

Ondřej Caletka | RIPE NCC | 22 October 2024

## RIPE Meeting Network

How we run a conference network for a networking conference



#### What is a **RIPE** Meeting

#### A week-long event twice a year

- 600+ attendees from all over the world
- Next up: **RIPE 89 Prague**, 28 October 1 November 2024

#### With a custom temporary Wi-Fi network

- AS2121
- 193.0.24.0/21
- 2001:67c:64::/48

Ondřej Caletka | RIPE NCC | 22 October 2024





### **Geolocation Issues**

#### **BSSID-based geolocation**

- Based on assumption that APs do not move
- Issues with Google **disappeared around 2016**

#### **IP-based geolocation**

- Privately curated lists by many commercial parties
- We publish a CSV list for Google
- This is now standardised as RFC 8805
- You have to tell providers; RFC 9092 discovery is not very popular
- We have a list of 7 geolocation providers to check prior every meeting; two still need manual updates



**Amsterdam - Warsaw - Athens** in one hour, 2014, coloured







### **Physical Network**

- Two VM hosts running VMware vSphere
  - SuperMicro SuperServer E300-9D-8CN8TP
  - 25 VMs including routers, firewalls, DHCP servers, DNS resolvers, Wi-Fi controller
- Switches
  - Juniper EX2200 (48×GE PoE+ + 4×10GE SFP+)
  - **Zyxel GS-1900-10HP** (8×GE PoE+, 2×SFP, VLAN)
  - MikroTik CRS305-1G-4S+IN (4x10GE)
- Access Points
  - Unifi UAP AC (S)HD



CN8TP OHCP oller

OGE SFP+) ×SFP, VLAN) E)





#### **Testing After Covid Break**



#### Ondřej Caletka | RIPE NCC | 22 October 2024





#### The Meeting Network Runs on Open Source

- Edge routers running BIRD
- Firewall using **nftables**
- DNS resolver cluster of Knot Resolver/BIND
- DNS load balancer running keepalived
- DHCP servers running Kea
- NAT64 using Jool
- Statistics collected using collectd + InfluxDB + Grafana
- Deployed using Ansible







### Logical Network Topology



Ondřej Caletka | RIPE NCC | 22 October 2024





### **Using IPv6-Mostly for the Meeting Network**

- Dual-stack network with NAT64, DNS64 and PREF64 RA option
- Devices decide themselves whether they need native IPv4 or not
- **IPv6-only is preferred** by 70+ percent devices
- Requires **perfectly working IPv6**, as well as NAT64/DNS64





## NAT64





#### **Issues with NAT64**

- Browser console reports **403 Forbidden**



#### • We use the Well-Known NAT64 prefix and a **pool of 256 IPv4 addresses**

# Everything works, except some VOD platforms (NOS.nl, ivysilani.cz, ...)







#### **Some VOD Platforms Care About Source IPv4 address**



Client

1

**GET token** 

token = F(**client IP**,...)

#### GET video?t=token

Ondřej Caletka | RIPE NCC | 22 October 2024



### <





### **Jool IPv4 Allocation Strategy**

- IPv6 packet
  - by default: source IPv6, destination IPv6, destination port -
  - hash collisions are resolved by a (slow) iterative process -
- Global option f-args influence what is hashed

  - but all sessions made by one host are causing collisions
- There's a <u>branch of Jool</u> with Ondřej Caletka's hashing algorithm
  - uses two hashes, one for choosing IPv4 address, other to choose port
  - no measurement data to prove it is indeed better, not merged



#### • Address and port tuple is determined by **hashing some parts** of the

#### setting it to 8 (source IPv6 only) resolves the issue with the VOD platforms





## USING LINUX AS A ROUTER





### **IPv6 Neighbor Advertisement (RFC 4861)**



### **R** - Router flag. When set, the R-bit indicates that the sender is a router. The R-bit is used by Neighbor Unreachability Detection to detect a router that changes to a host.







### **Symptoms**

- Everything works after network attachment
- Linux hosts work forever
- macOS loses default gateway after 6 seconds

Ondřej Caletka | RIPE NCC | 22 October 2024





### What makes a Linux router respond with R=0?

- Turned out this to be the (only) feature of perinterface forwarding sysctl switch
- IPv6 forwarding is just a **global switch** on Linux
  - yet there are still per interface switches
- NetworkManager used to reset per-interface switch during interface setup
  net.ipv6.conf.all.forwarding = 1
  - fixed in version 1.44.0

4

- net.ipv6.conf.default.forwarding = 1
- net.ipv6.conf.lo.forwarding = 1
- net.ipv6.conf.eth0.forwarding = 0
- net.ipv6.conf.eth1.forwarding = 0





#### NetworkManager vs Full BGP Feed

- Burns CPU just by listening to all netlink events
- Worked around by **stopping it after** boot
- Fixed earlier this year







### **Dealing with ARP Noise**

- Caused by omnipresent Internet-wide scans
- Kernel-space ARP implementation has no negative cache
- **arpd** to rescue!
  - part of iproute2
  - implements ARP in userspace
  - has **negative cache**
- 30 times less ARP messages on an empty network
  - before: **250 pps**, 84 kbps
  - after: **8 pps**, 2.7 kbps



#### # arpd -k -a2 eth0 eth1 eth2 eth3





## 





### **Wi-Fi Network**

- UniFi controller running on Debian Linux
- Manual channel configuration, mostly 5 GHz only
- Legacy eSSID names changing every meeting
- Multicast and Broadcast control kills IPv6 NDP
  - you have to allow-list MAC addresses of all wired IPv6 hosts
  - unintended **RA-guard-like function**





#### RIPE 88 Wi-Fi stats



# Questions & Comments



Ondrej.Caletka@ripe.net https://Ondřej.Caletka.nl





