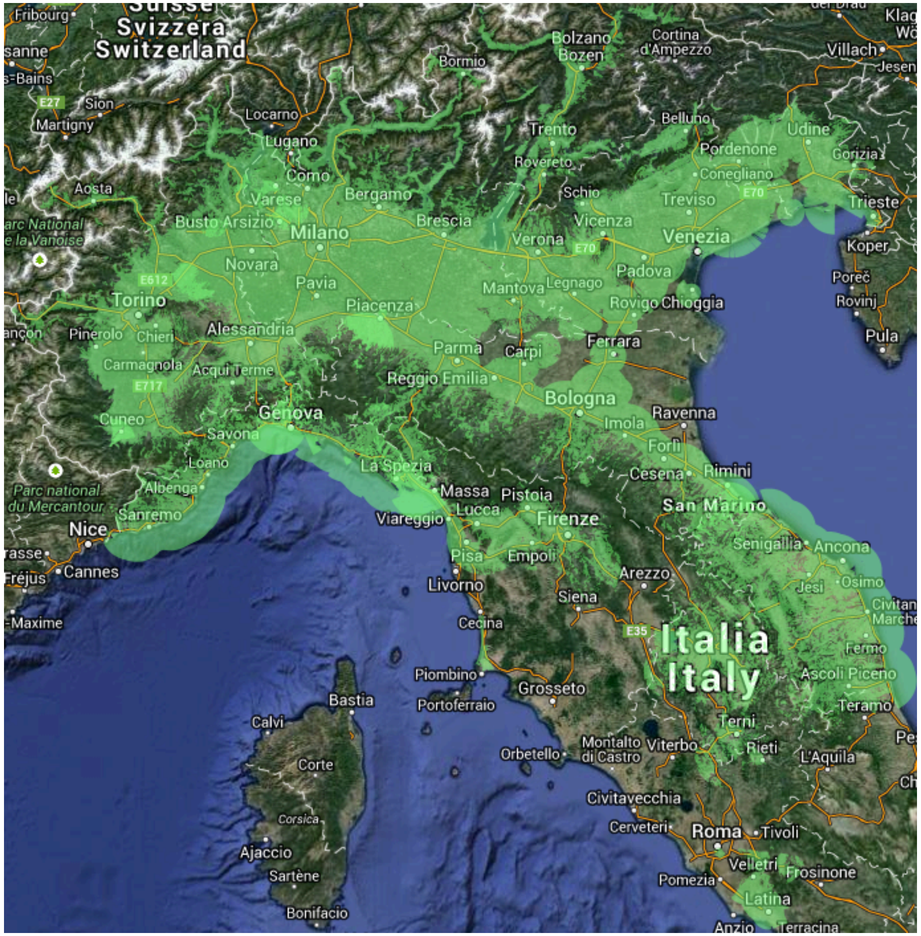
SDN For Real

Giacomo Bernardi, CTO
NGI SpA, Italy

Who are we?

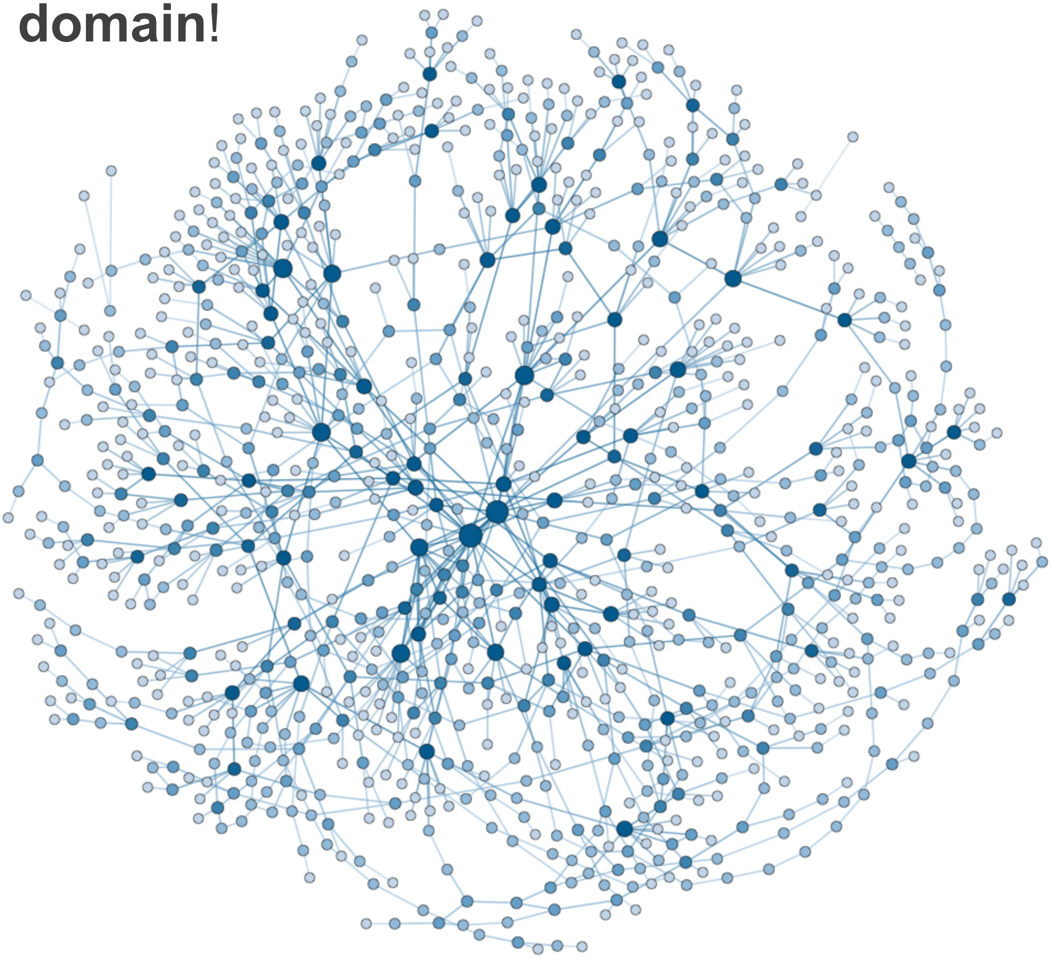
- NGI is an Italian **Fixed Wireless Access (FWA)** carrier.
- Dual-play services from **10Mbps to 1Gbps**, also as white-label wholesale.
- Currently 130,000 customers, on **1,200 radio towers** in North and Central Italy.
- Each month:
  - 4,500 new customers
  - **~100 new towers**



The shocking news

- Our network was **a single L2 domain**!
  - …indeed, **it was a single broadcast and PVST+ domain**.

- If we didn't take action, we'd **very soon** hit a brick wall!



A “particular” network

- Virtually all of our backhaul is wireless (only ~70 DWDM links)

- Thanks to volume discounts on hardware and frequency licensing, the incremental cost for new microwave **PTP link is low**:
  - As a consequence, we often link two towers just because they are in line-of-sight.

- We ended up with a `link/POP` ratio of **2.2**, and a network diameter of **29 hops**.

- We are still very inefficiently exploiting our high mesh factor!

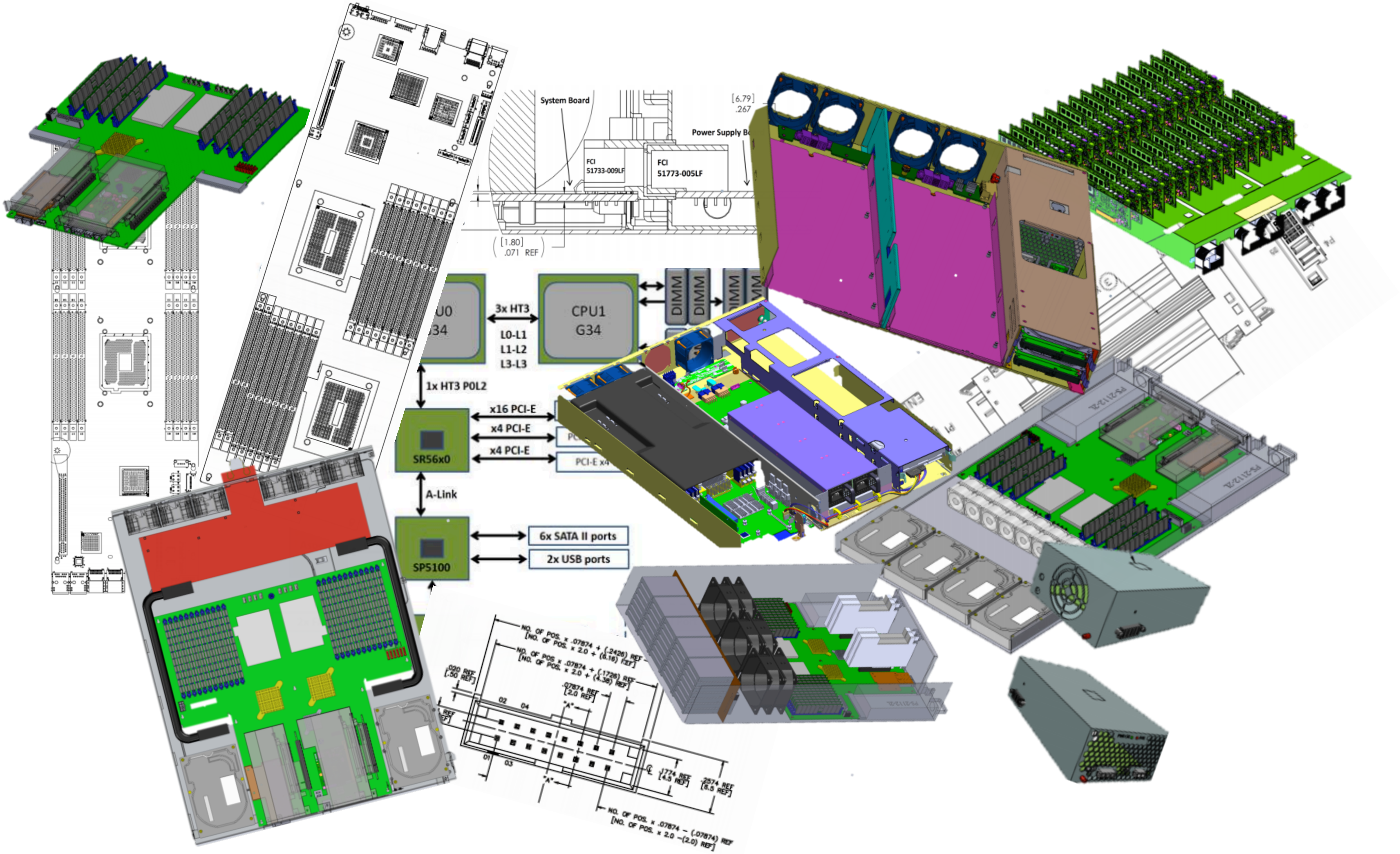
What’s off the shelf?

- We summoned “**The Vendors**”:

- Their proposed network architectures were based on static pseudowires over IGP protocol, or with some sort of network automation to provide redundancy.

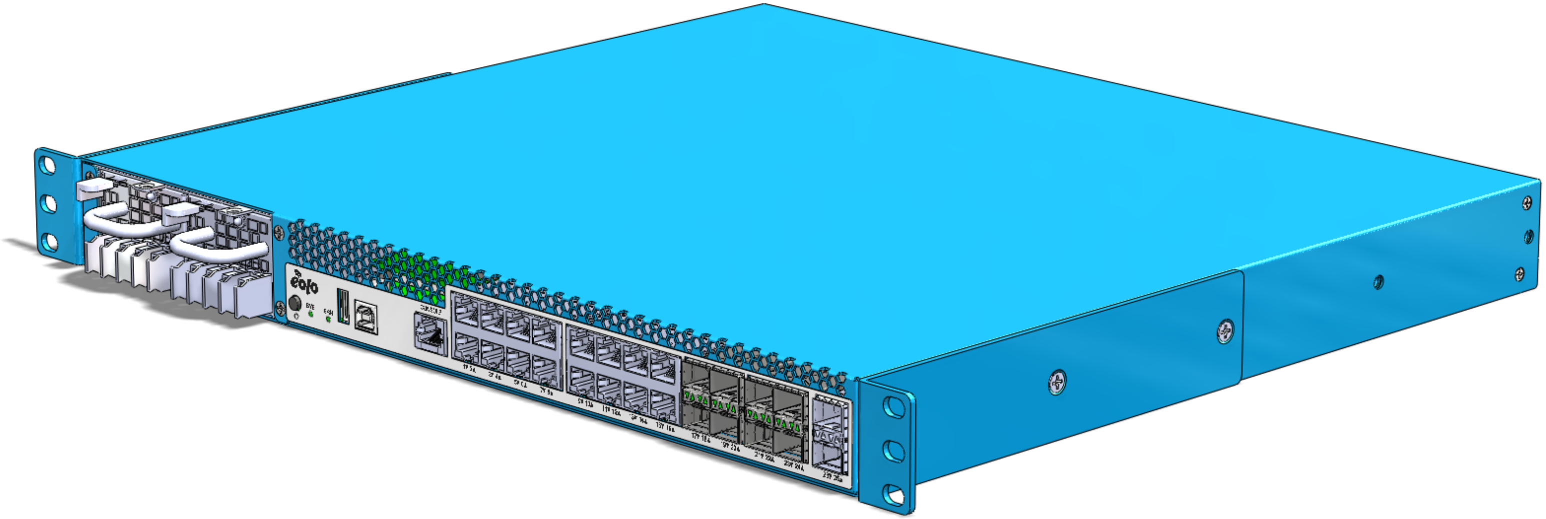
- **Suboptimal because:**
  1. They cannot exploit multipath-load balacing
     - and we have a lot of multi-path!
  2. They don’t scale
     - a single OSPF Area 0 with 10,000 routers?
  3. We would be forced to segment the network in regions that wouldn’t easily communicate

Is networking so special?

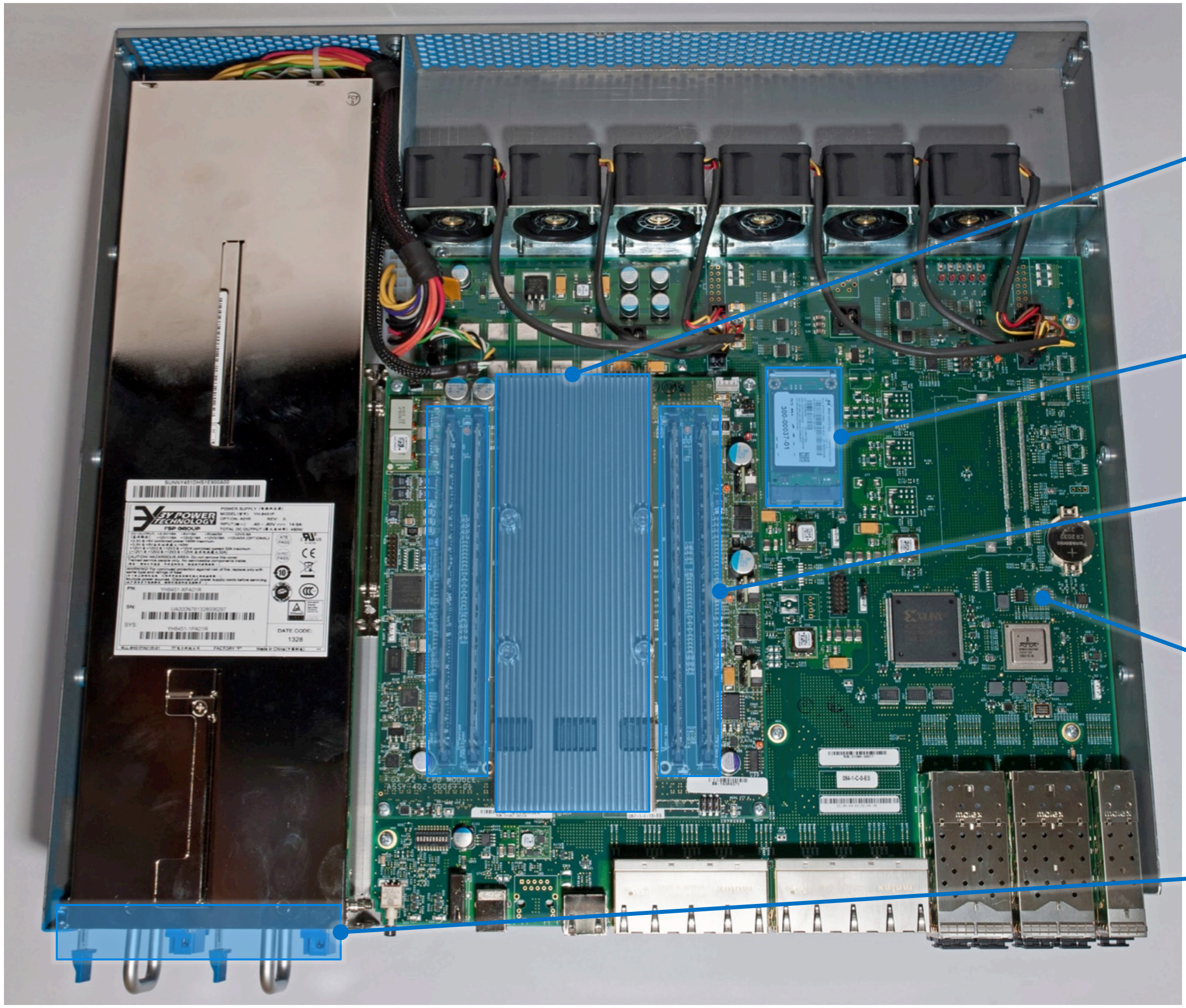


Yeah, let’s reinvent the wheel!

- We agreed with an OEM to build **our own “tower router”** in a few thousand units.



Yeah, let's reinvent the wheel!



Currently based on ~~Tilera's~~ EzChip's **TileGX 72-core CPU** architecture

128GB SSD storage

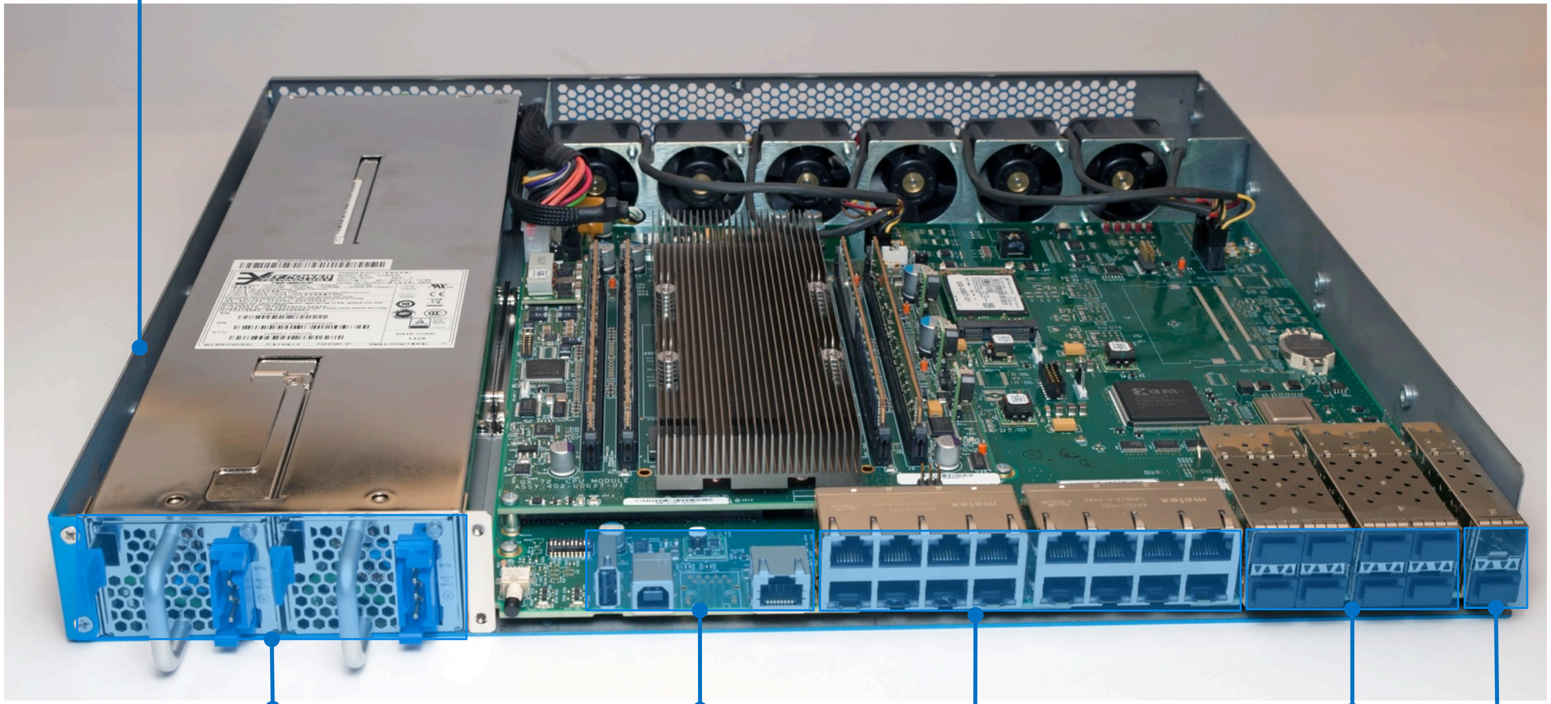
16GB RAM
4x DDR3 controllers

Extensive hardware monitoring

Low-power: ~150W in the worst scenario we tested, typical ~90W

Yeah, let’s reinvent the wheel!

Only 40cm deep. All cabling on the front side.



Dual DC or AC PSU

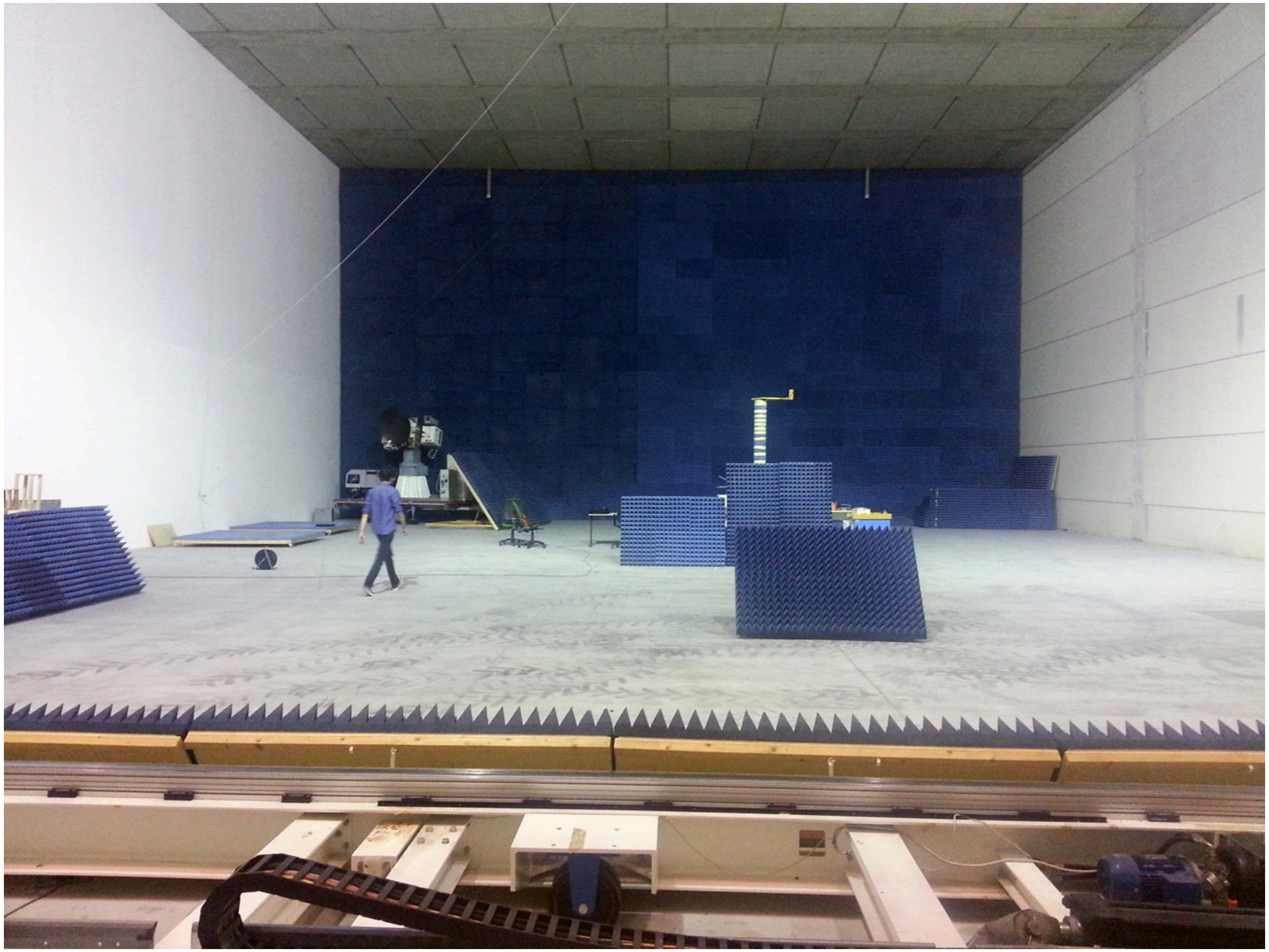
USB/RS232

8x 100M/1G SFP

16x 10M/100M/1G RJ45

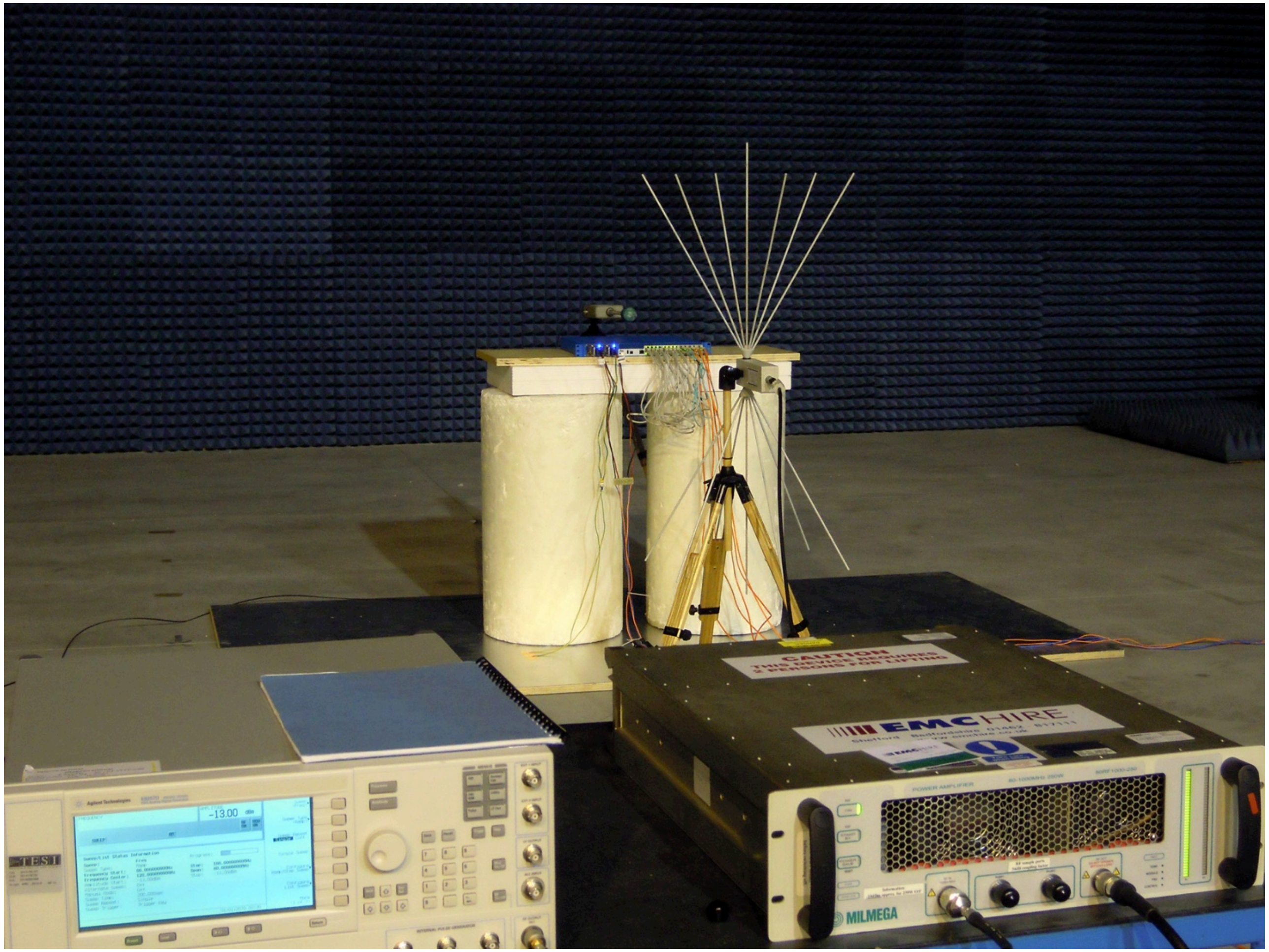
2x 1G/10G SFP/SFP+

Testing the wheel



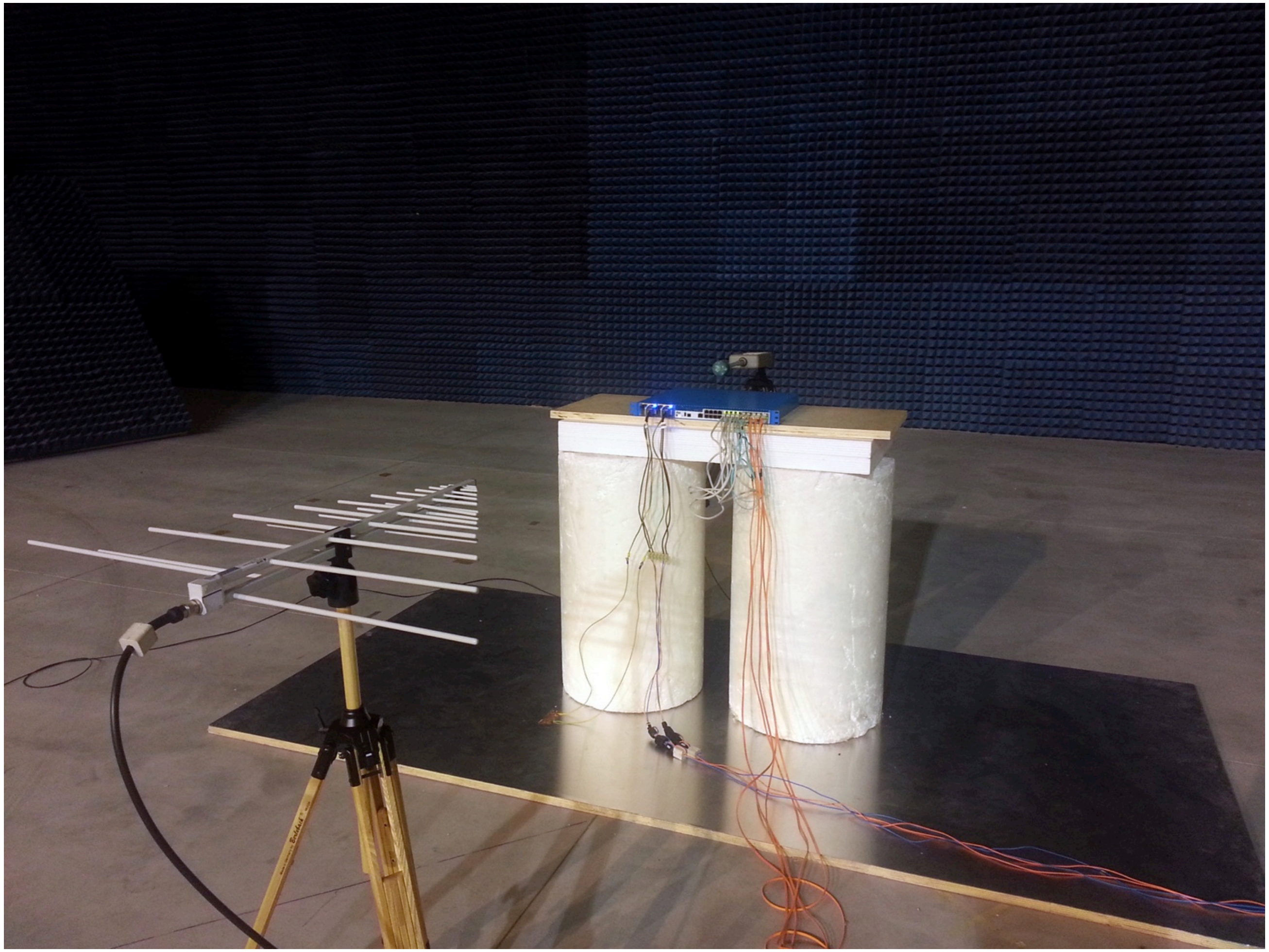
- We brought the first prototypes to a lab for EMC, Safety, Environmental and IEEE-1613 testing.

Testing the wheel



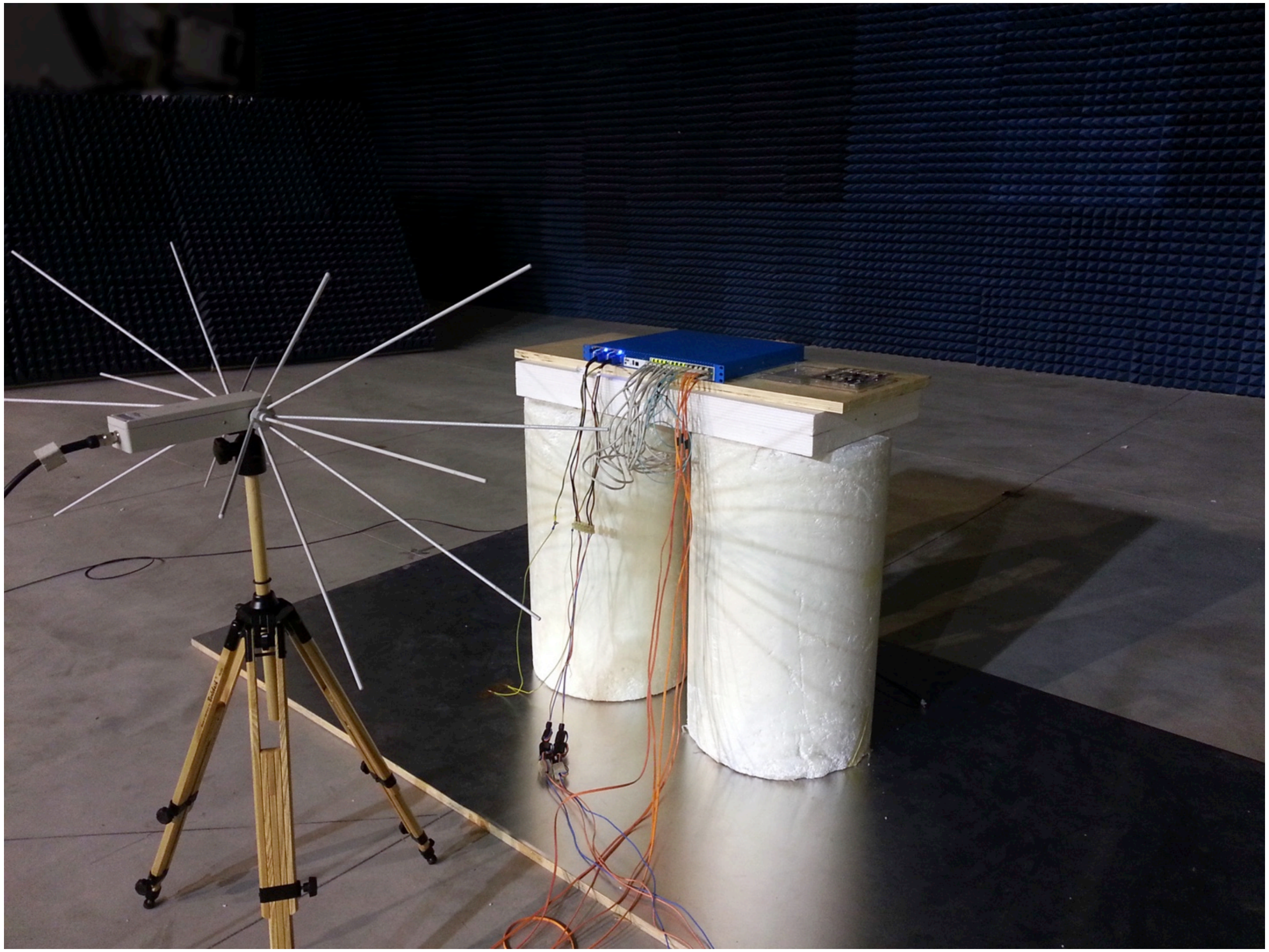
- We brought the first prototypes to a lab for EMC, Safety, Environmental and IEEE-1613 testing.

Testing the wheel



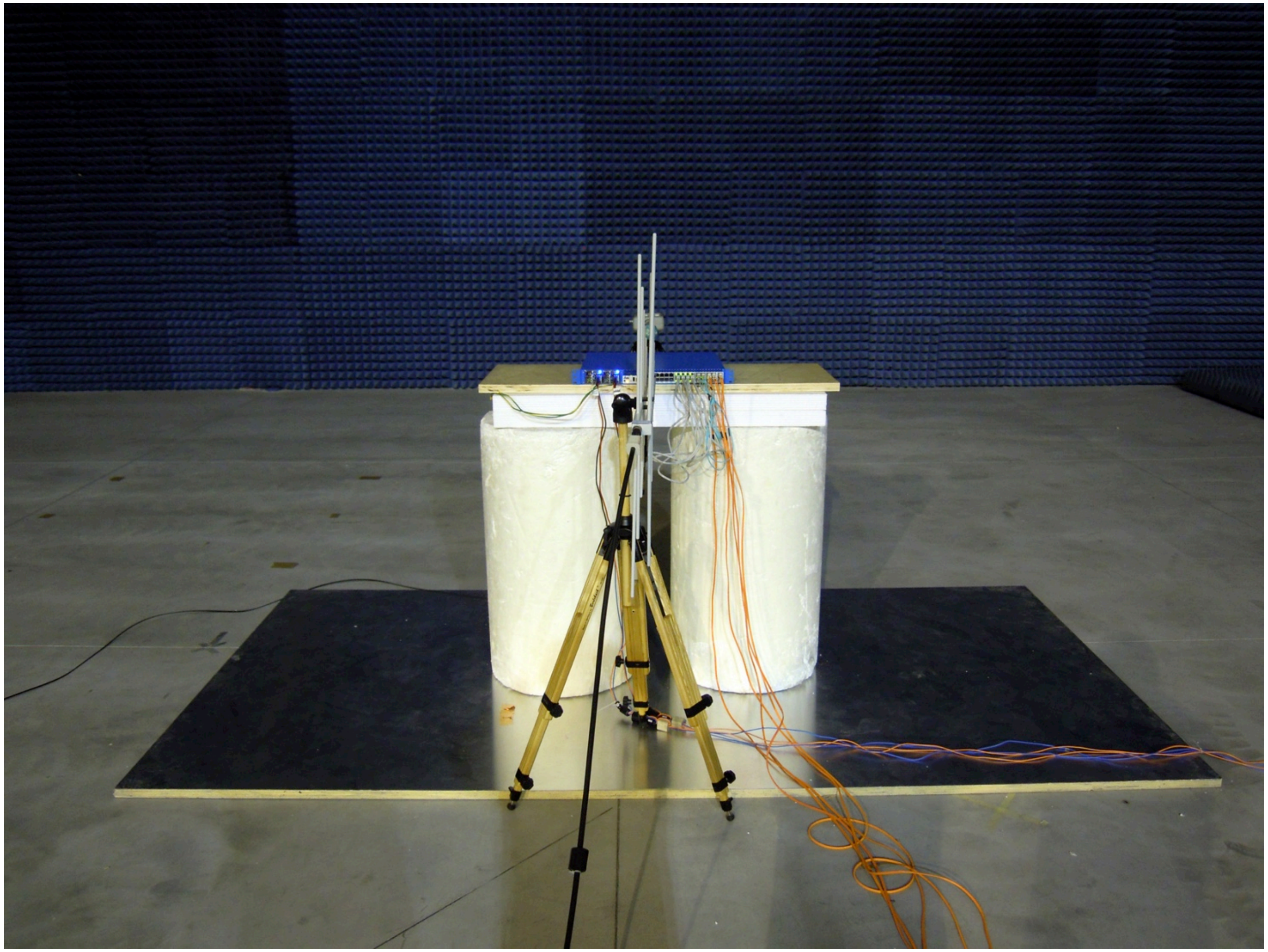
- We brought the first prototypes to a lab for EMC, Safety, Environmental and IEEE-1613 testing.

Testing the wheel



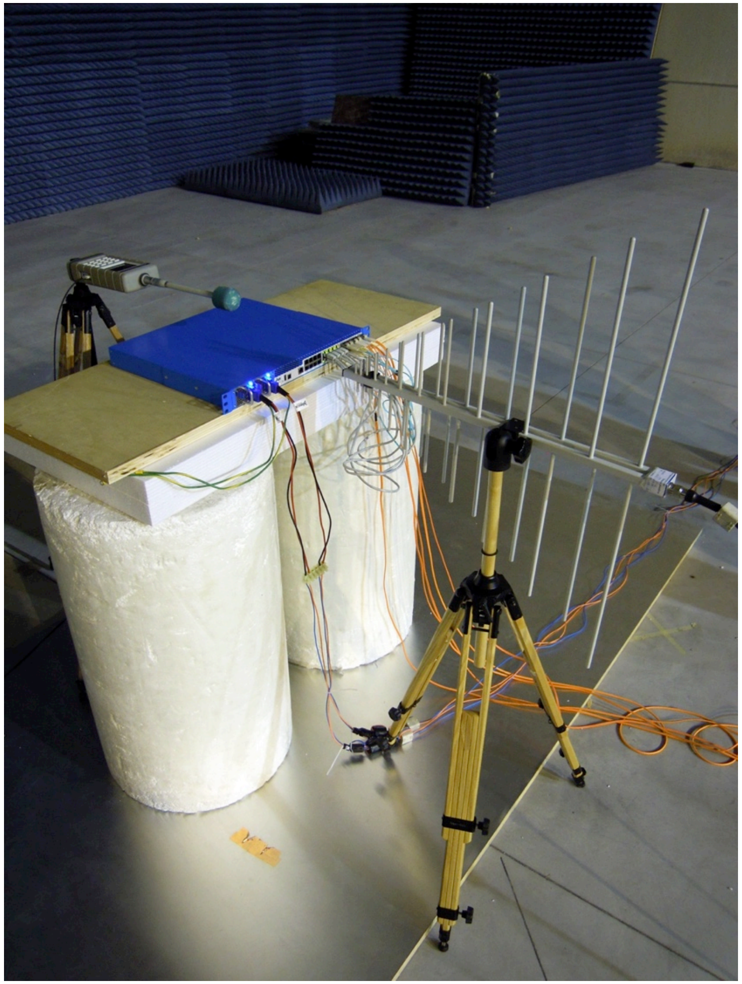
- We brought the first prototypes to a lab for EMC, Safety, Environmental and IEEE-1613 testing.

Testing the wheel



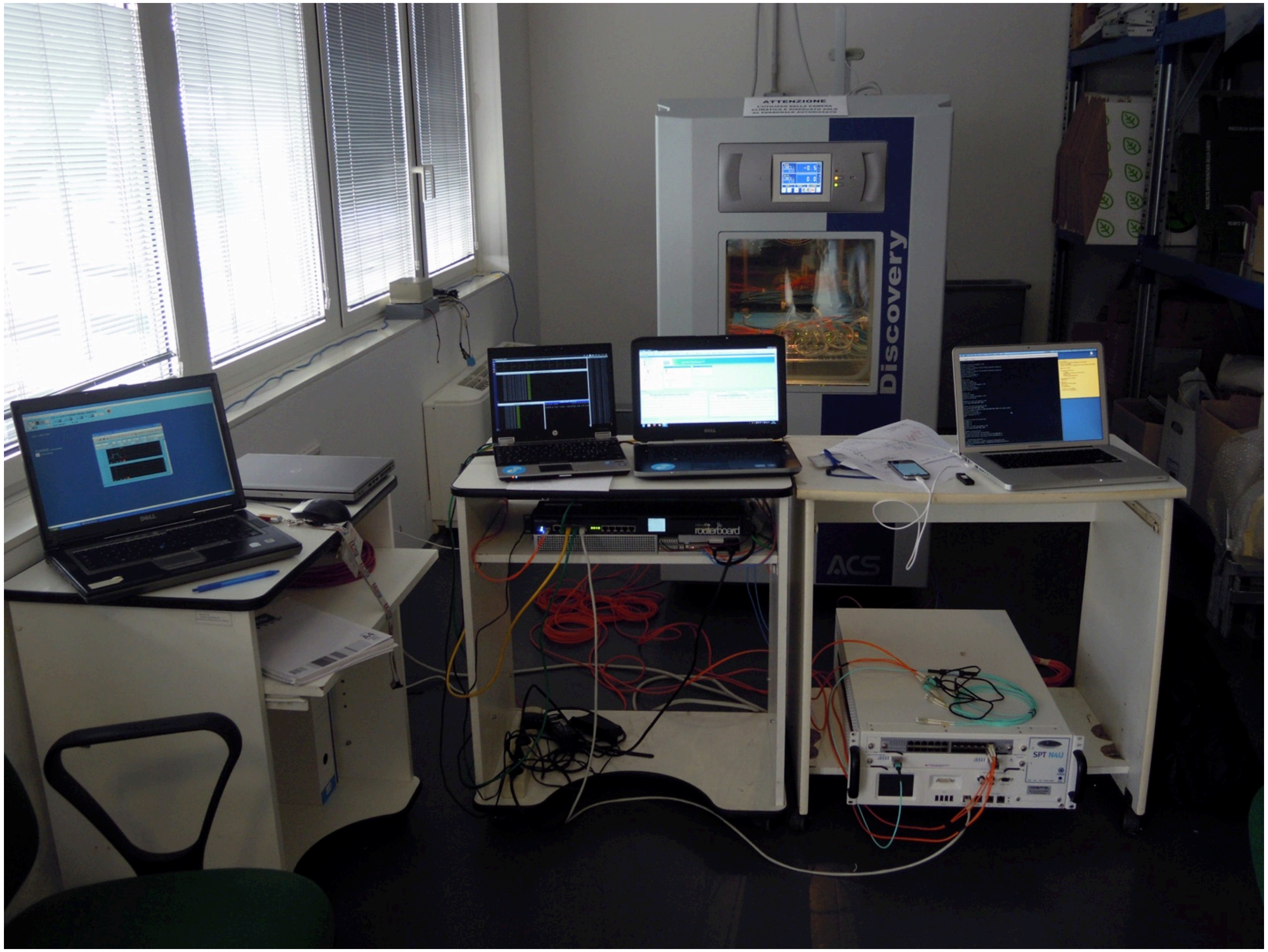
- We brought the first prototypes to a lab for EMC, Safety, Environmental and IEEE-1613 testing.

Testing the wheel



- We brought the first prototypes to a lab for EMC, Safety, Environmental and IEEE-1613 testing.

Testing the wheel

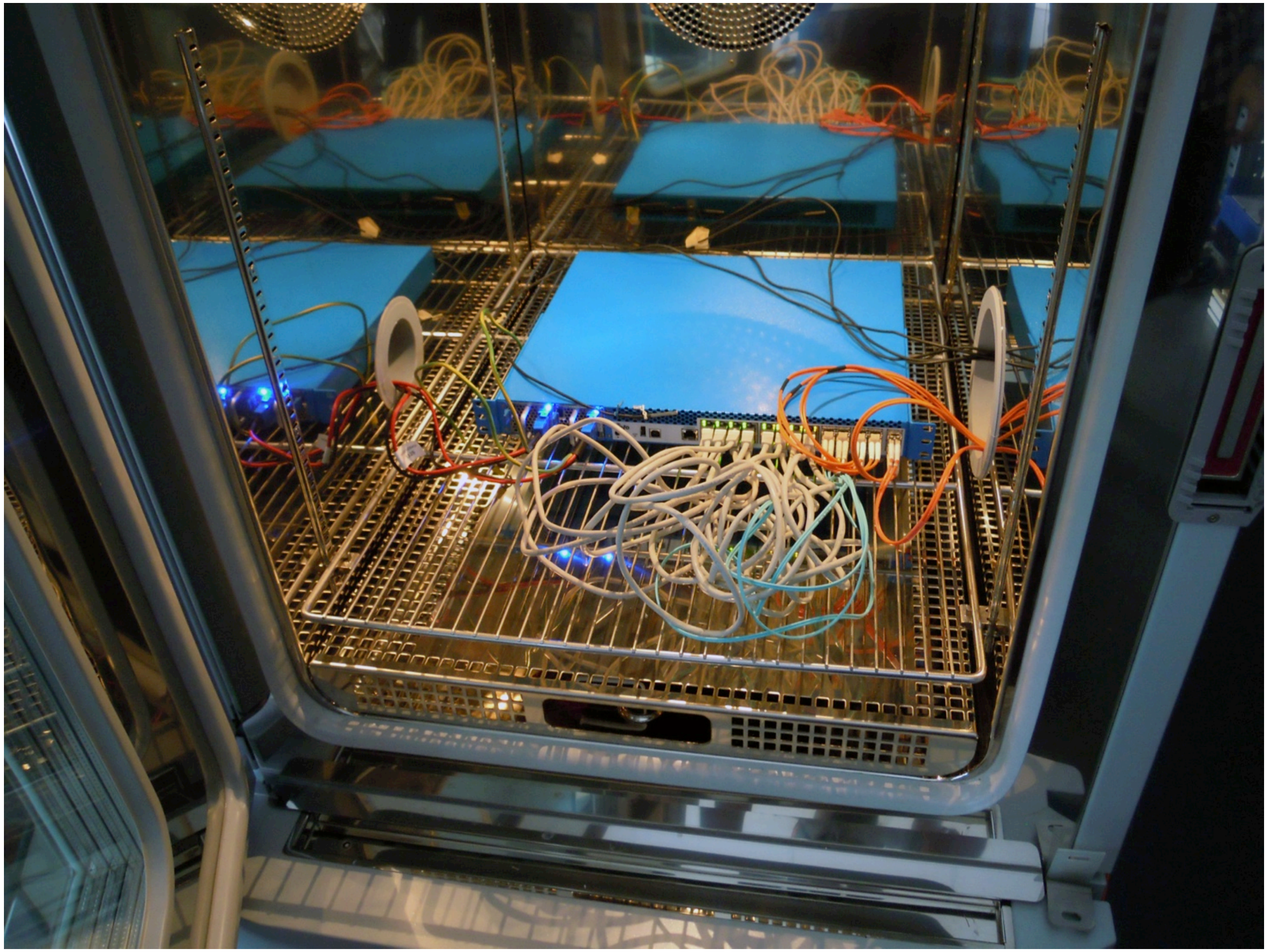


- Managed to ensure error-free operation in **-15C/+75C** environment, and in presence of EMI fields of up to **180V/m**. Yeah!

Testing the wheel

- Managed to ensure error-free operation in **-15C/+75C** environment, and in presence of EMI fields of up to **180V/m**. Yeah!

Testing the wheel



- Managed to ensure error-free operation in **-15C/+75C** environment, and in presence of EMI fields of up to **180V/m**. Yeah!

Testing the wheel

- We even had the device and its packaging tested.

Software architecture

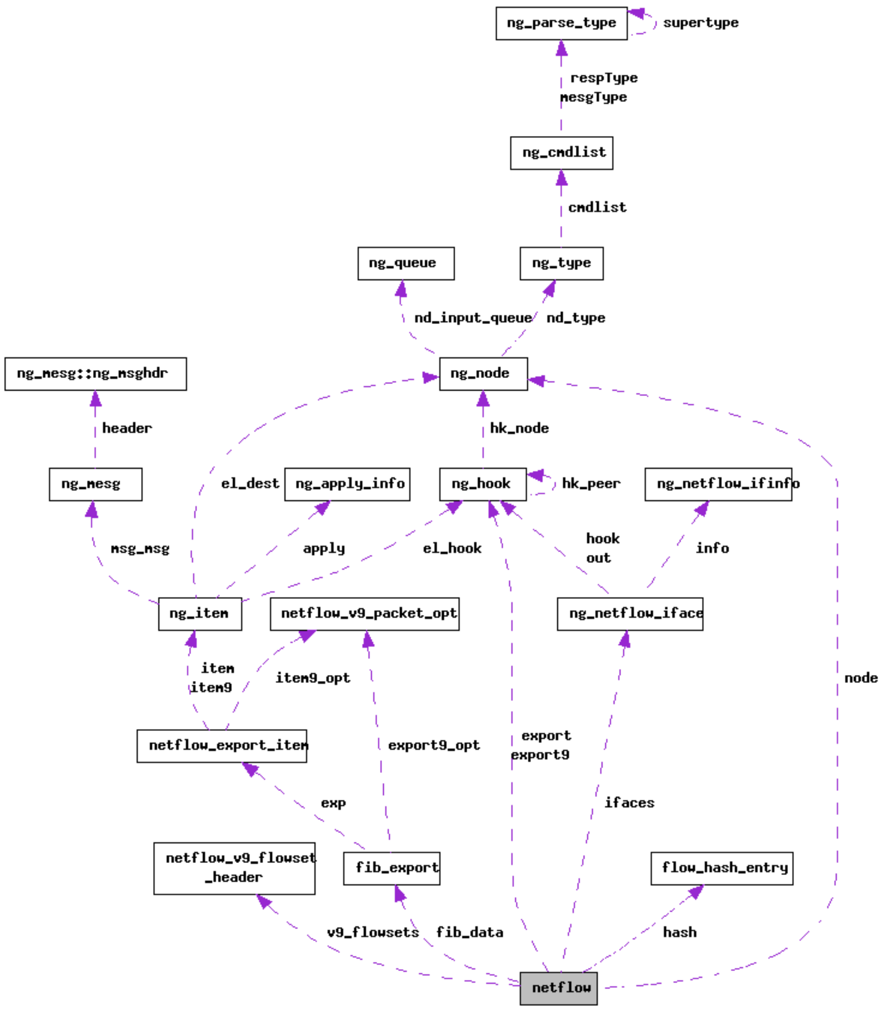
- Linux based, but we are **not** using Linux's kernel space network stack for the dataplane.

- We partered with **6WIND S.A.** (the people behind the Intel DPDK), to license their userspace network stack and use it on our routers.
  - **Simple idea:**
    - Each port has a dedicated pool of CPU cores to handle its packets.
    - Each core runs a single userspace process implementing the nework stack.
    - The kernel scheduler is instructed to ignore these CPU cores.
  - **Pro:** great performances, in our scenario we get full-duplex linerate on all ports (~180Mpps), leaving headroom for future apps.
  - **Con:** it "taints" the kernel out of GPL, limiting the extent of software we can release to the public domain.

Software architecture

- We use a concept similar to the **BSD Netgraph**.

- Other stuff we need:
  - **Open vSwitch**, which we tested with up to 1M flow rules.
  - BGP and OSPF (Quagga)
  - **PPPoE**,
  - **DHCP** relay,
  - **BFD**,
  - …



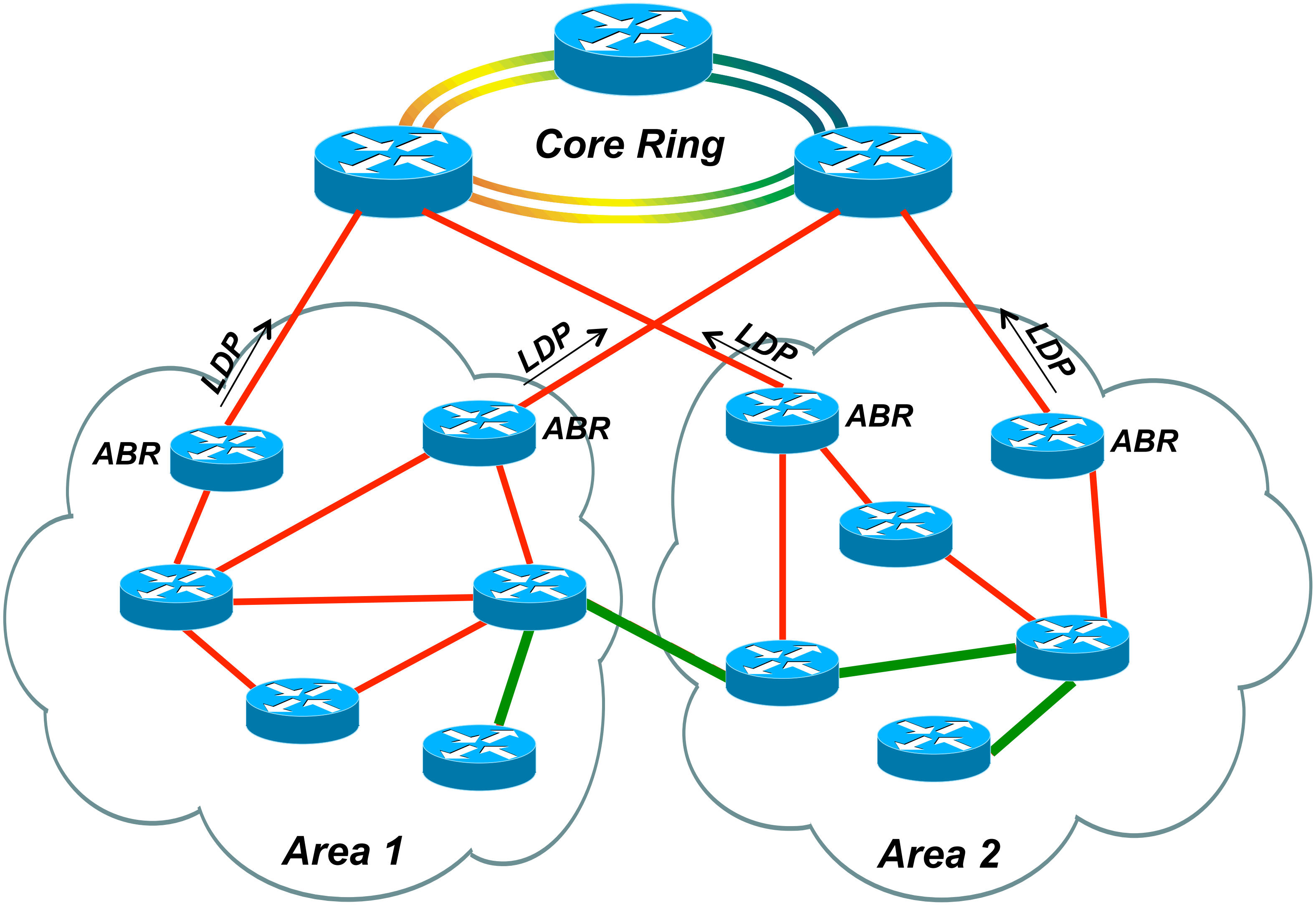
Multipath on steroids

- We operate an **almost-completely wireless network**:
  - Our FDD links guarantee **low latency** (~200µs/link OWD)
  - But their network capacity may vary with changes in modulation

- Roughly **75% of our traffic is "delay unaware"**, i.e. the user's experience doesn't degrade with tiny (e.g., < 5ms?) increase in latency.

- In case of regional unexpected saturation, we can route part of such traffic over a slightly longer path.
  - **…and we have many of those paths!**
  - Classification does not need to be perfect, and can be tweaked.

The new network topology



Thank you!

giacomo.bernardi@ngi.it

eolo

Backup slides

Functional principles

- Each POP is assigned a globally unique 4-digit ID.
- An MPLS label is constructed using this scheme

Number of digits	Function	Details
1	Traffic direction	1 = uplink 2 = downlink
1	Traffic type	1 = realtime 2 = delay aware 3 = delay unaware
4	POP ID	Right-aligned with zeros padding

- For example label “210133” indicates the uplink “delay aware” traffic from POP ID 133.

Functional principles

- The controller **maintains** (quasi-)real-time **knowledge** of:
  - The network topology,
  - Current usage of each backhaul link,
  - Current capacity of each backhaul link.

- For each **POP**:
  1. It determines **which ABR** to use,
  2. It calculates a “**main**” and a “**backup**” paths from the choosen ABR to the POP, ensuring they are as diversified as possible.
  3. It deploys the necessary OpenFlow forwarding rules on all the intermediates POP in order to implement the main and backup paths.
  4. On each node of the main path, an additional rule with lower priority is deployed to re-route traffic back to the last branching point.

- **OpenFlow matching** is done on igress port + MPLS label.

- **Fast-reroute** is implemented by having BFD invalidate the OpenFlow rules that egress to an “invalid” interface. Lower-priority rules will automatically match.