  "rtc_transports": [
    {...TRANSPORT_1}, // Ethernet wire (where do I published)
  ],
  "versions": [
    "v0",
    "example.mscXXXX.asymmetric_encryption"
  ],
}
```



m.rtc.member

Instead of going out in the yard we share our current **state** with the room.

Update

```
// event type: "m.rtc.member"
{
  "slot_id": "play.basketball#RED",
  "application": {
    "type": "play.basketball",
  },
  "member": {
    "id": "xyzABCDEF0123"
    "claimed_device_id": "DEVICEID"
    "claimed_user_id": "@user:matrix.domain"
  },
  "rtc_transports": [],
  "m.relates_to":{
    first event.
      rel_type: "m.reference",
      event_id: "$join_event_id"
  },
  "versions": [
    "v0",
    "example.mscXXXX.asymmetric_encryption"
  ],
}
```

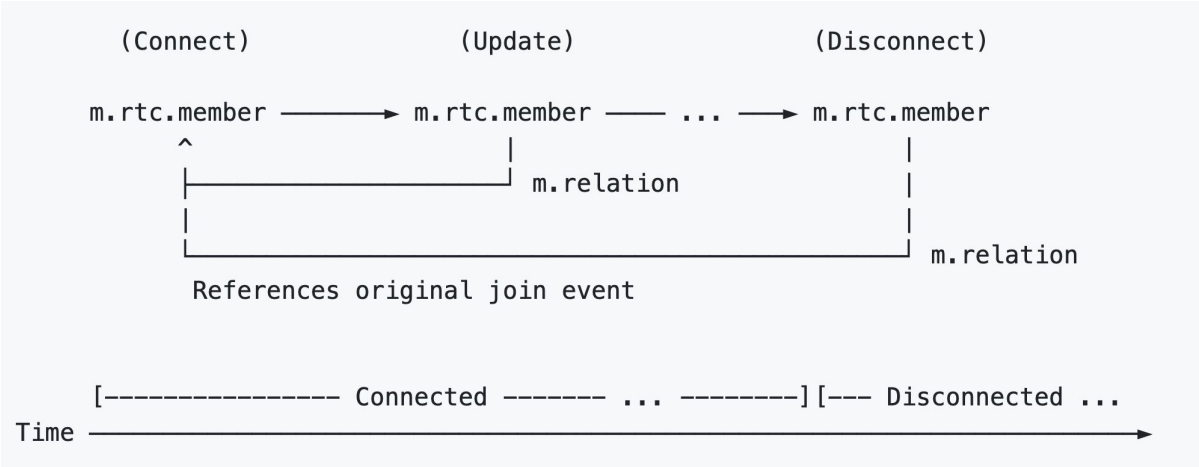
Disconnect

```
// event type: "m.rtc.member"
{
  "slot_id": "play.basketball#RED",
  "m.relates_to":{
    first event.
      rel_type: "m.reference",
      event_id: "$join_event_id"
  },
 "disconnect_reason": {
    "class": "server_error",
    "reason": "ice_failed",
    "description": "Failed to …",
  }
}
```

m.rtc.member

Instead of going out in the yard we share our current **state** with the room.

Send disconnect event ahead of time as a **delayed event** (Leave guarantee)



`m.rtc.member` Event

*Instead of going out in the yard we share our current **state** with the room.*

Requirements:

- Delivery guarantee
- Minimal metadata leakage (E2EE)
- Leave update on client disconnect (delayed event)

Previously state events!

- Share data with a room without back pagination
- Each m.rtc.member event with different state key
- Cancellable Delayed events for reliable membership lifecycle

Issues:

- State needs to be keyed by user + device + application
  - Not user protected (msc owned user state)
  - Bloating state over time
  - No need for state res
- No encryption → Metadata Leakage

`m.rtc.member` Event

*Instead of going out in the yard we share our current **state** with the room.*

We want to solve metadata leakage:

- Just use E2EE room events?
- Could work: a keyed map can also be computed locally
- **No delivery guarantee**

`m.rtc.member` Event

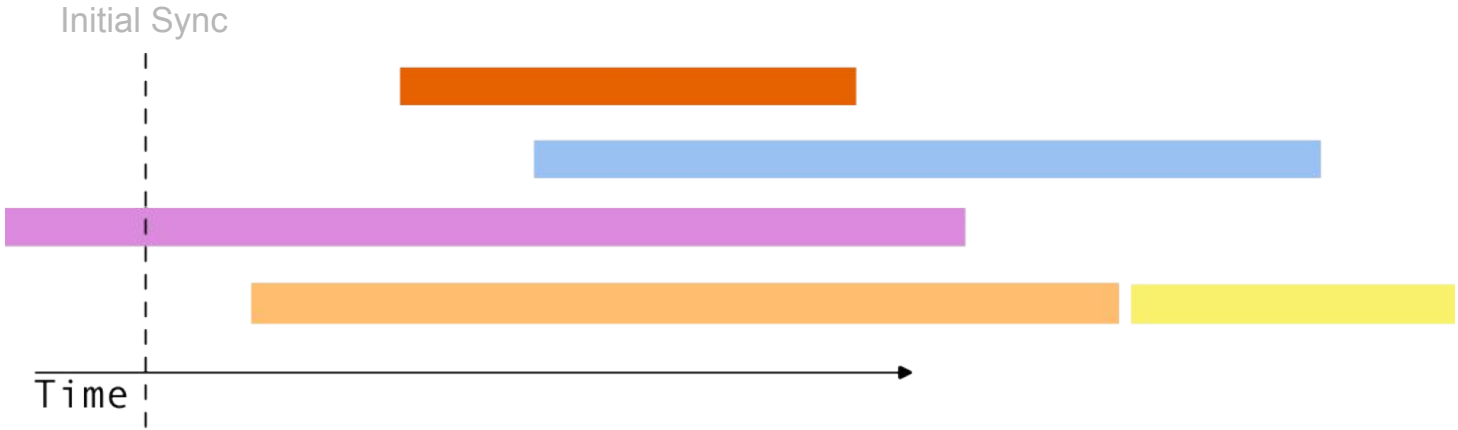
*Instead of going out in the yard we share our current **state** with the room.*

We want to solve metadata leakage:

- Just use E2EE room events?
- Could work: a keyed map can also be computed locally
- **No delivery guarantee**

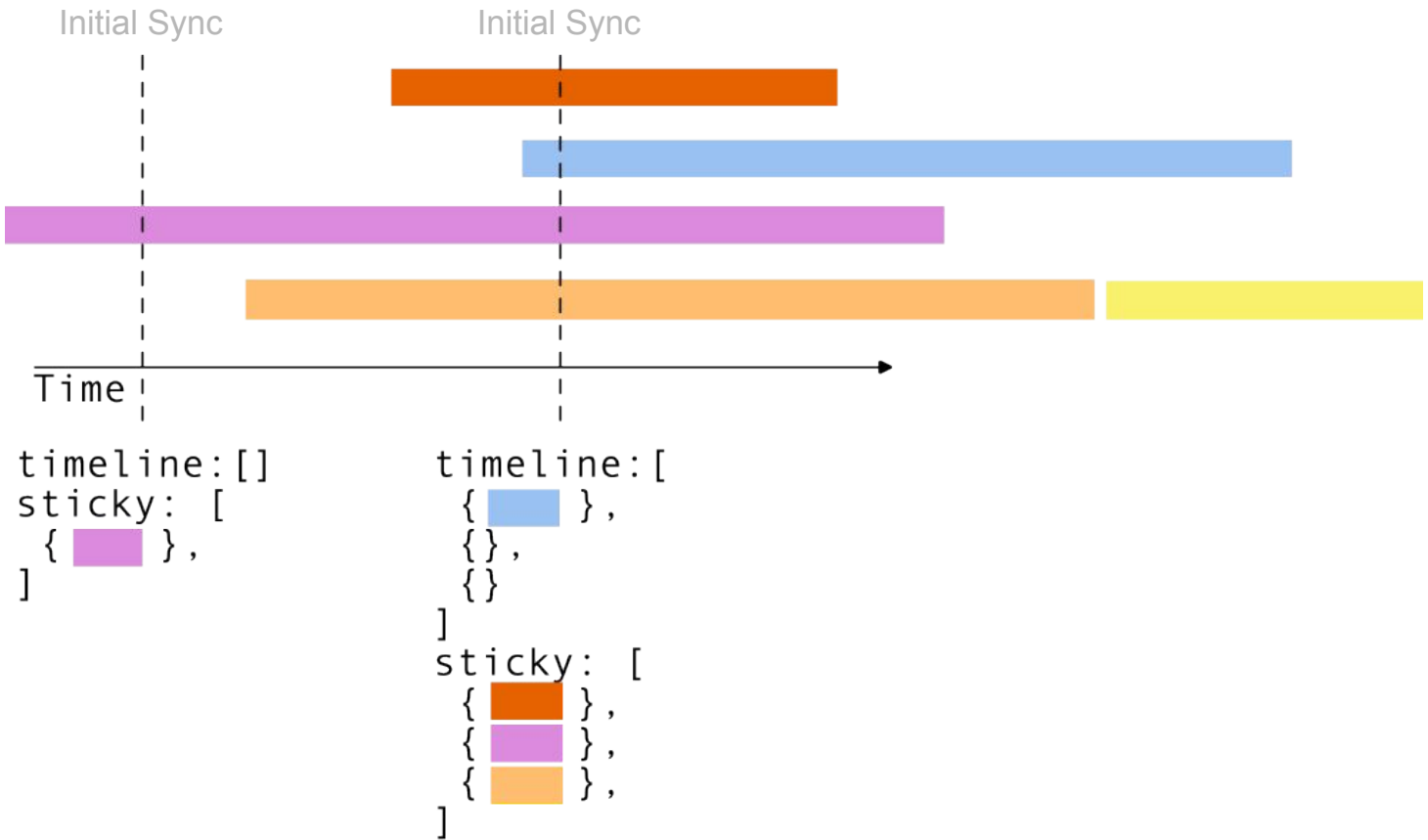
Sticky events MSC4354: timeline events with (temp) delivery guarantee

Sticky Events

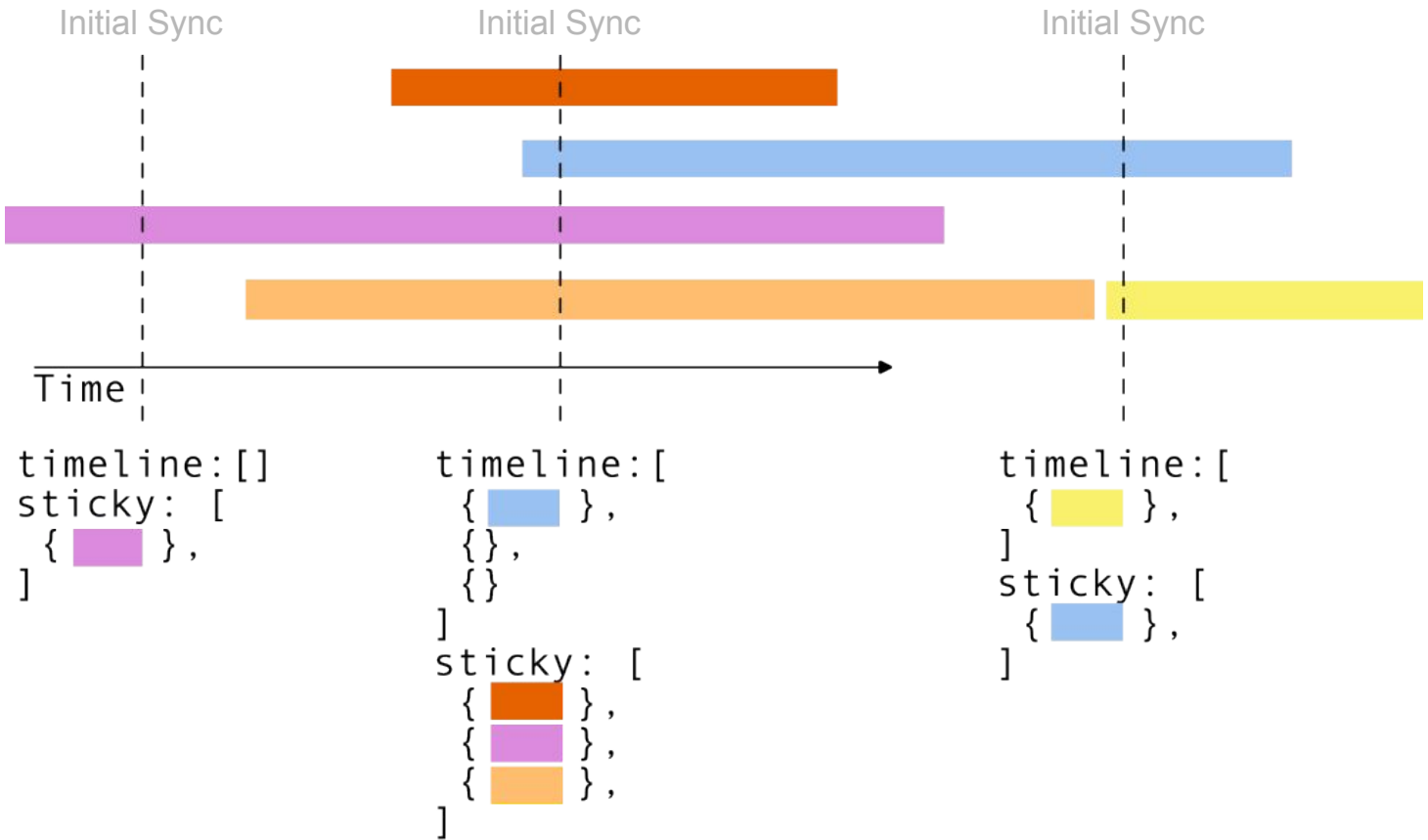



```
timeline:[]
sticky: [
 { },
]
```

Sticky Events



Sticky Events

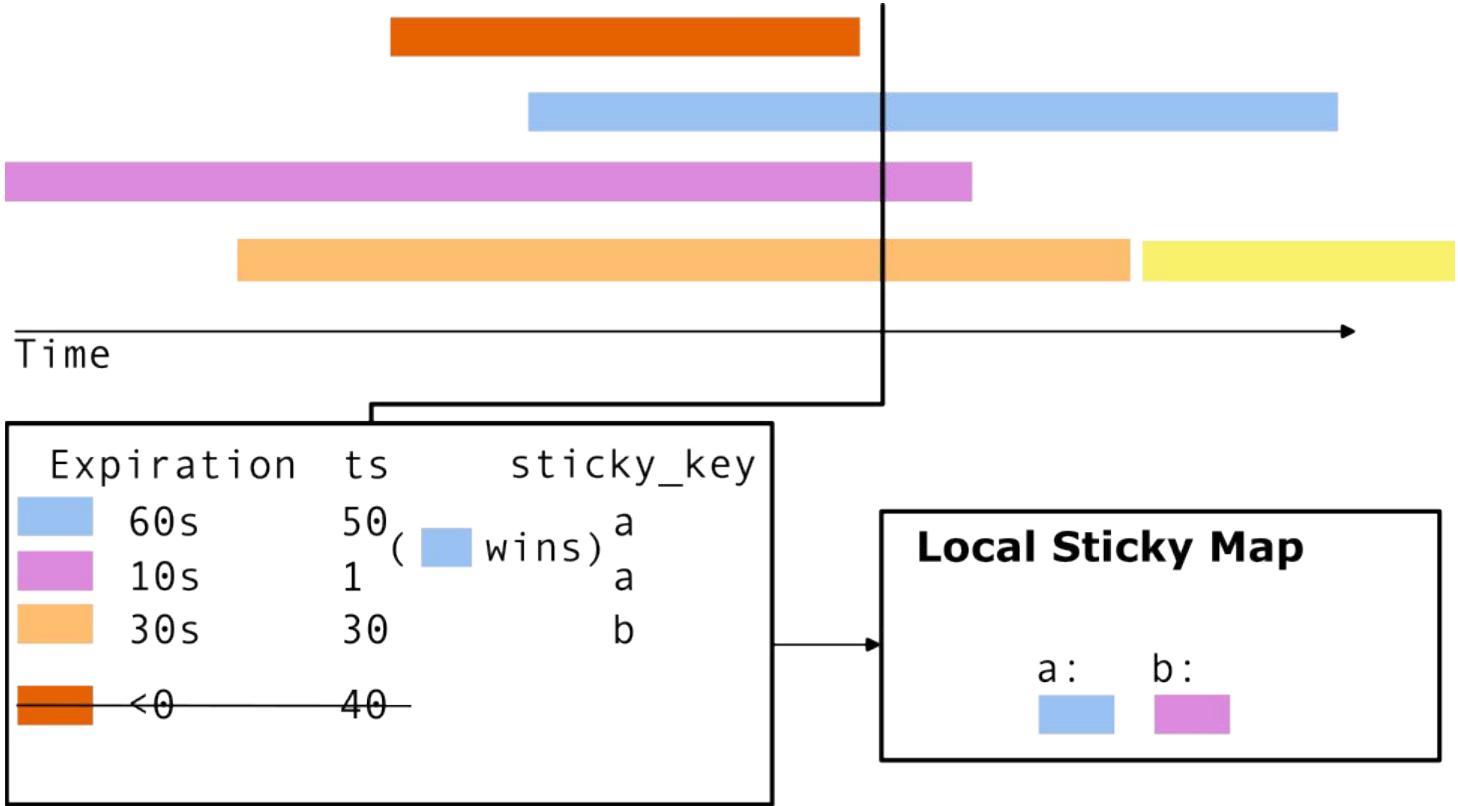


Encrypted Application state

- E2EE Local Map
- 4-uple: **room_id**, **event_type**, **user_id**, **sticky_key**
- Delivery guarantee like room state (while being sticky)
- Sticky lifetime ensures auto cleaning (No sync bloat)
- **MatrixRTC** is first adopter of user owned encrypted state
- **Cancellable delayed events** are compatible

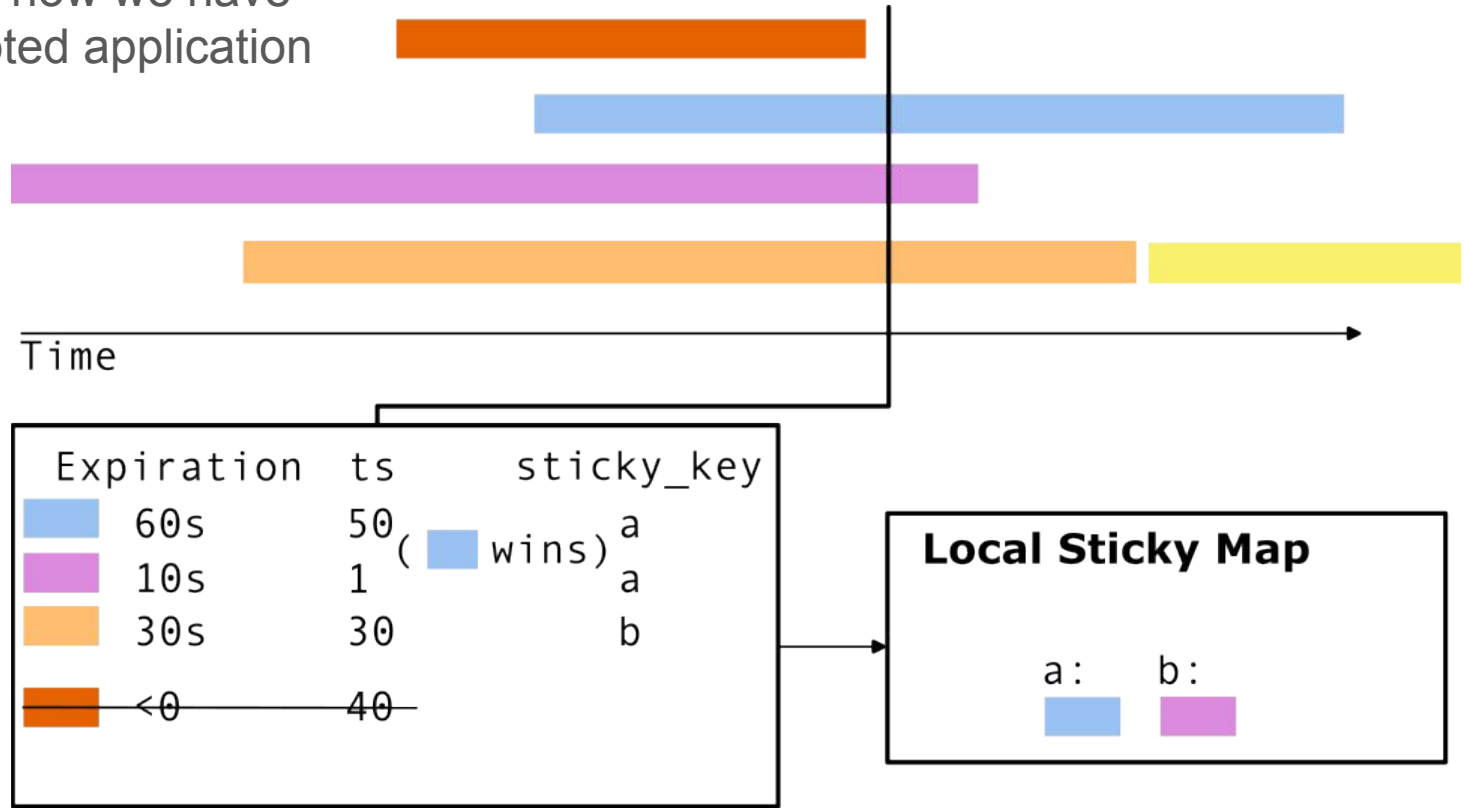
```
// event type: "m.rtc.member"
{
  "slot_id": "m.call#ROOM", // Group identity: Basketball + T-shirt color
  “sticky_key”: “xyzABCDEF0123” // same as member.id
  "application": {
    "type": "play.basketball", // Basketball
    “play.basketball.id”:”red” // T-shirt color
 },
  "member": {
    "id": "xyzABCDEF0123" // Name Tag
  },
  "rtc_transports": [],
  "versions": [],
}
```

Sticky Events



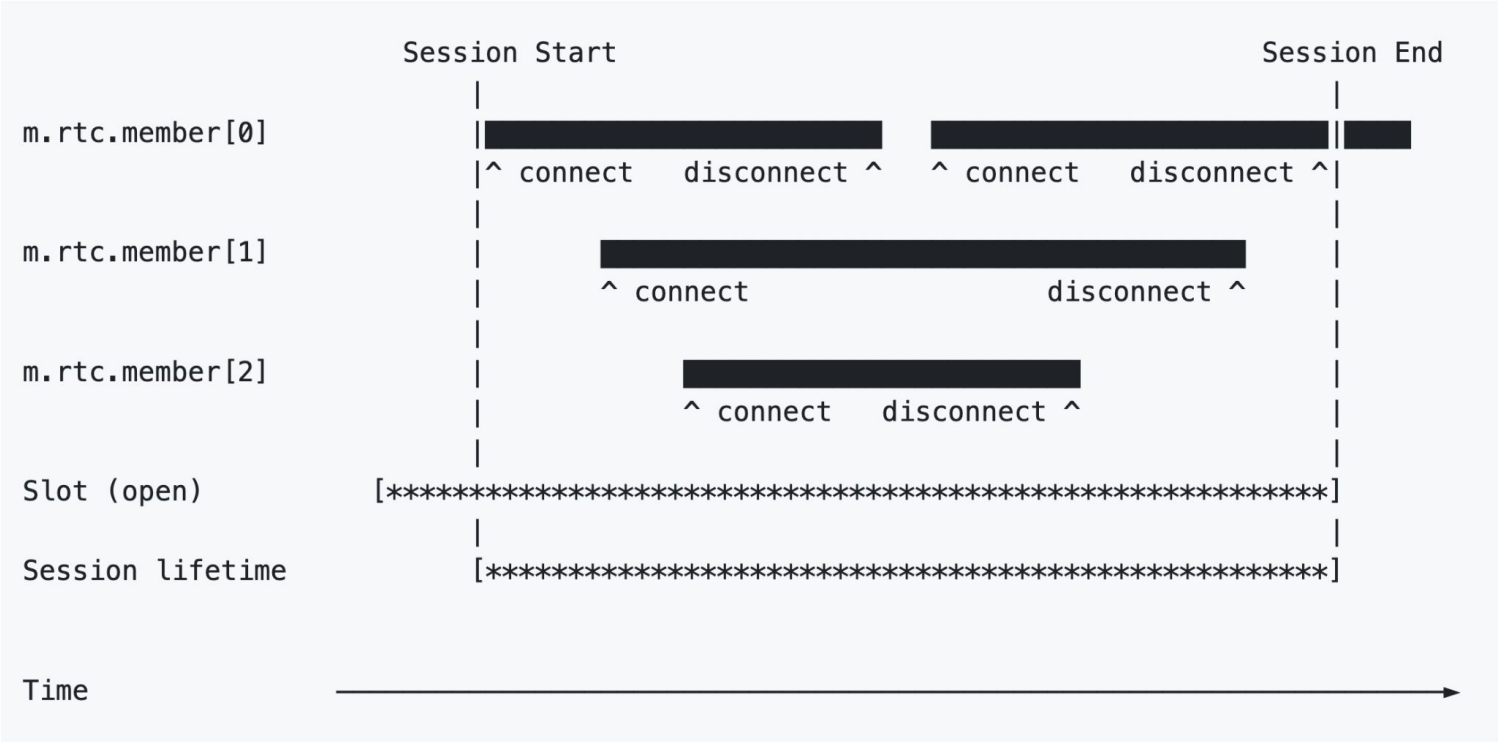
Sticky Events

This is how we have encrypted application data

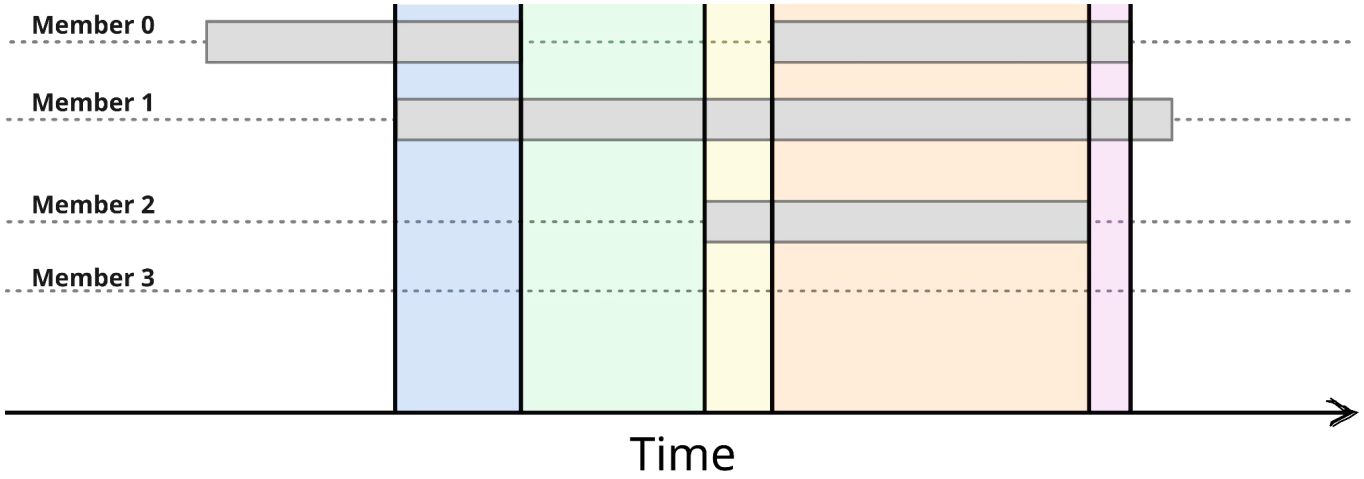


Call History

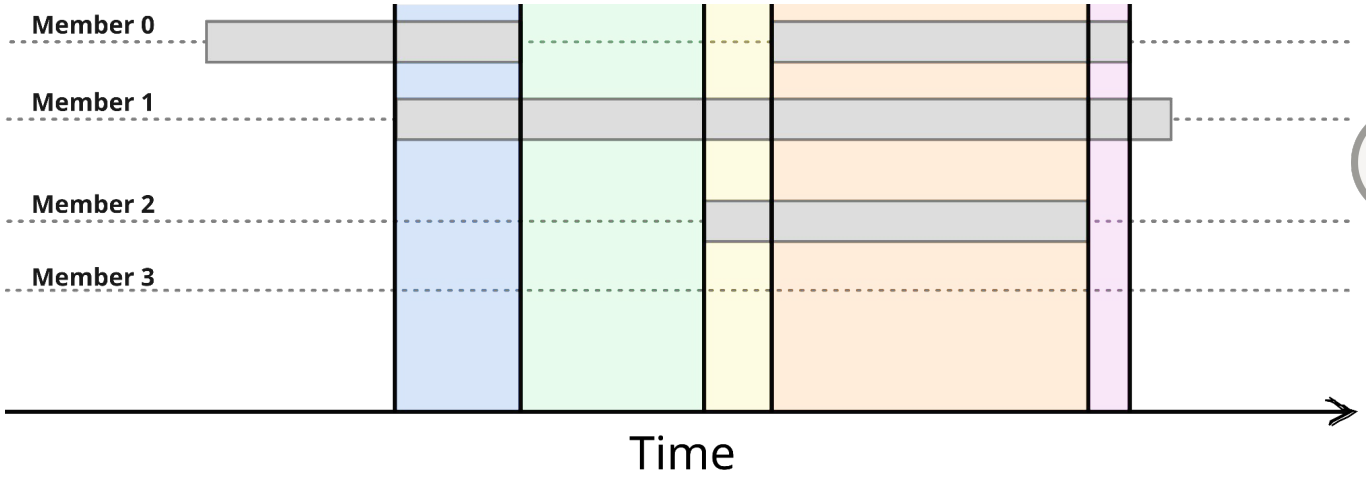
Sticky events are timeline events

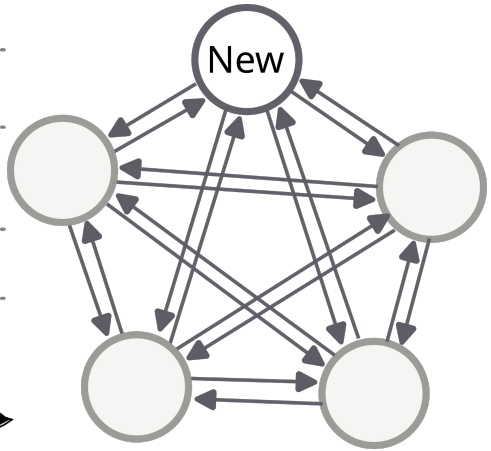


To Device Key sharing Join/Leave

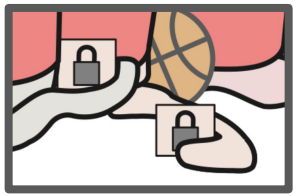


To Device Key sharing Full Mesh





- Forward secrecy
- Post compromise security
- Works well in real world



Shared secret

matrix

“Similar to password written on yard floor, encrypted”

- Everyone in the room can decrypt the media streams
- Very high performance

Slots simplify consensus (which key to use)

- Key is shared as an encrypted room event
- Slot reference key to use
- Once everyone sees same current slot event

“Shared (temporal) room secret with resolved race conditions”

(due to slot state resolution)

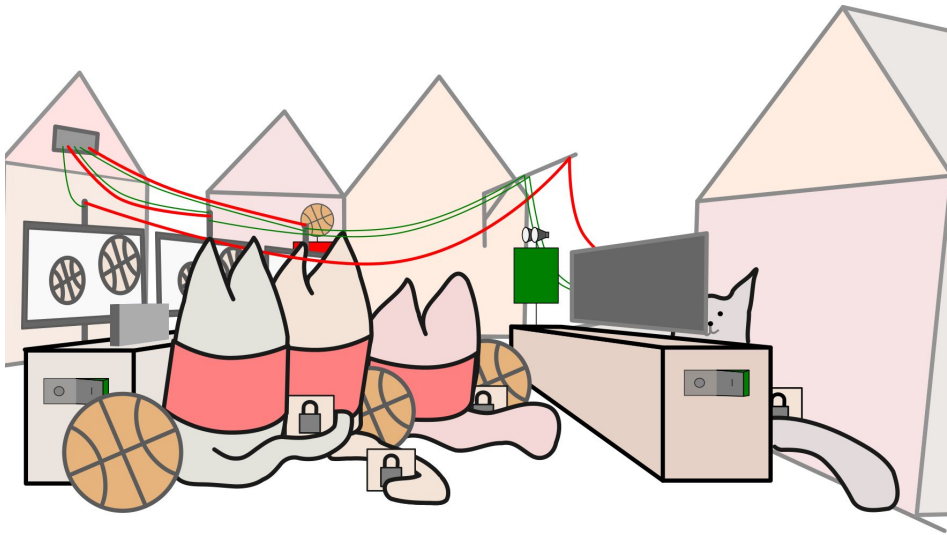
Multi SFU

New concept (replaces `focuse_active`, `foci_preferred`):

Transport

“Each member defines where/how they plan to transport their real time data to other members”

→ **No election required**

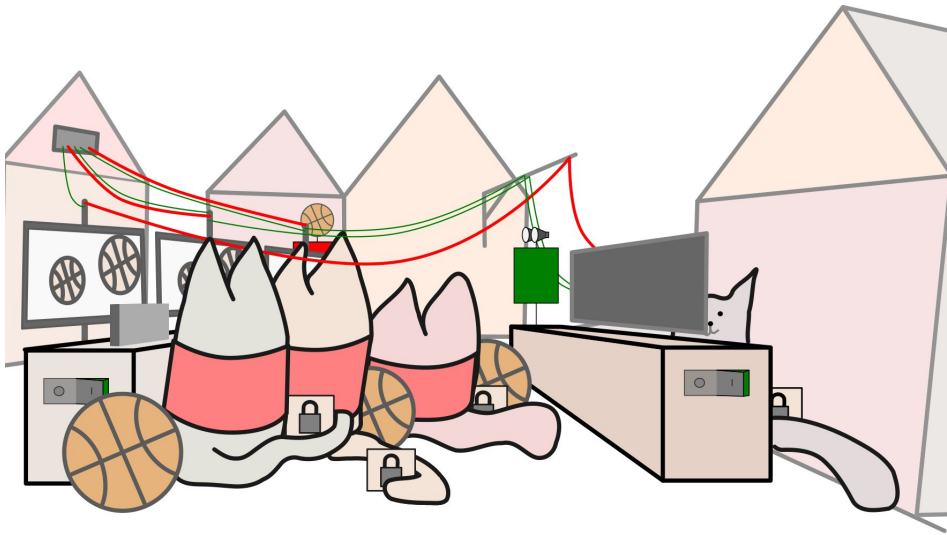


Multi SFU

“Each member defines where/how they transport their real time data to other members”

```
{
  …
  "rtc_transports": [
    {
      Type: “livekit”,
      Livekit_service_url: “https://…”
    }
  ],
  …
}
```

- Includes all data required to access media
- Can publish via multiple transports
- Can use different transport systems (livekit)
- Each application can define what transports its compatible with

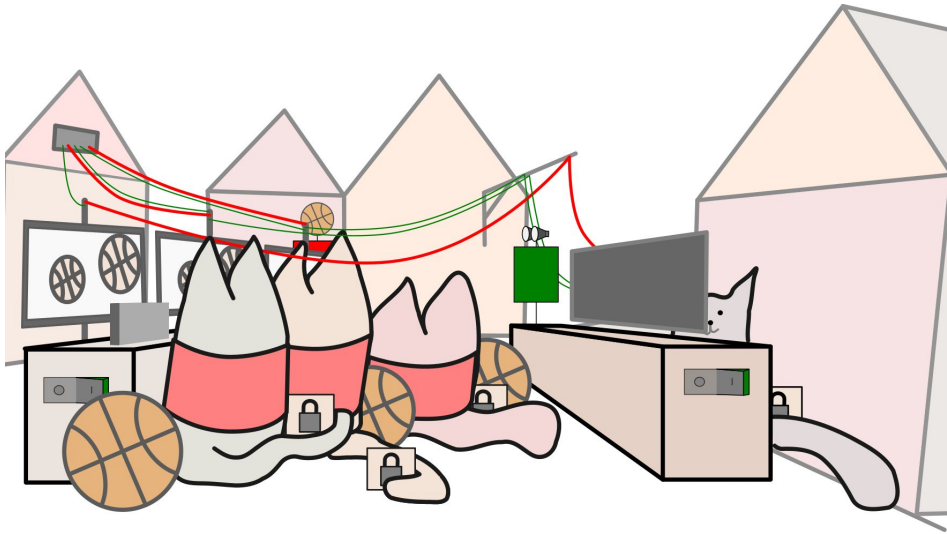


Multi SFU

Big implementation change

- Each member can use a different transport configuration!

- A client needs to connect to multiple SFU **simultaneously**
- Refactor to abstract a `Connection`
- Source of truth: membership event (Show participant and share keys based on)



Demo Matrix 2.0 Calling

Demo Add Element Call → Cinny

Call History TODO