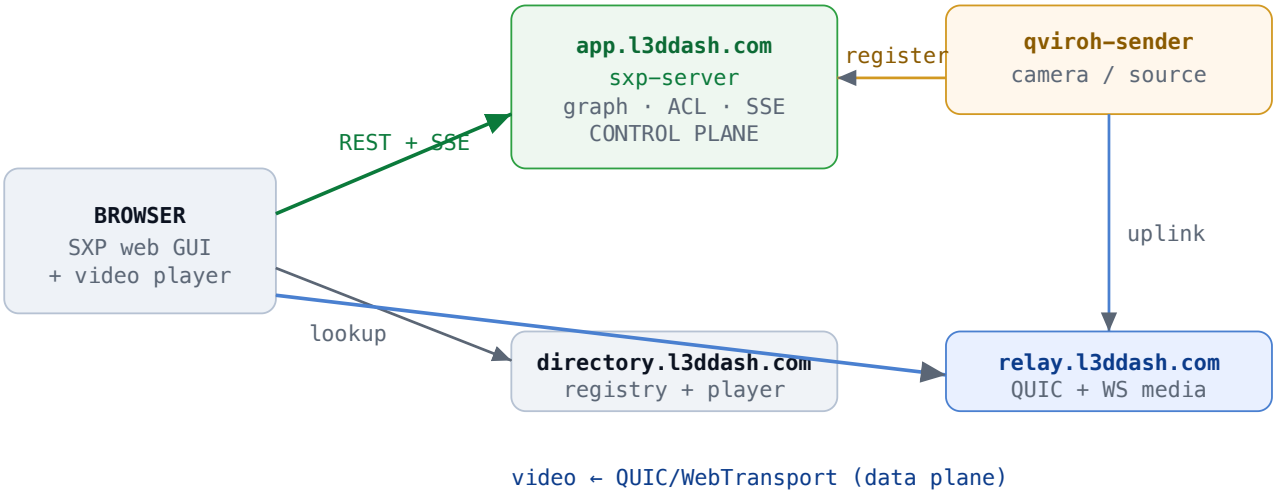
# SXP app · connection graph over QUIC

Andrew Cross's `connection.h` property graph, promoted to a live HTTP + SSE Discovery Server with a web GUI — its transport seam filled by the **qviroh QUIC/WebTransport media plane** instead of WebRTC · 2026-07-05

`connection.h` describes a distributed, access-controlled **property graph** (Devices, Connections, Groups, Discovery) over an HTTP + SSE server, with a **peer-to-peer data plane behind one seam**. The header is deliberately silent on the media — video is opaque bytes. This app makes that model real and fills the seam with **QUIC**, directly demonstrating the wedge: SXP's discovery/graph model kept intact, WebRTC's ~80 Mbit/s Windows data-channel ceiling swapped for a transport built for pro bitrates.

## Architecture — one control plane, one media plane, three servers




Green = SXP control plane (the property graph, over HTTP+SSE). Blue = QUIC media plane (the transport seam). The app server never touches the video bytes — exactly the header's plane split. The directory/relay are the same servers that already carry qviroh's video.

## The workflow the GUI walks you through

1. **Register / sign in** → the server issues a token (JWT-analogue) and the browser opens one **SSE stream** = the reactive core.
2. **Create a Device** (a graph node). Give it a `source` to bind it to a live qviroh feed (e.g. `remote`). Its id is a deterministic `SHA-256(user:host:user:type:name)`.
3. **Create a Group** (an access-control list) and **add your device** to it.
4. **Grant a user access** to the group (paste their user id). Only members/owners may discover it — a non-member is denied `403`.
5. **Create a Discovery** on the group → a live, SSE-backed list of visible devices (`sources_changed` keeps it current).
6. **Create a Connection** between two devices. It **activates** the moment the caller can see both endpoints, emitting `connection_opened`.
7. **Video flows** — for an active connection to a data-capable device with a source, the GUI plays it over the **QUIC media plane**. That is the transport seam, filled.

**The point in one line:** every arrow above except the blue one is Andrew's header, faithfully. The blue one is where WebRTC used to be — now it's QUIC, and the video you see in the GUI proves the swap is transparent to the graph.

## Run it — the hosted demo (nothing to install)

### Solo (one browser tab)

Open **https://app.l3ddash.com** → **Register** → click ⚡ **Quick demo**. It creates a viewer and a camera bound to the live `remote` feed, connects them, and plays the video over QUIC.

### Two users (two tabs / two machines)

A. **Alice** registers, creates a camera device with source `remote`, creates group `studio`, and adds the camera to it.

B. **Bob** registers and copies his **user id** to Alice.

C. **Alice** pastes Bob's id into *add_user_to_group*, then sends Bob the group's id (⧉).

D. **Bob** pastes the group id into *create_discovery* — Alice's camera appears live. Bob creates his own viewer, connects it to the camera, and the video plays.

## Run it — locally from source

```
# control-plane graph is the sxp-stub crate; the server is sxp-app
cd sxp-app
cargo run --release -- --bind 127.0.0.1:8090 \
   --media-directory https://directory.l3ddash.com \
   --media-relay https://relay.l3ddash.com
# then open http://127.0.0.1:8090

# the underlying graph model, as a narrated tour + tests:
cd sxp-stub && cargo run --example walkthrough && cargo test
```

## HTTP + SSE API (what the GUI calls)

| Method / path | connection.h | Does |
|---|---|---|
| `POST /api/register` | Admin::create_user | issue user id + token |
| `POST /api/session` | User::open_user | open session; start its SSE bus |
| `GET /api/events?session=` | the SSE stream | device_changed, connection_opened/closed, sources_changed, group_changed, data_receive |
| `POST /api/device` | create_device | node; id derived from `name` |
| `POST /api/group` | create_group | access-control list |
| `POST /api/group/add-device · add-user` | Group::add_* | grant membership (the ACL) |
| `POST /api/discovery` | create_discovery | subscribe to a group (403 if no access) |
| `POST /api/connect` | create_connect | edge; activates when both endpoints visible |
| `POST /api/json` | set_public/private_json | mutate metadata (only public rides the bus) |
| `POST /api/data` | data_send | route a payload over an active connection |

SXP app · sxp-stub (graph model) + sxp-app (HTTP/SSE server + GUI) · media plane = qviroh QUIC/WebTransport · control plane on app.l3ddash.com, media plane on directory + relay.l3ddash.com · engagement context in `reference/VISION-MAP.md`.