Funkfeuer Vienna's v642 Project

Our journey towards IPv6 and OLSRv2 – Battlemesh v10

David Hopfmüller <david@hopfmueller.at>

2017-06-09

About Funkfeuer

What's with the name?

Funkfeuer is the German word for *radio beacon*. Our founding members liked the analogy of their nodes announcing a free network out to the world.

`0xFF` serves as a handy abbreviation.

History

Funkfeuer started around 2003 (nobody really remembers the exact date). A commercial local ISP had experimented with WLAN on the last mile, and decided to abandon the concept (and the installed hardware).

Funkfeuer was founded and took over the network.

Funkfeuer grew up

In the early 2000s, Funkfeuer saw much growth: A good mix of available hardware (WRT54G FTW!), easy-to-use OpenWRT firmware and an unmatched offer (at that time): high-speed Internet just at the cost of hardware.

Just a few years later, Funkfeuer had grown to **more than 250 nodes all across the city of Vienna**. In addition, similar networks were started in other Austrian cities.

Funkfeuer node – looks familiar, eh?

Figure 1: Node `ho6`. Photo: Markus Gschwendt 2016

Organizational setup

Funkfeuer is set up as a **registered society** under Austrian law. We see several benefits with that approach:

- A legal entity makes it much easier to deal with other parties (contracts, sponsoring, interaction with government bodies, ...) compared to a bunch of (varying) individuals
- Well-defined (and commonly well-known) structure for member participation

Internet uplinks

In contrast to other community networks, Funkfeuer uses **central Internet uplinks** (2x 1G currently). There was an opportunity to directly peer at VIX (Vienna Internet Exchange), which provides quite reasonable^{TM} upstream connectivity.

Financing

In order to pay for uplinks and other running costs, Funkfeuer decided not to only rely on sponsors. Instead, **we started a community housing project.** Folks can house anything from RPis to 19" servers at a small fee.

RPI-Housing

Yes, you did get that right! We just recently announced availability of this long-asked offer:

- One RPi-sized computer (you'll see why)
- 5 V only, 8 W typical
- Remote reboot
- EUR 5 per month (yearly upfront payment, to keep things maneagable)

RPi housing

Figure 2: RPi housing. Photo: Wolfgang Nagele

Switchero remote power management

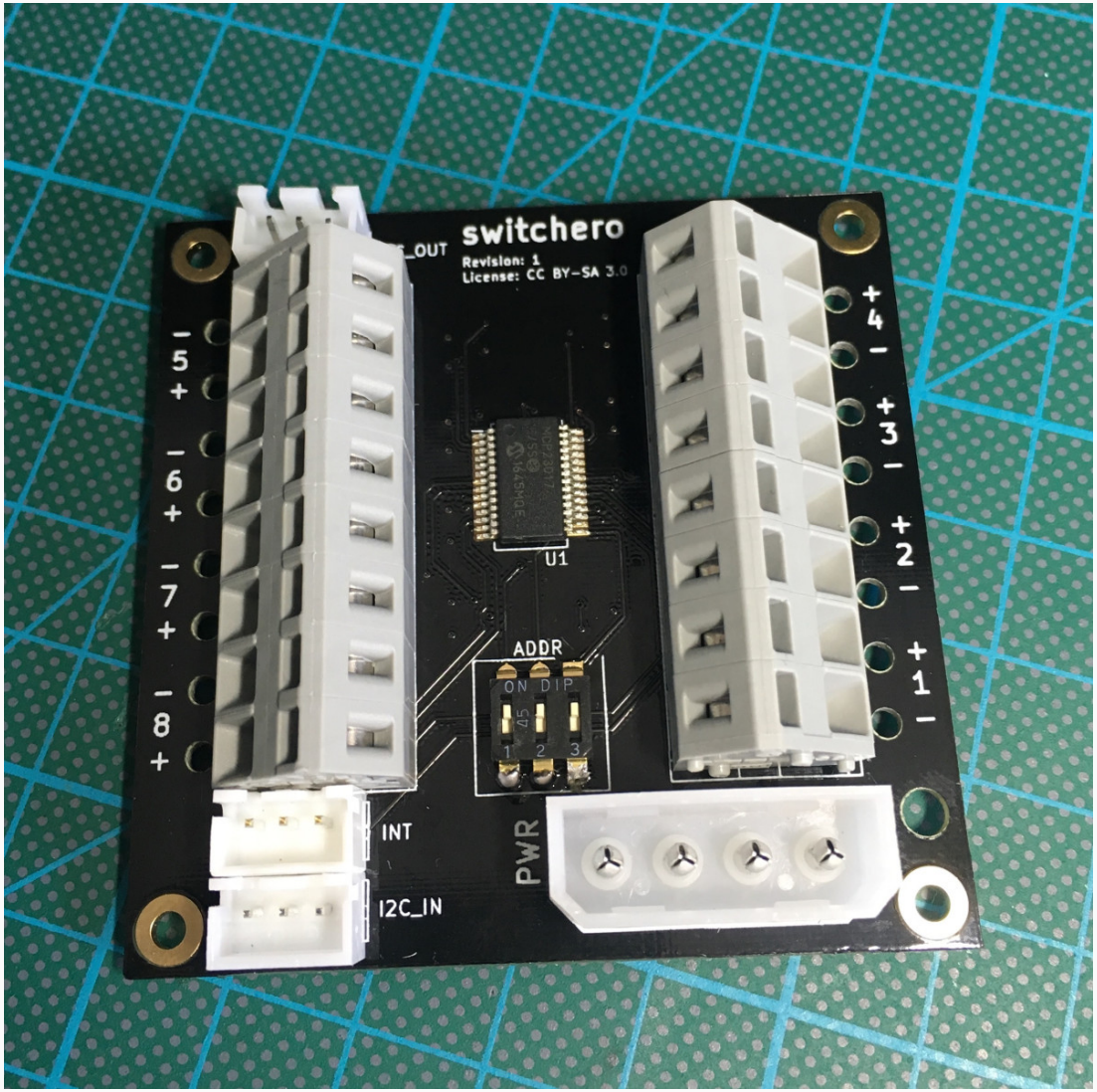

Figure 3: Switchero board. Photo: Wolfgang Nagele

Our traditional setup

- OSPF at the (uplink) edge
- OLSR (v1) as the routing protocol
- IPv4 everywhere
- IPv6 as an experimental setup (separate OLSR instance)

This setup worked quite well for us for the past 10+ years.

Bring back that experimental feeling!

It was in 2016 that several of us shared very similar thoughts:

- It's 2016, and we're still stuck with IPv4 (mostly)
- OLSR v1 served us well, but v2 is out there for good reasons
- It hurts on the lower layers as well: IBSS is dead, Master/Slave isn't that great either

v642 was conceived

We called our new baby **v642** for a reason (actually, for three reasons):

- IP**v6** is to become our primary network protocol
- Despite all efforts, IP**v4** is going to stay for many years, so we need a transition mechanism
- OLSR**v2** should replace v1

IPv6 considerations

In terms of IPv6, our goals were:

- Auto-configuration of a node's main IP address and the OLSRv2 identifier using EUI-64 based on a MAC address
- Exclusive use of link-local IP addresses for routing protocol communications
- IPv6 address space for each node. This resulted in a /48 per node. Why? Because we can!

IPv4 transition mechanism

As said, we knew that we will need to provide IPv4 connectivity in some way. We definitely wanted to avoid two OLSR instances.

Among the many choices of translation mechanisms, we decided to go with 4in6, that is plain IPv4 packets in plain IPv6 packets.

Reasoning behind 4in6

Some of the reasons to go with 4in6:

- Broad support through Linux kernel implementation (at least so we thought...)
- Stateless tunnels make sense with lossy networks
- Easy and straightforward configuration

Ubiquiti's 6in4 implementation

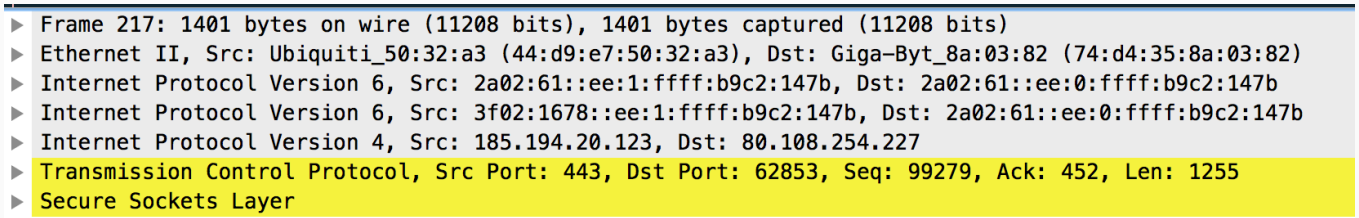

Figure 4: Ubiquiti's 6in4 implementation. Screenshot: Wolfgang Nagele

Thanks for joining!

Questions?