# Building automated wireless mesh networks

Attila Balogh, Magyar Telekom





# about me

- just a random IP guy
- working for Magyar Telekom
- engineering networks since ’97
- RouterOS user since 2004

how i met ~~your~~

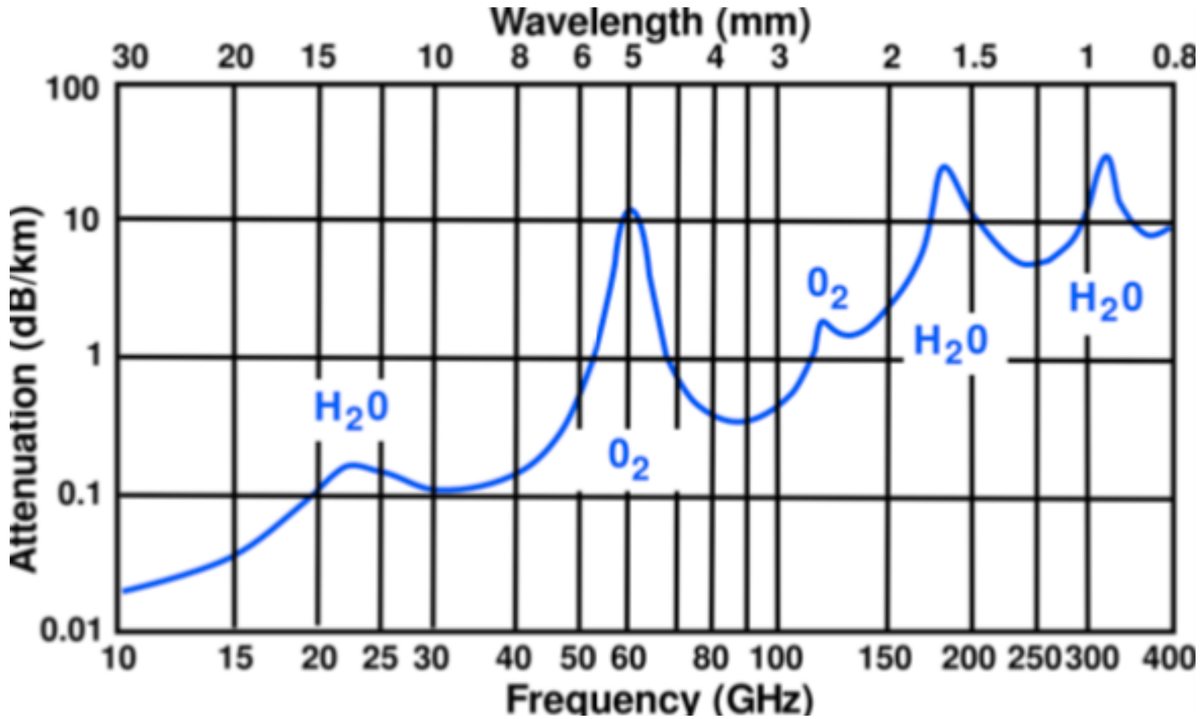
# 60 GHz



# PART I.

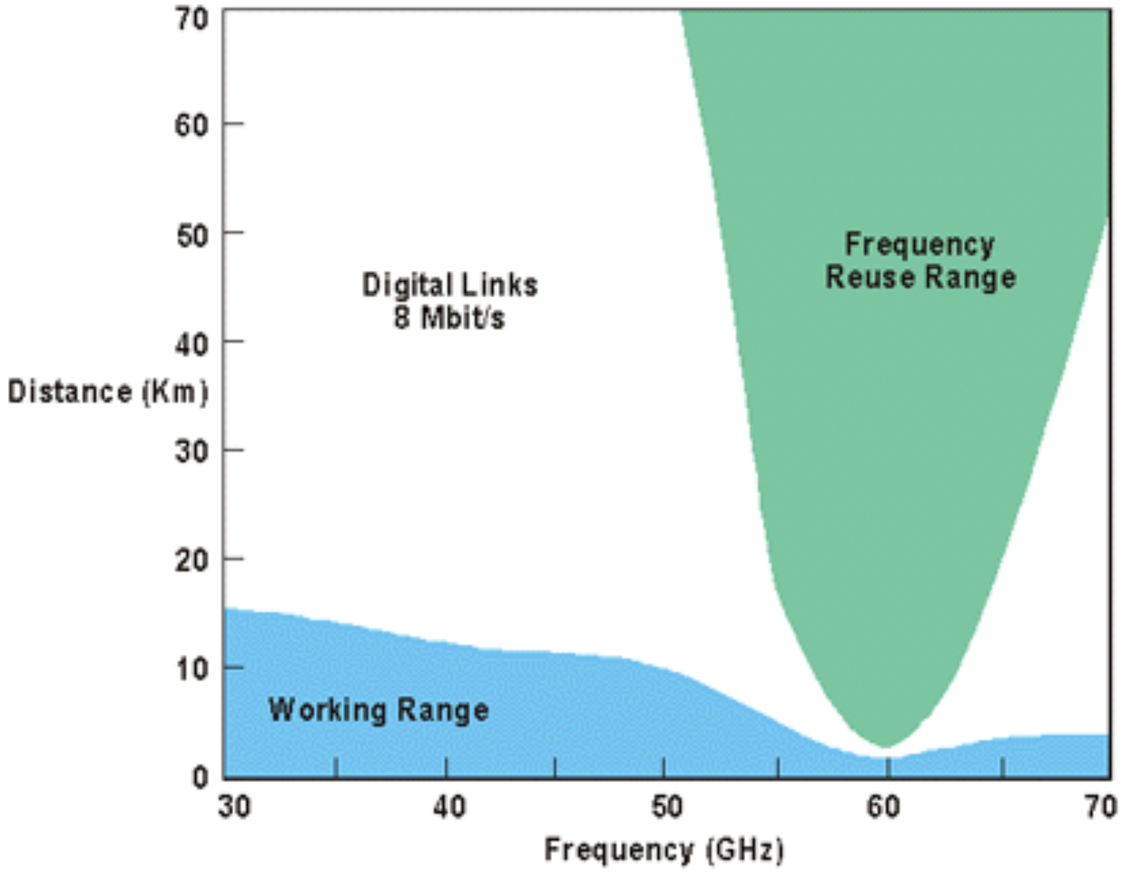
created by Craig Thomas and Carter Bays for CBS

# The spectrum (that no one wanted)



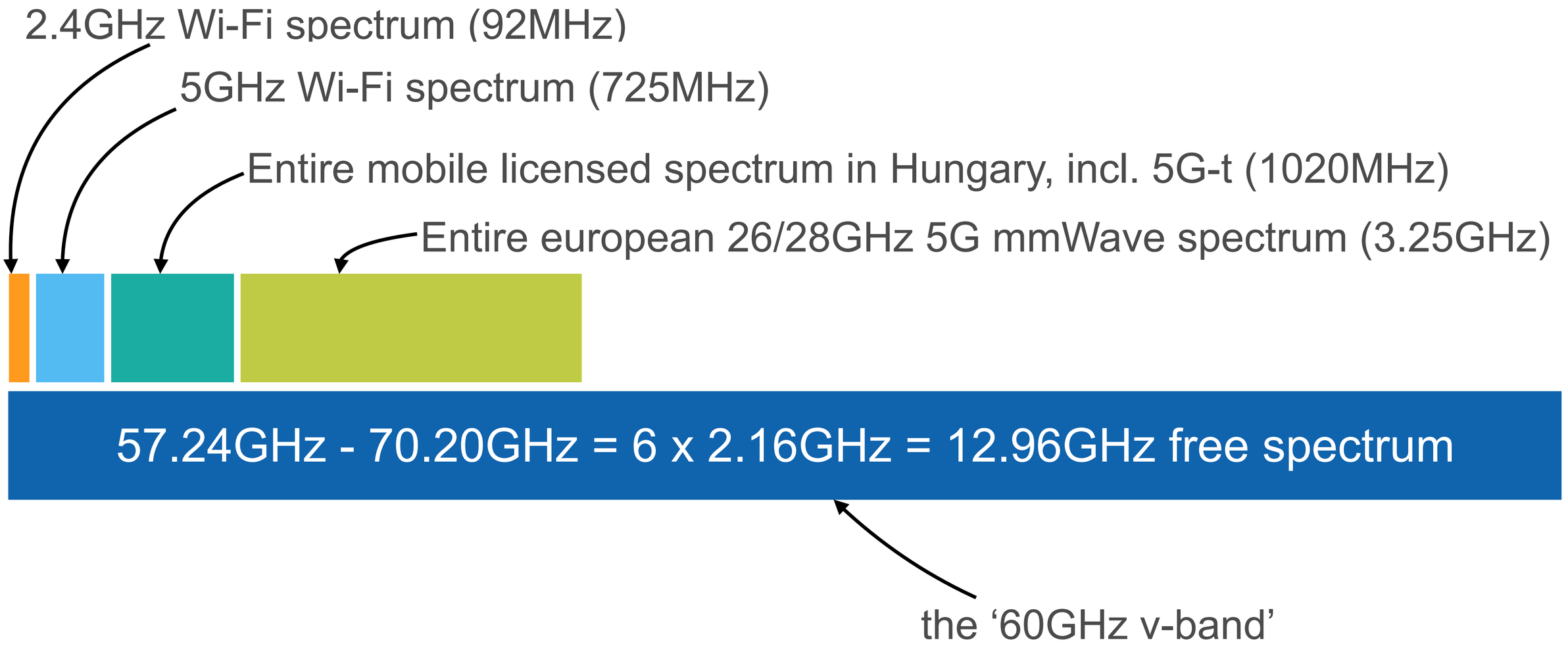

- line of sight
- rain fading
- oxygen absorption
- doppler frequency shift
- unlicensed

# The spectrum (that **WE** need)




- minimal interference (high FSL, $O_2$ absorption)
- beam-like, can be focused
- reflections - to overcome line-of-sight problems
- unlicensed, and big
- upper 2 channels similar to e-band

# DEFINE ‘BIG’

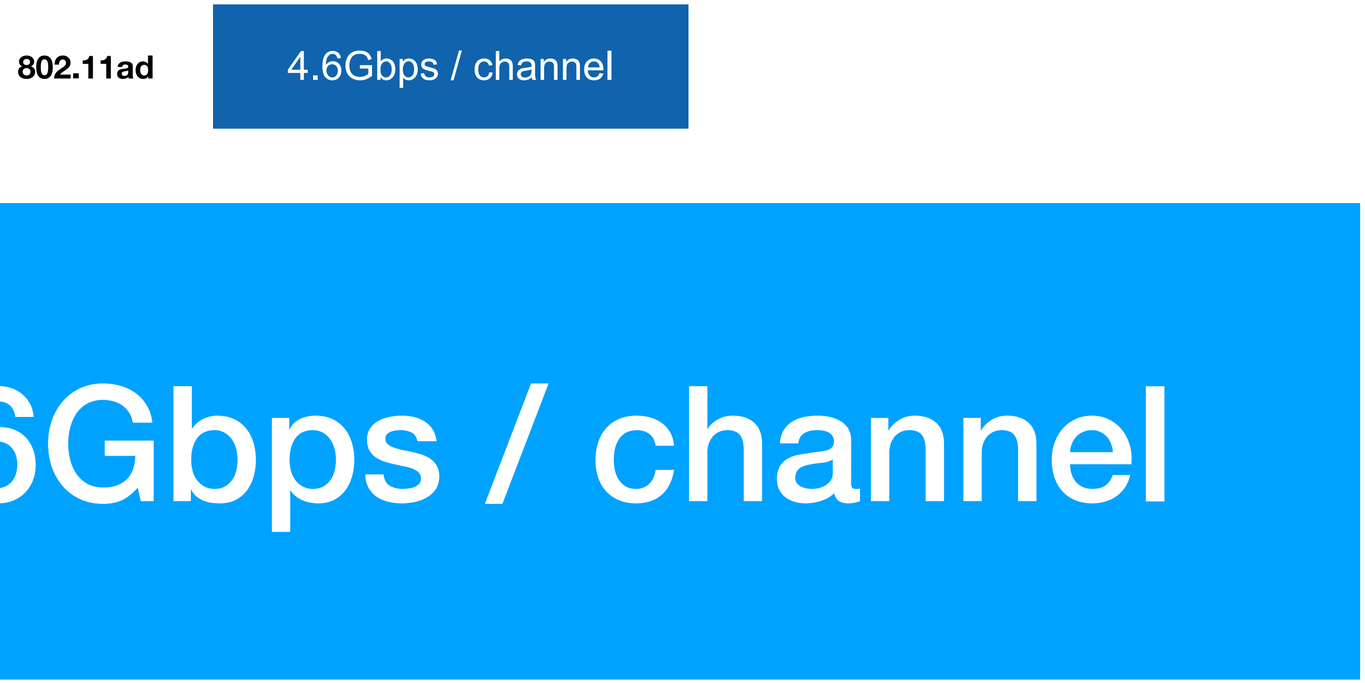

2.4GHz Wi-Fi spectrum (92MHz)
5GHz Wi-Fi spectrum (725MHz)
Entire mobile licensed spectrum in Hungary, incl. 5G-t (1020MHz)
Entire european 26/28GHz 5G mmWave spectrum (3.25GHz)
57.24GHz - 70.20GHz = 6 x 2.16GHz = 12.96GHz free spectrum
the ‘60GHz v-band’

# BUT WHAT IS IT GOOD FOR?

**802.11ad**

4.6Gbps / channel

# BUT WHAT IS IT GOOD FOR?



# Once upon a time…



TELECOM INFRA PROJECT









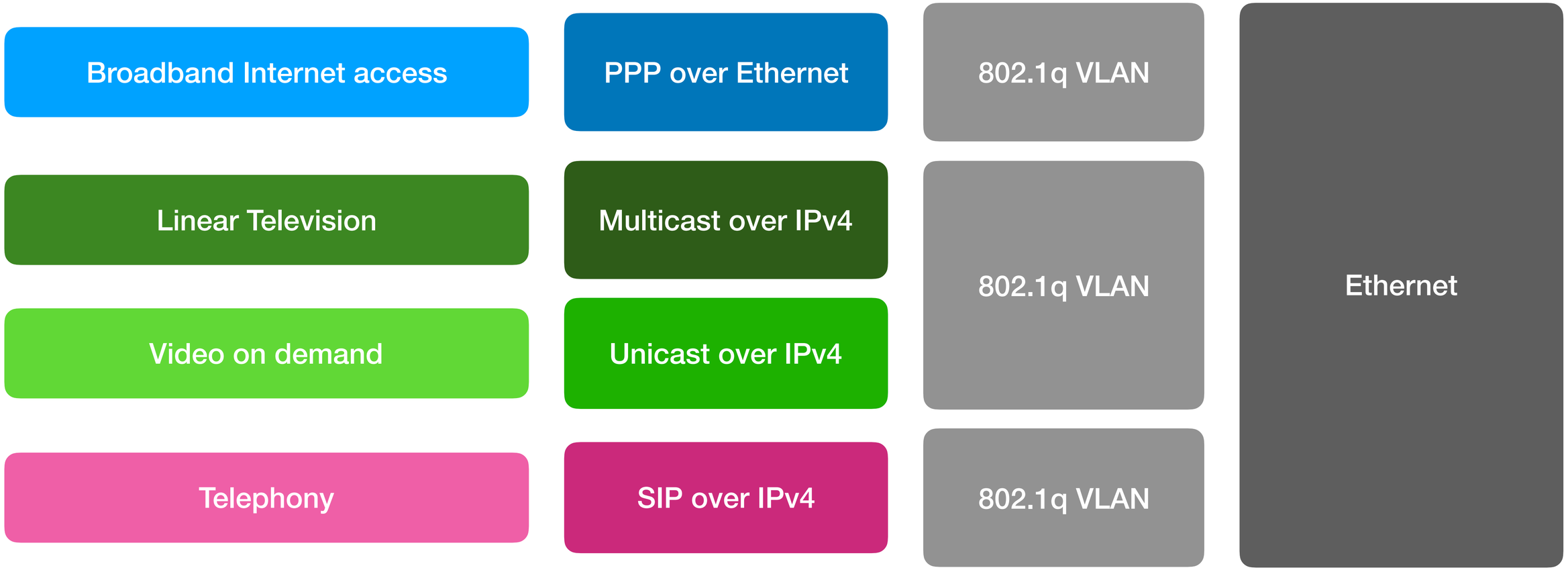

# … in a village far, far away

# Enter Terragraph

- 60 GHz single frequency mesh network
- bases on 802.11ad + GPS synchronised TDD
- Layer 3 routed mesh with IPv6
- Self organising, Zero touch provisioning
- scales up to 100s of nodes

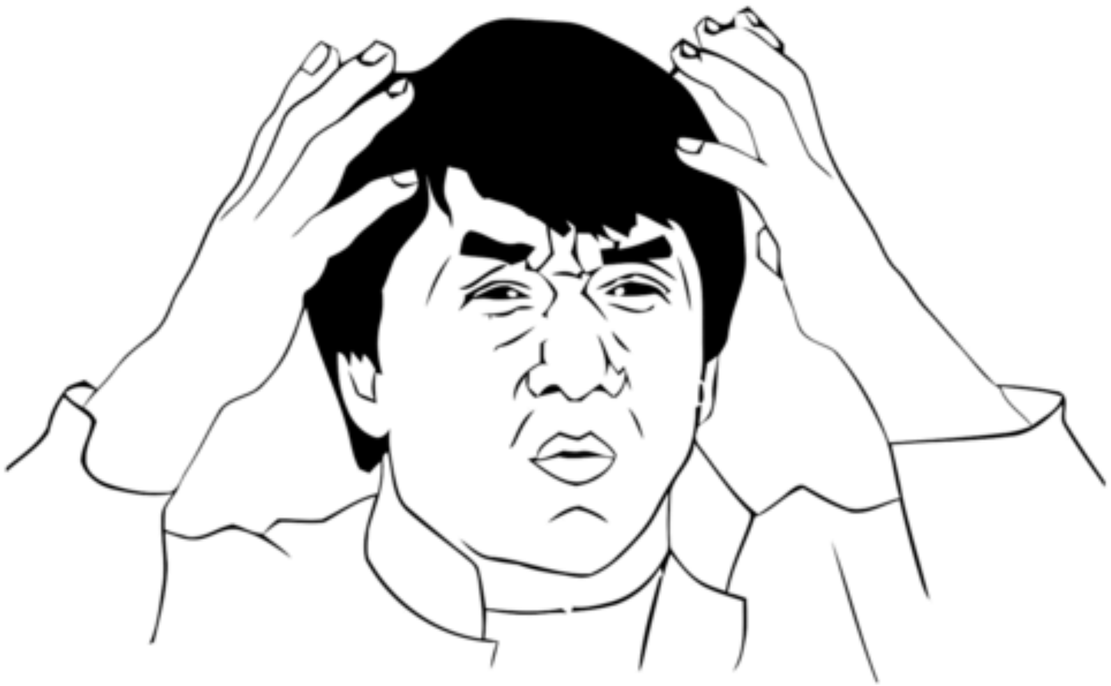


# meanwhile in telecom industry



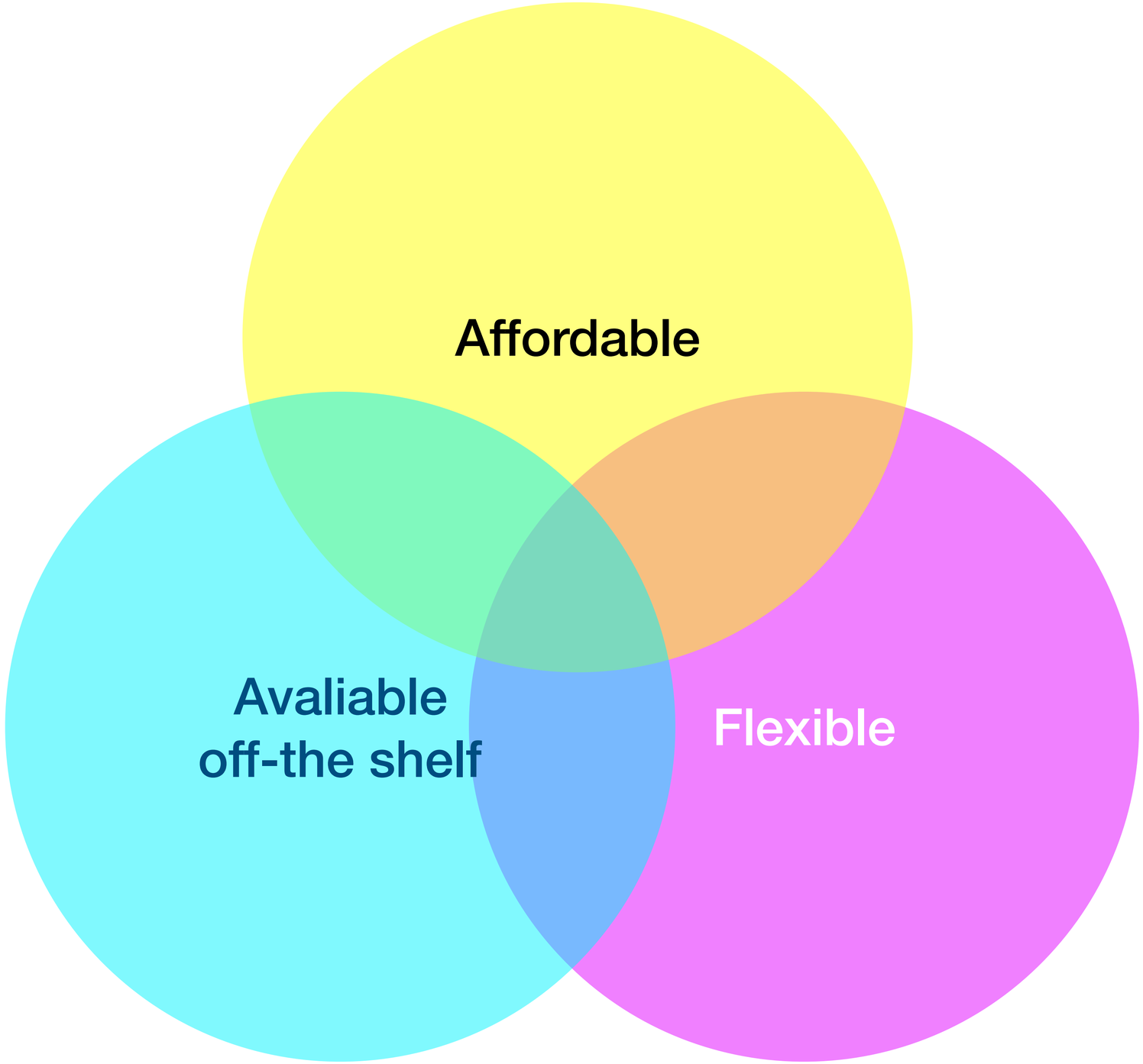
**What I need**

Ethernet



**What I have**

IPv6


Affordable
Avaliable
off-the shelf
Flexible


MikroTik
ROUTEROS

# idea #1 - L2TP

- Broadband Internet access
  routed L2TP - IPv4 + IPv6 over PPP over L2TP
- Multicast IPTV
  BCP over L2TP - Ethernet bridging over L2TP
- Session negotiation, authentication, keepalive is done by L2TP
- supports MLPPP - PPP fragmentation -> no MTU limits
- No support for ‘client-server’ tunnels over IPv6 in RouterOS

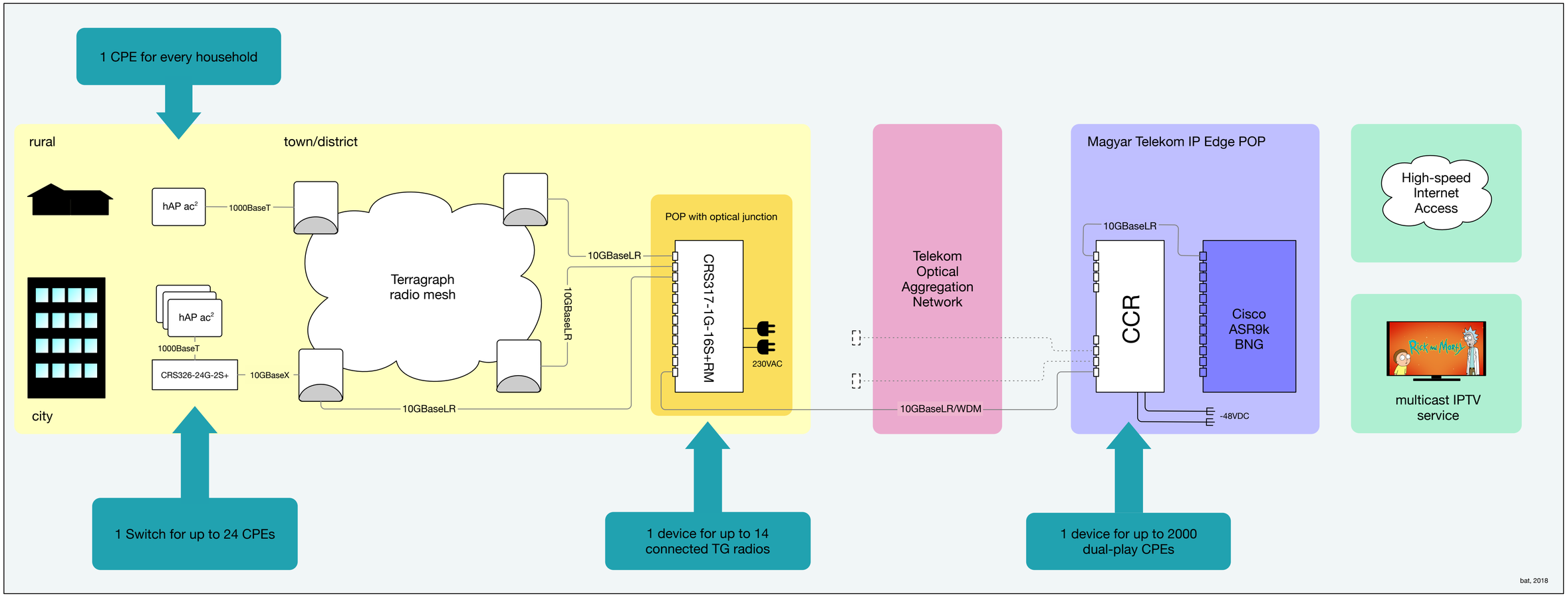
# idea #2 - EoIPv6

- Ethernet user plane -> VLANs are possible
- Works over IPv6
- Supports keep-alive and thus some link management
- Supports tunnel IDs: multiple tunnels are possible between the same nodes
- no link negotiation: client and server needs to know each other
- pre-configuration required on both sides

# bridging the gaps

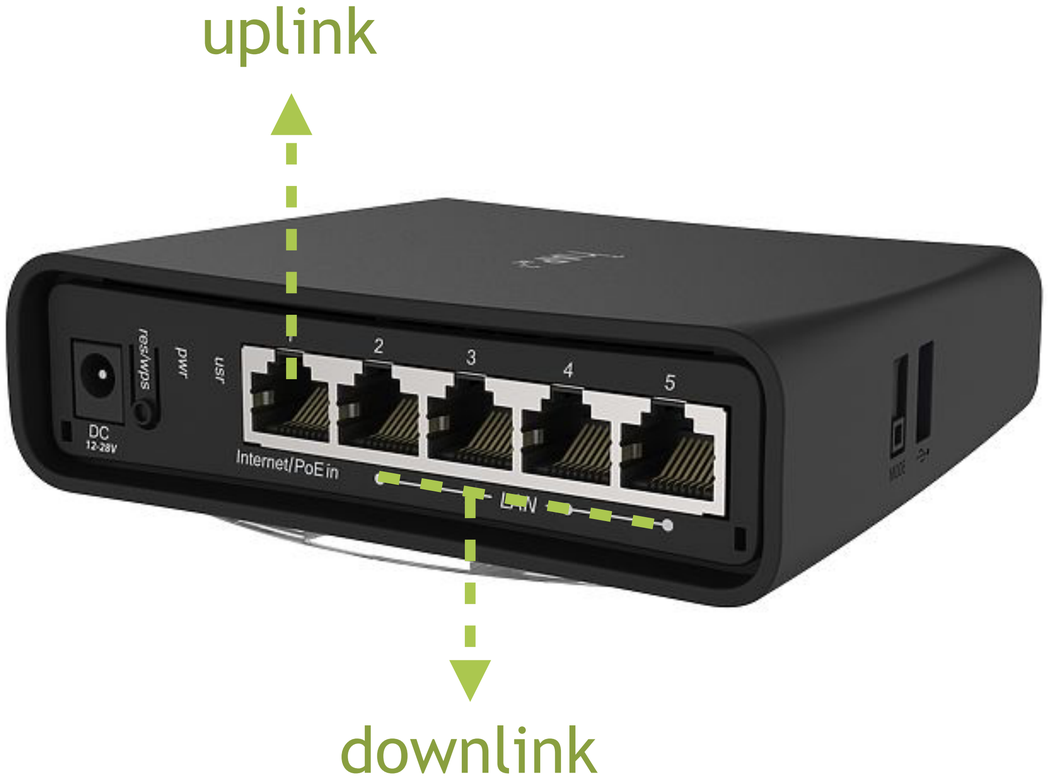
- use FQDNs to establish EoIPv6 tunnels
- use a central controller for tunnel IDs and configuration
- TR-069 is not capable enough for this task
- build a custom system that uses similar concepts
- use scripting on the routers to interact & fix bugs

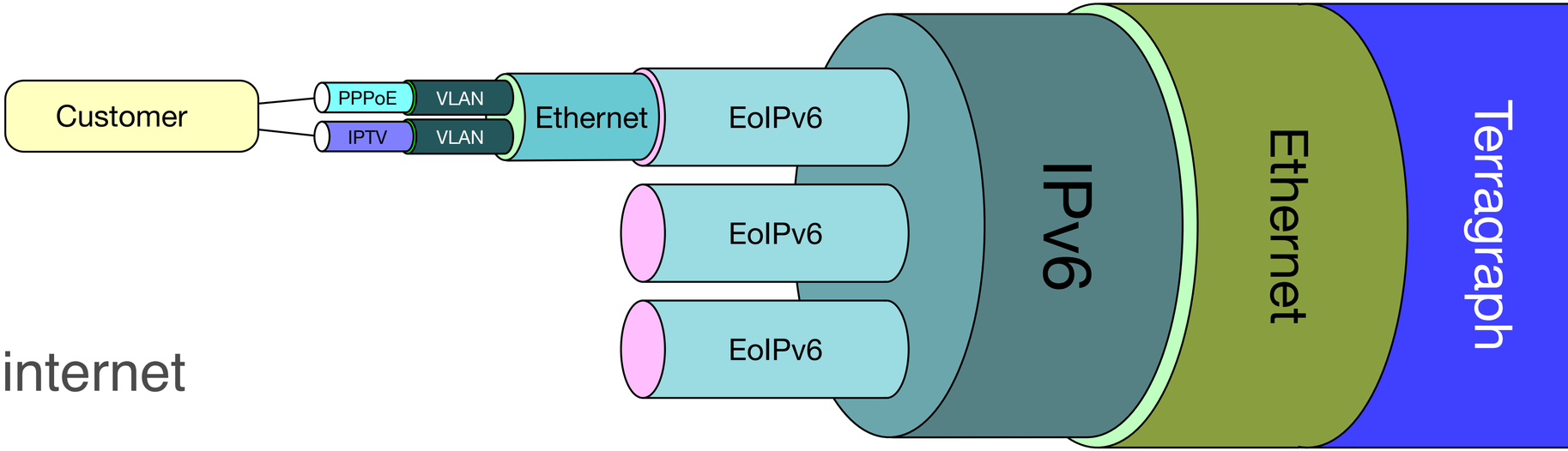
# this is how we do it



# the current indoor CPE in detail

- 4-core ARMv7 (Qualcomm IPQ4018)
- 5xGE ports
- dual-band 2x2 MIMO WiFi (b/g/n/ac)
- passive POE-in supported
- features/scripting/automation
- up to 700Mbps throughput with complex tunnelling






- 500Mbps broadband internet
- SD and HD live TV service with multicast
- Video on Demand service via unicast

| RTT: 12ms | Max DL: 956Mbps | Max UL: 965Mbps | Simultaneus 35M/35M | OSR 1:15 |
|---|---|---|---|---|

# the service architecture - protocol stacks

High-speed Internet Access




multicast IPTV service

IPv4 | IPv6

PPP over Ethernet

Ethernet frame

802.1q VLAN

IPv4

Ethernet frame

802.1q VLAN

Ethernet over IPv6 tunnel

Terragraph IPv6 only routed network

# how it is done, again?

- CPE serial numbers as identities
- DHCPv6 client to acquire address and to trigger registration
- [fixing EoIPv6 source in case of address change]
- register/update FQDN
- talk to the controller
- 1st time: stand still until new job arrives
- watchdogs, watchdogs and more custom watchdogs



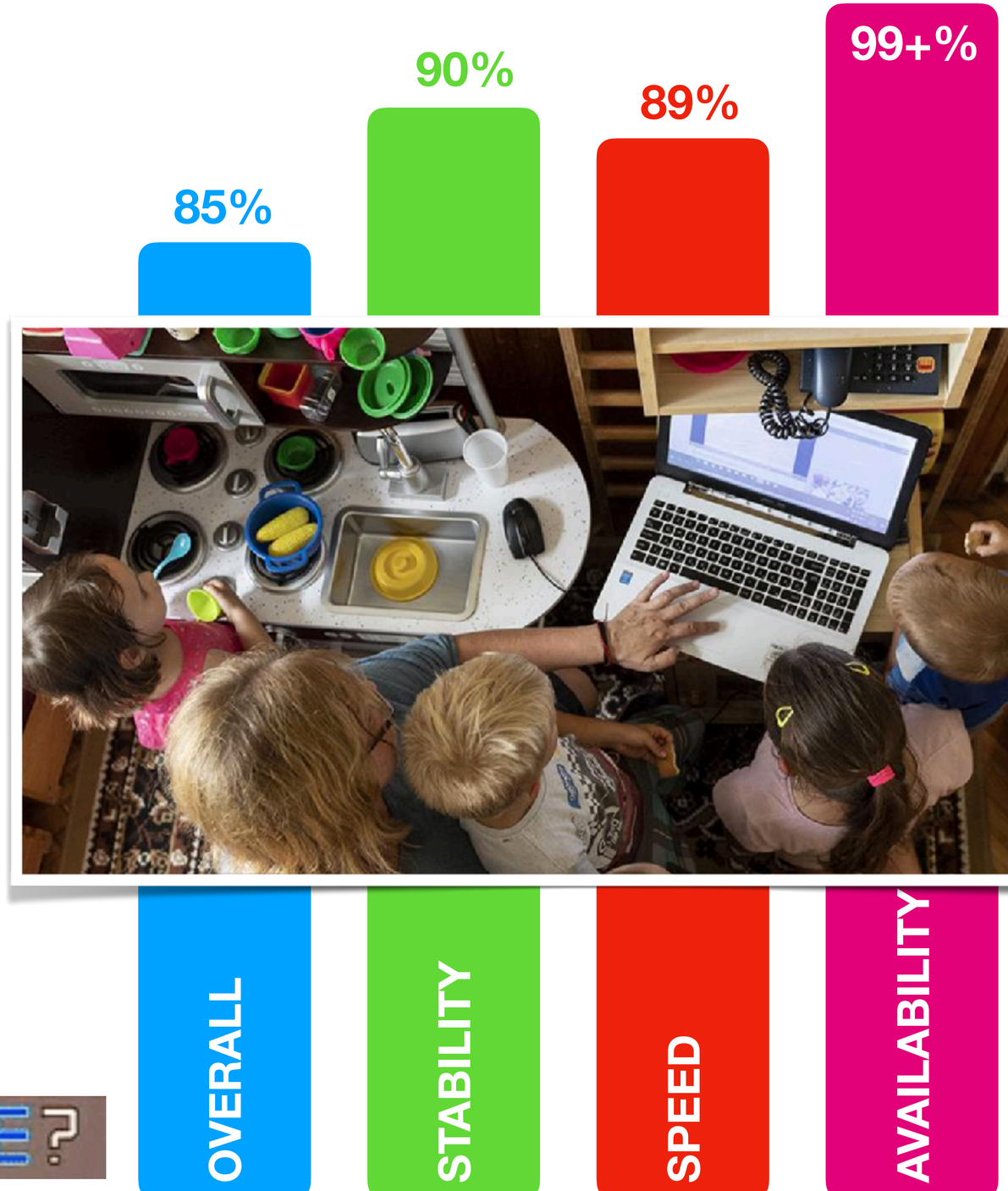
I don't know who you are. What I do have are a very particular set of skills, skills I have acquired over a very long career. I will find you, and I will manage you.

# device management (reinvented?)

- ‘Internet of routers’ as in ‘Internet of Things’
- don’t talk, but listen
- CPEs post status, performance data, health data, logs (telemetry)
- CPEs look for jobs to execute:
  - configuration changes
  - command execution
- brought to you by the almighty **`/tool fetch & import`**

# RESULTS & OUTLOOK

- Fully automated subscriber provisioning
- network in operation for 9+ months
- 3 month network availability 99.2%
- on-line customer surveys and focus group discussions
- customer satisfaction over 85%




WOULD YOU LIKE TO KNOW MORE?




# PART II.

# MIKROTIK ACCELERATES THE ADOPTION OF 60 GHZ TECHNOLOGIES WITH TERRAGRAPH

24th Feb, 2019 | Announcements



Press Release.
25 February 2019

Riga, Latvia - MikroTik is announcing a collaboration with Facebook to build high-speed connectivity solutions with Terragraph, helping to accelerate the adoption of 60 GHz fixed wireless access technologies to deliver gigabit services and connect more people, faster. The 60 GHz band

# Wireless Mesh network

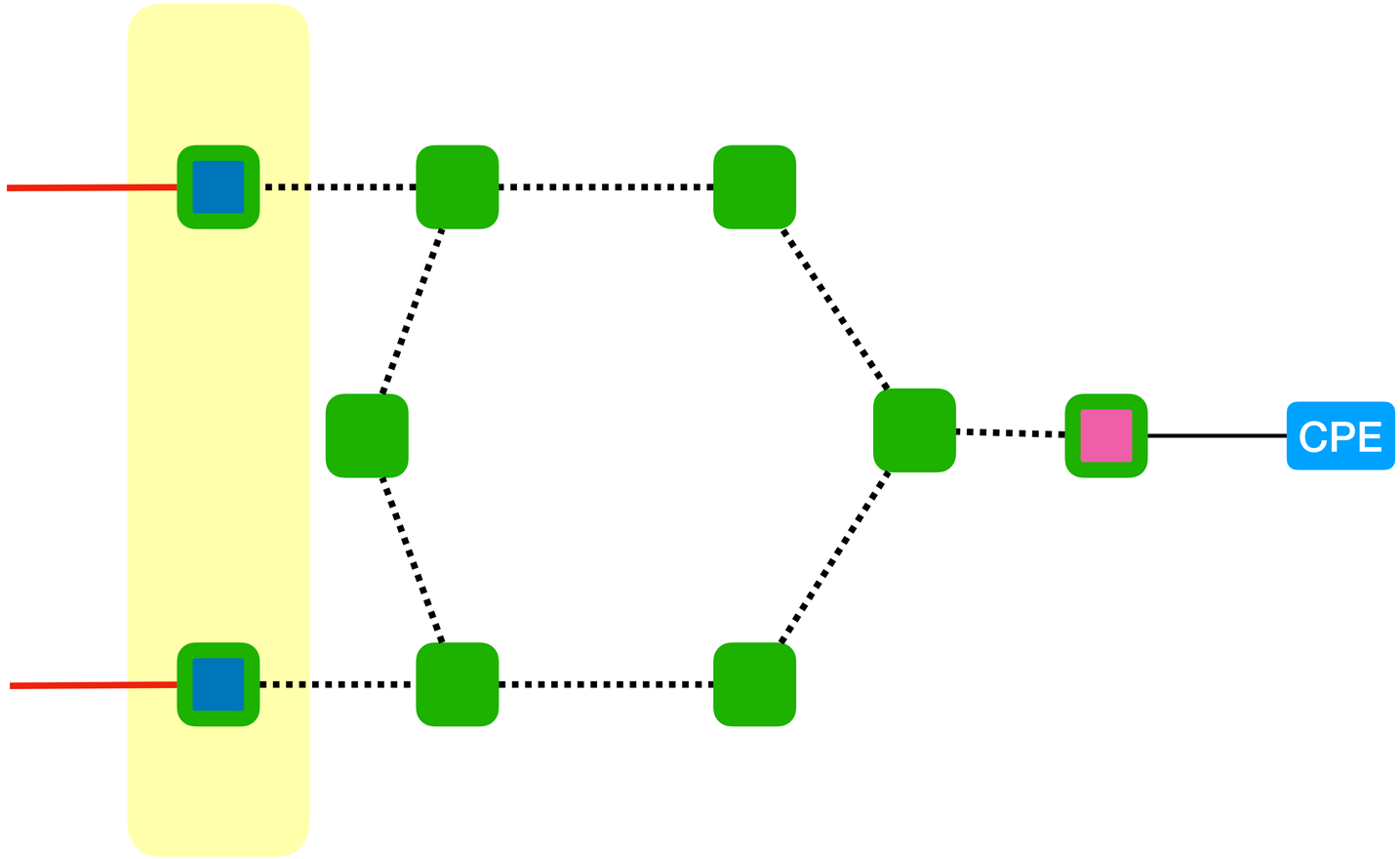
(~100 servings)

- ✓ WiGig support with at least MCS8
- ✓ L3 Routing in radio units
- ✓ Dynamic Routing protocol
- ✓ IPv6 support
- ✗ AP-AP wireless links
- ✗ synchronised network with TDD media access

# solving the issues

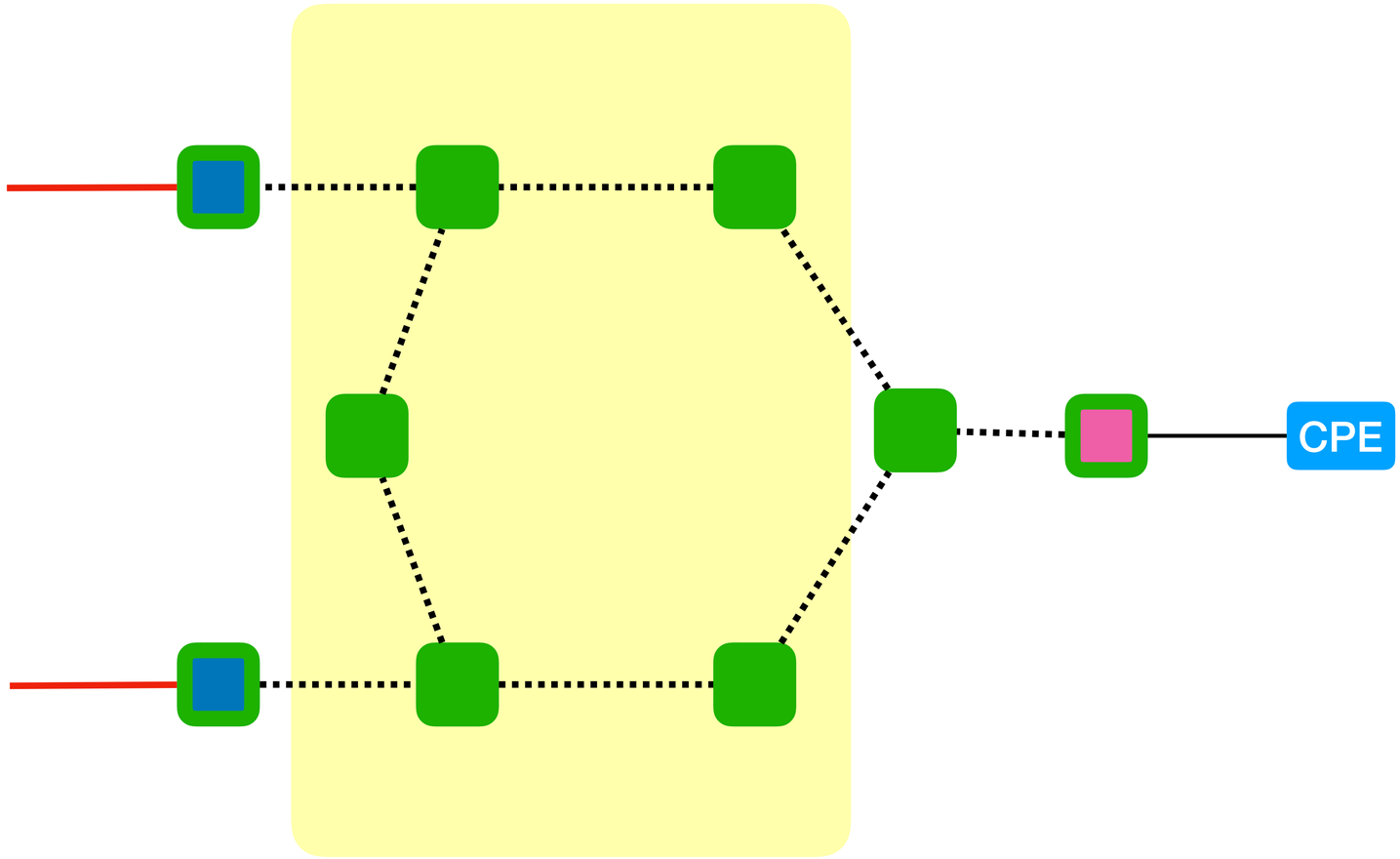
- Open/R >> OSPFv3 (some missing features delivered by HTTP/S)
- no AP-AP links -> dedicated radio for backhaul and for drop
- no TDD/TDMA media access -> relative low number of CPEs / radio
- link coordination -> use multiple SSIDs and ‘w60g station’ entries
- no network management system -> build one!

# the network
## terminology



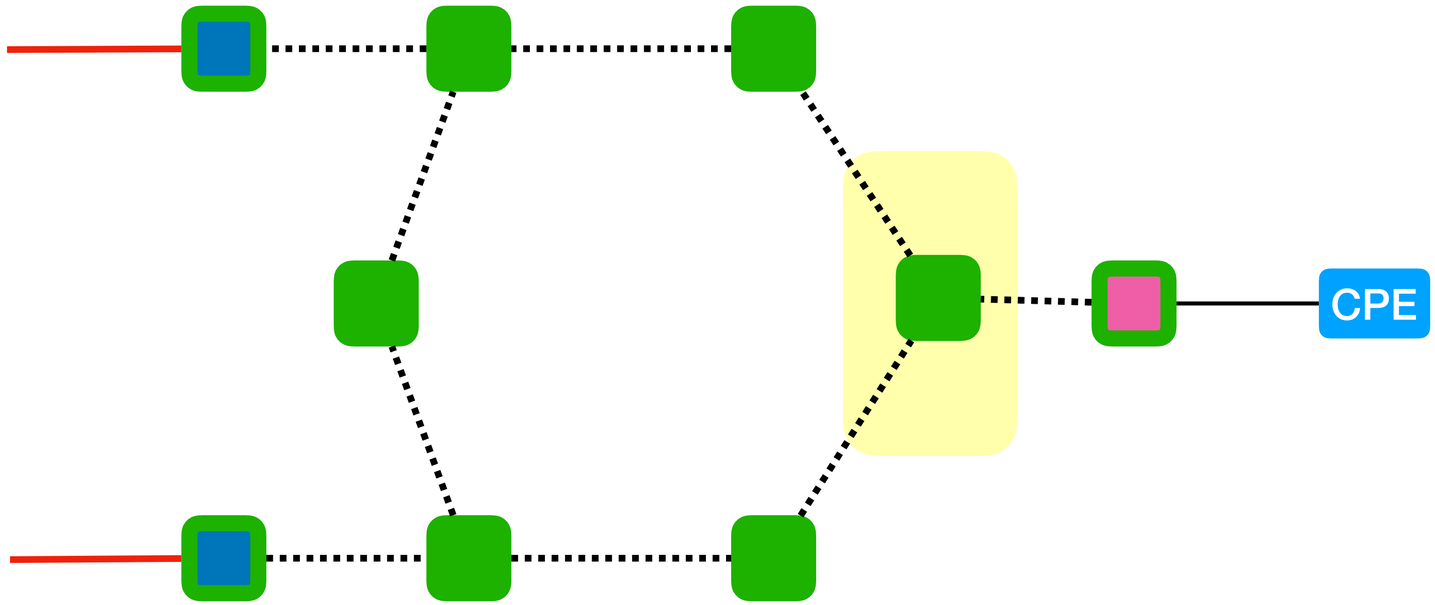

- **Fiber POP**
- connected via 1Gbps SM/ MM fiber
- speaks BGP to the outside
- originates default route
- ASBR in OSPFv3
- at least 2 in each mesh

# the network
## terminology



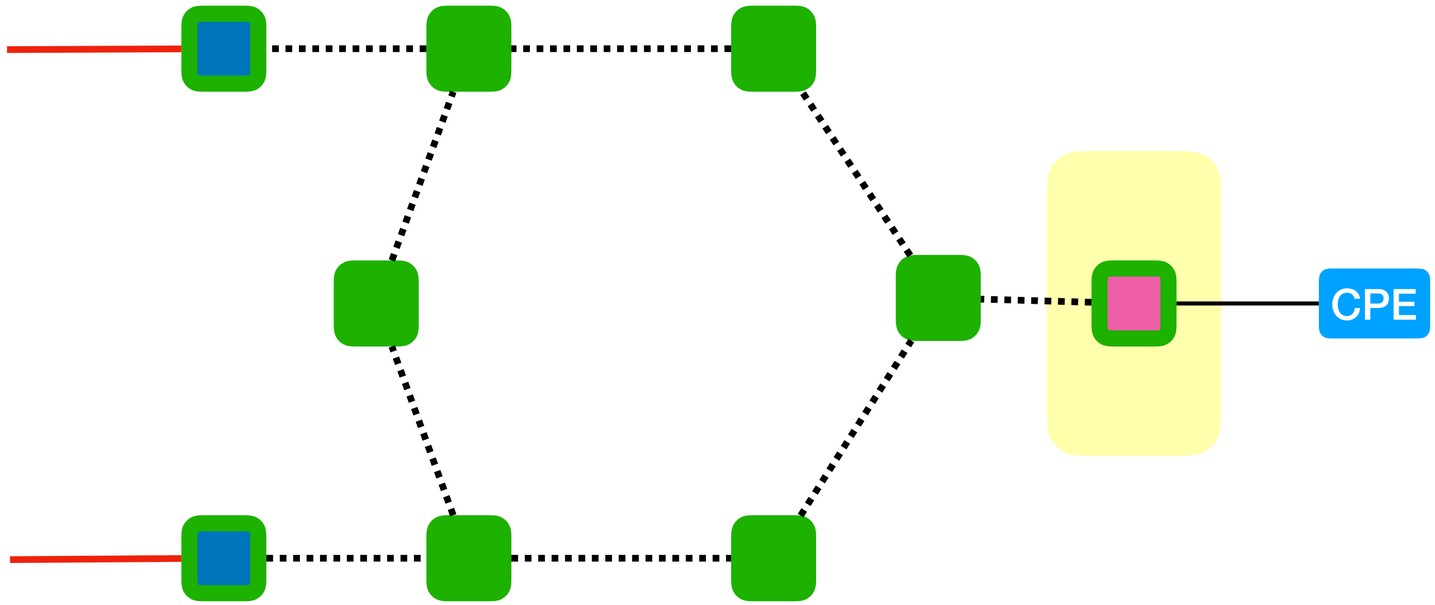

- **DN BH - backhaul distribution node**
- connected via MCS8+ to neighbouring DNs
- speaks OSPFv3
- wired connection to DN APs

# the network
## terminology



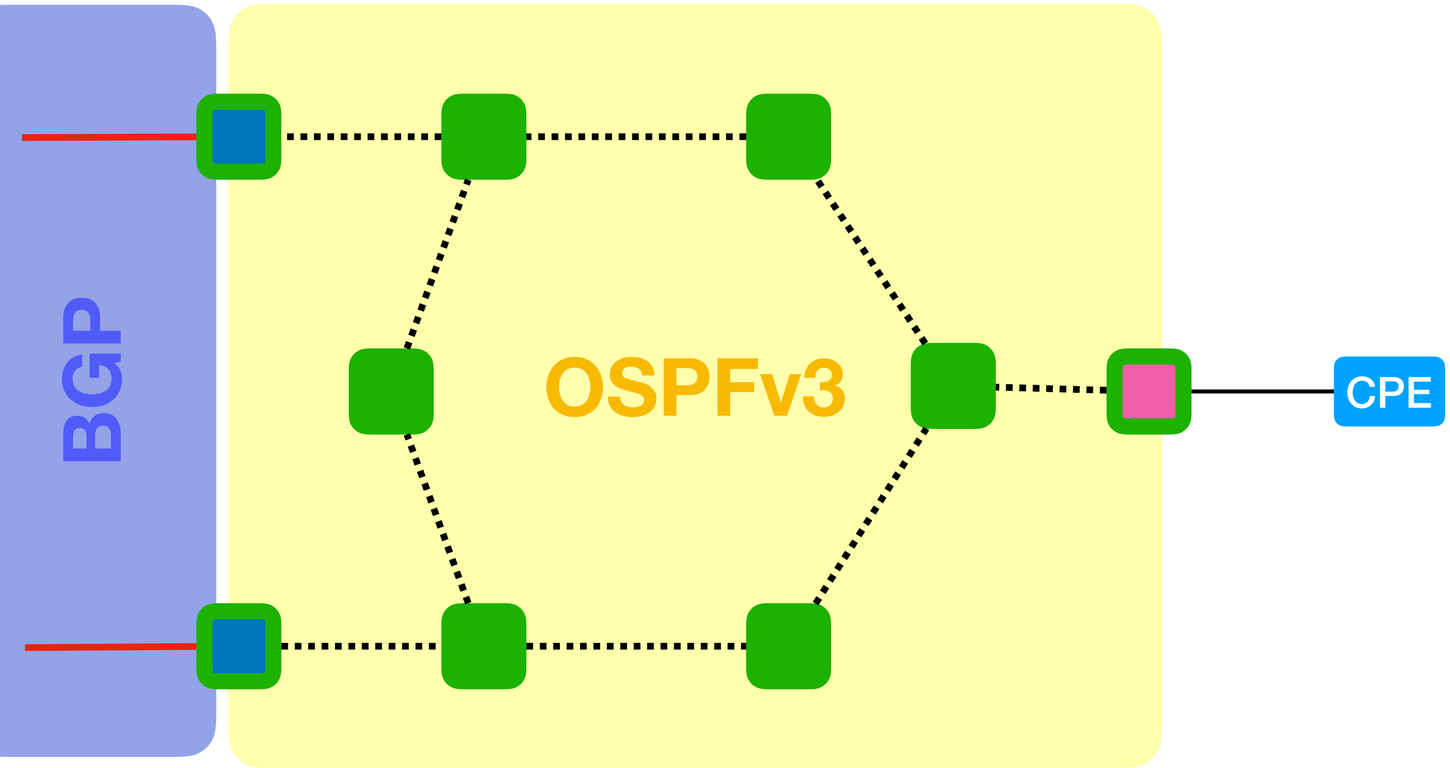

- **DN AP - drop distribution node**
- CNs connect to it wirelessly
- multiple links / sector
- speaks OSPFv3
- wired connection to DN BHs

# the network
## terminology




- **CN - client node**
- connects to DN AP wirelessly
- wired connection to CPE
- starts EoIPv6 tunnels
- speaks OSPFv3
- DHCPv6 PD towards the CPE

# the network
## routing



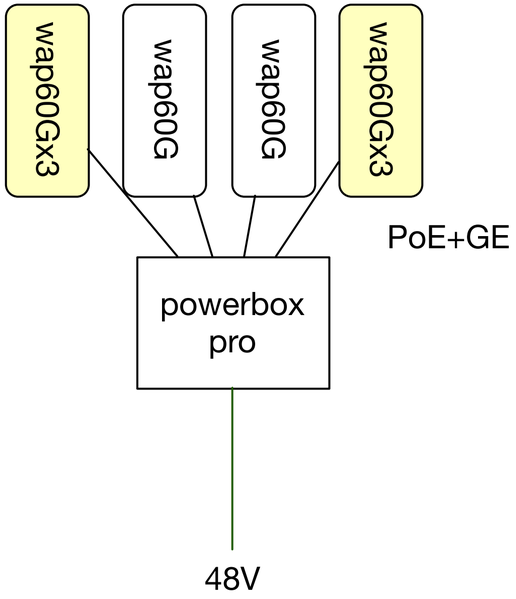

- **Dynamic routing across the entire network**
- node addresses
- subscriber pools

# the hardware

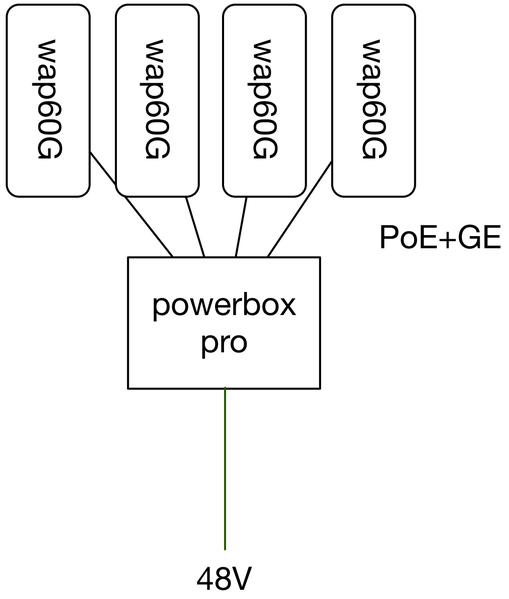
- wAP60G (backhaul & CPE)
- PowerBOX Pro (sector interconnect)
- wAP60Gx3 AP (access point)
- SXTsq60 Lite (low end CPE)



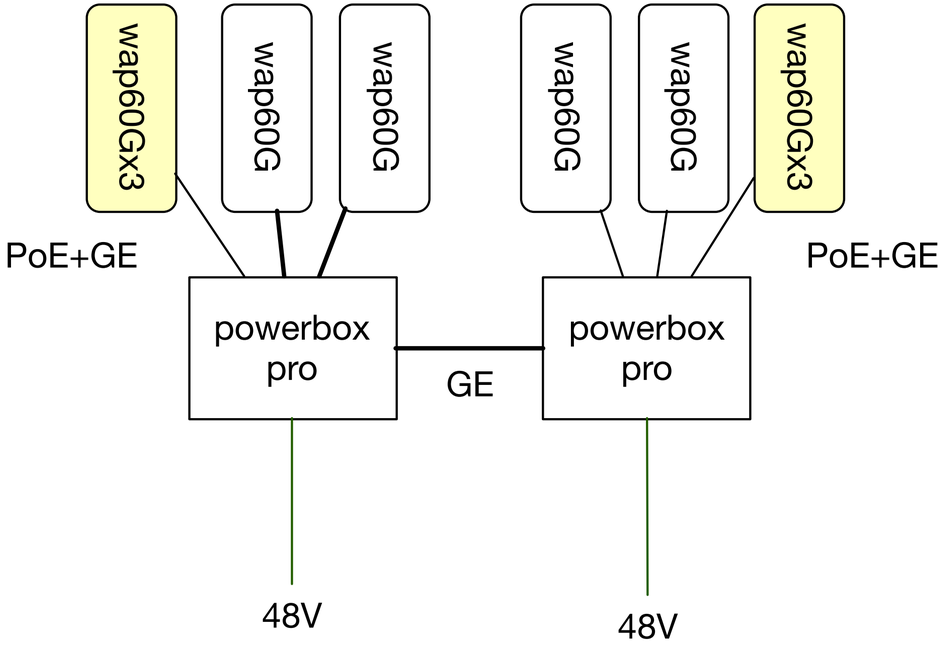
# the network
## building blocks




mesh node with drop




mesh node



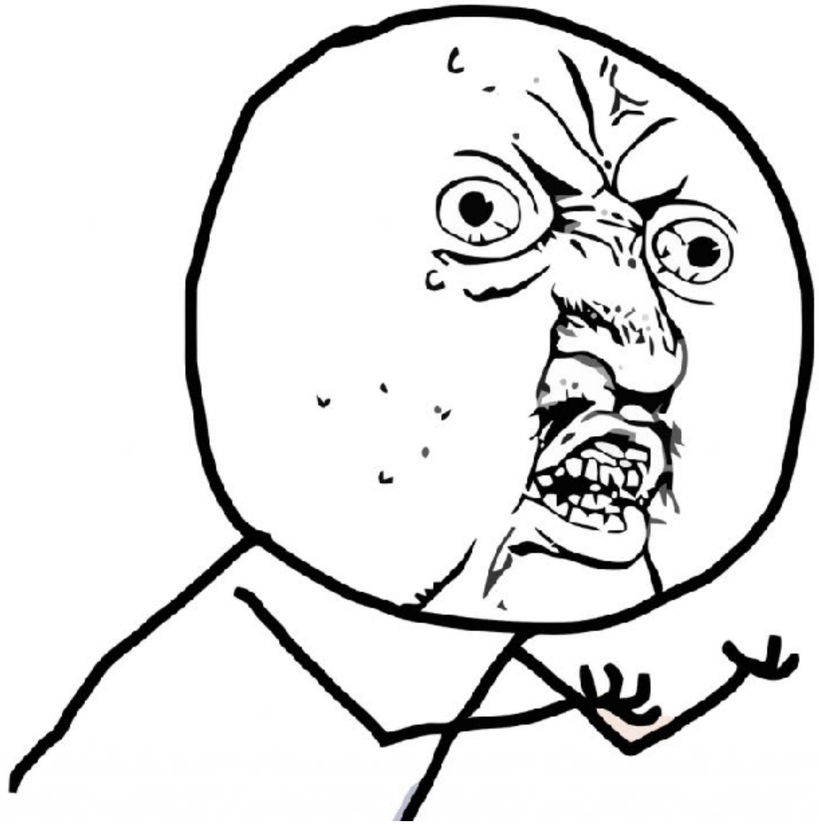

big mesh node with drop

# Why IPv6

- plenty of GUA available -> no need to NAT
- Link-local addresses -> link auto addressing with topology hiding
- 64 bits of address range -> flexible unique address generation
- readily supported by routerOS*

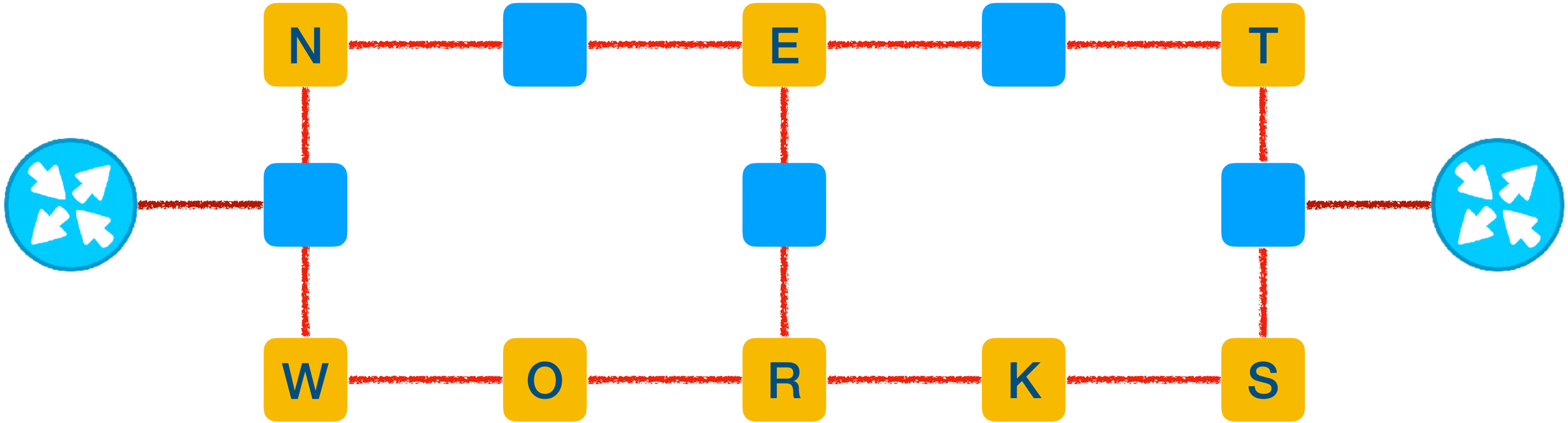
# Why OSPFv3

- complex topology, multiple routes, dynamic solution is required
- supports IPv6, works with link local addresses
- faster than RIPng
- point-to-point link support
- BFD support for faster change detection



# Why you don’t use just a flat layer-2?

# mesh




# have loops

*I think that I shall never see*
*A graph more lovely than a tree.*
*A tree whose crucial property*
*Is loop-free connectivity.*
*A tree that must be sure to span*
*So packets can reach every* LAN.

## LAYER 2 NETWORKS DON'T MIX PARTICULARLY WELL WITH LOOPS

**new frame arrived, src: 00:1c:4d:12:34:56 dst: 00:2e:1a:33:44:55, done. new frame arrived, src: 00:1c:4d: 12:34:56 dst: 00:2e:1a:33:44:55, done. new frame arrived, src: 00:1c:4d:12:34:56 dst: 00:2e:1a:33:44:55, done. new frame arrived, src: 00:1c:4d:12:34:56 dst: 00:2e:1a:33:44:55, done. new frame arrived, src: 00:1c:4d:12:34:56 dst: 00:2e:1a:33:44:55, done. new frame arrived, src: 00:1c:4d:12:34:56 dst: 00:2e:1a: 33:44:55, done. new frame arrived, src: 00:1c:4d:12:34:56 dst: 00:2e:1a:33:44:55, done. new frame arrived, src: 00:1c:4d:12:34:56 dst: 00:2e:1a:33:44:55, done. new frame arrived, src: 00:1c:4d:12:34:56 dst: 00:2e:1a:33:44:55, done. new frame arrived, src: 00:1c:4d:12:34:56 dst: 00:2e:1a:33:44:55, done. new frame arrived, src: 00:1c:4d:12:34:56 dst: 00:2e:1a:33:44:55, done. new frame arrived, src: 00:1c:4d:12:34:56 dst: 00:2e:1a:33:44:55, done. new**

**Bridge port received packet with own address as source, probably loop**

# auto addressing

- links with link local addresses just work fine
- to be able to talk to the ‘outside’, GUA is needed
- let’s acquire one from the controller!
- controller is on the outside, so cannot be reached without a GUA



# auto addressing, with some help

- what if there could be a hint…
- OSPFv3 can carry routing information of GUA prefixes in a link-local only network
- a special suffix could indicate that this is a ‘hint’
- take the common prefix hint, add something unique -> unique GUA
- routerboard serial numbers are unique by definition

# addressing example

:local hint [/ipv6 route get [find  where dst-address~"bad:babe/128"] dst-address ]

2001:db8:001:603::bad:babe/128

2001:db8:001:603::bad:babe/128 && ffff:ffff:ffff:ffff::

2001:db8:001:603::/128

:local mysn [/sys routerboard get serial-number ]

81ED08262E2B

:pick, :pick, :pick, 0000 -> ("0000:" . $part1 . ":" . $part2 . ":" . $part3)




2001:db8:001:603::81ED:0826:2E2B/128

/ipv6 address { remove [find where global and interface="lo0"]; add interface=lo0 address=$loaddr }

# addressing rules

- network infrastructure prefix: /64
- node IPv6 address: /128 (loopback)
- network UNI pool: /48
- node UNI address: /64

# external connectivity

- BGP advertises mesh aggregate network to the outside (/64-/40)
- OSPFv3 propagetes the BGP provided default into the mesh
- node addresses are not exposed as /128
- all nodes are only reachable via IPv6
- DHCPv6 PD assigned prefixes are redistributed to BGP though OSPF

# client connectivity

ethernet emulation

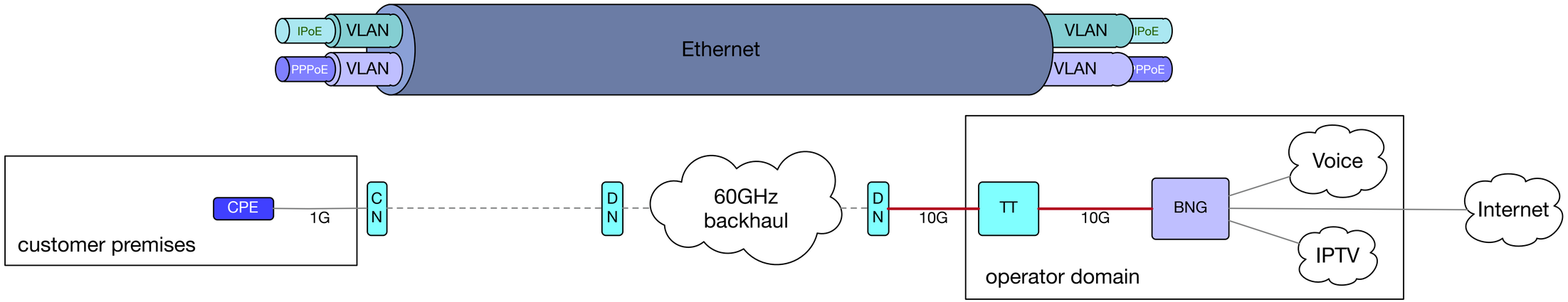
a.k.a legacy

native routed IPv6

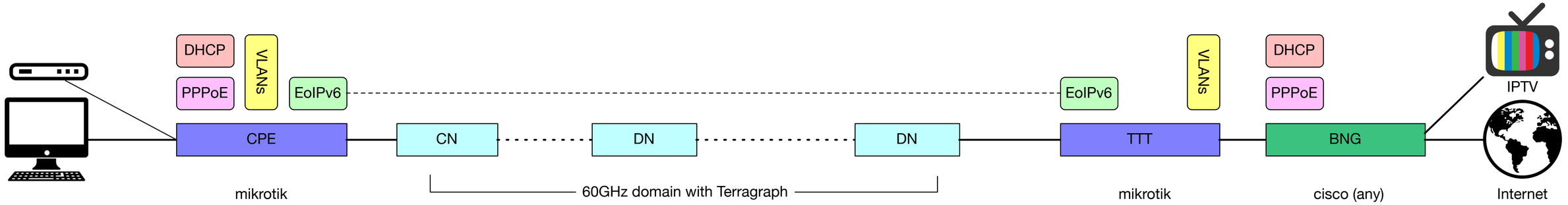
lean and simple

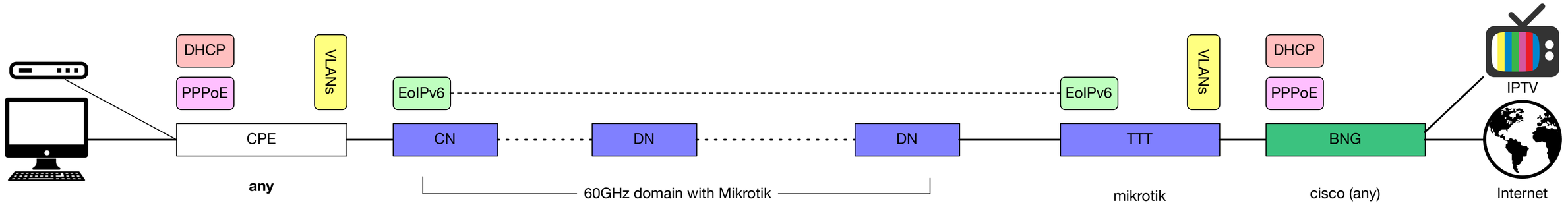
# legacy service compatibility

- EoIPv6 tunnels are started in CNs (wAP60G)
- tunnels terminated in CCR (ttt)
- tunnel creation done based on FQDNs and central orchestrator
- 2 VLANs in each tunnel (HSI 1000, IPTV 1001)



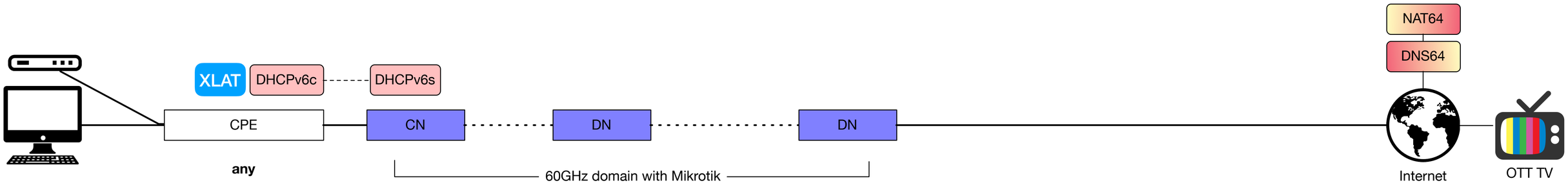
# legacy service compatibility




- no crucial difference to existing setup (tunnel initiation changes)
- less functionality in the CPE

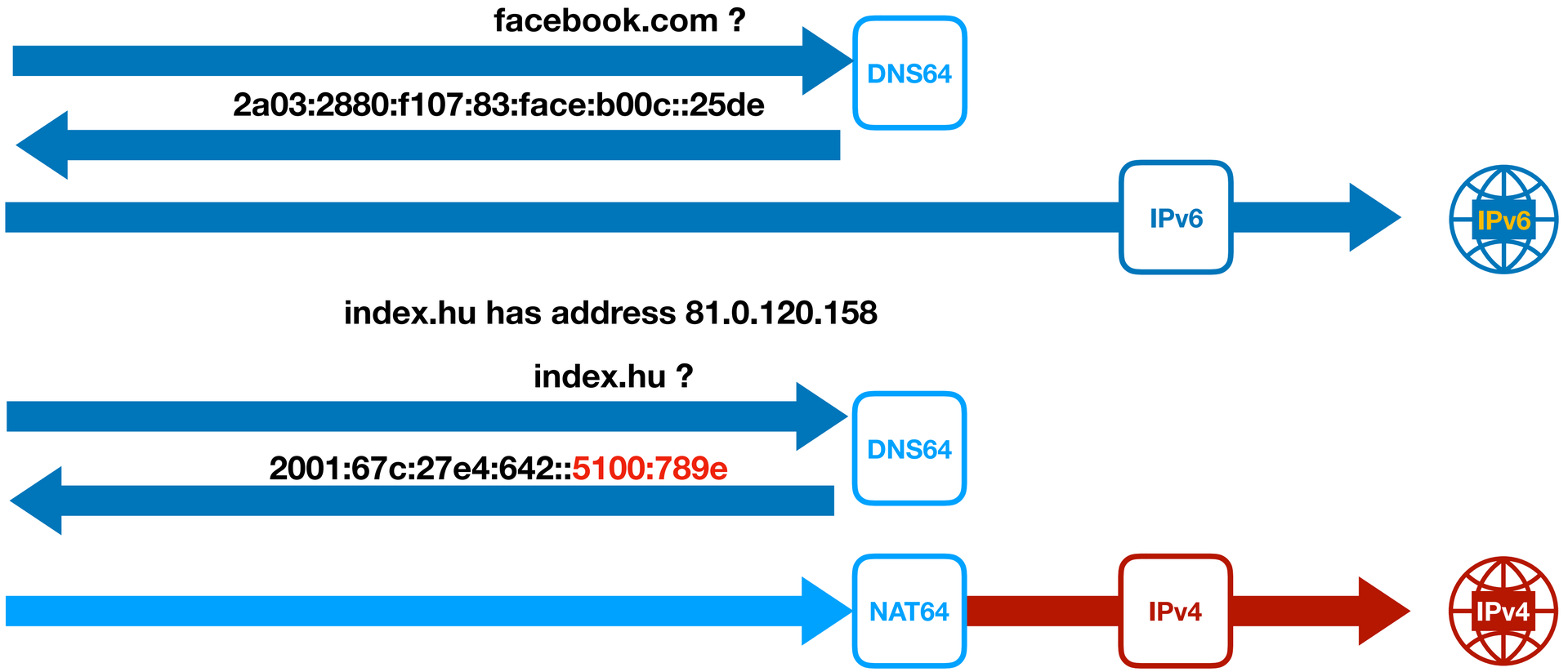
# modern service delivery




- service creation at the CN, session logging via controller
- end-to-end routed service with IPv6 only access
- NAT64/DNS64 based access to legacy IPv4 resources
- XLAT464* for IPv4 only devices behind the CPE

# DNS64/NAT64 crash course






5100:789e 0x51 0x00 0x78 0x9e 81.0.120.158

try it here: https://go6lab.si/current-ipv6-tests/nat64dns64-public-test/

# automation details

- client initiated communication
- DNs and CNs push and pull data from controller using HTTPS
- node identification based on serial numbers
- non-managed DN nodes show up as TN (trans-node)
- controller to provide final settings (SSID, password, mode, frequency)
- CPE management can be done based on TR-069

# toolchain

- nginx (reverse proxy, static content)
- influxDB (time series DB)
- mySQL (config data - so far)
- grafana (data visualisation)
- perl/dancer (web application)
- `/tool fetch` & `/system script` & `/system scheduler` (routeros integration)

# room for improvement

- more responsive interaction on automation
  - less complex protocol, like MQTT (pubsub) would be nice
- MCS9+ for backhaul
- support for Terragraph MAC & PHY layer
- Terragraph compatible AP-AP link support
- multi baseband units for more throughput
- N-BaseT for more capacity



**thanks for the attention**