# Netconf’23 - mDNS wake-up and offload

**Florian Fainelli**

Sep 25th 2023

# What is this about?

- Most Set-top-box devices use wired Ethernet or Wi-Fi connections
  - Wake-on-LAN uses a HW filter + post-processing in firmware/OS or FW-only (ARM-TF)
  - Wake-on-WLAN uses the on-chip ARM CPU running the 802.11 stack
- Stringent EU CoC and Energy Star power requirements with network standby < 500 mW
- Network standby means that the device must wake-up and/or respond to specific packets, typically when an user starts streaming from their phone towards an Android TV device
  https://developers.google.com/cast/docs/design_checklist/cast-button

# Wake-up on mDNS

- Supported upstream by bcmgenet and bcmsysport by programming of a network filter specifically for wake-up (RX_CLS_FLOW_WAKE = -2), e.g.:

```
ethtool -N eth0 flow-type udp4 dst-ip 224.0.0.251 dst-port 5353 user-def 0x1 action -2
```

- Hardware does not support matching only the specific mDNS service of interest (e.g.: _googlecast_tcp.local) and wakes-up for *every* mDNS matching the filter, in a busy network (printers, Apple devices, IoT, etc..) this can result in very little time suspended -> power consumption increases

- Similar to Wi-Fi, make use of a smaller CPU running firmware and do a second pass filtering

- No upstream defined API for communicating the desired matching towards the FW: relies on the HW filter above to minimize false positives and then adds a hard coded string search on top -> non ideal

# mDNS offload

- Querying for a “Casting” device is not enough and can still result in false positives:
  - E.g.: phone roams from outside to inside, queries available

- Phone streaming to device is the ultimate way of identifying the intent:
  - Match on the first SYN/ACK and let TCP play through the delay in transitioning from FW to Android/OS

- Requires maintaining a database of mDNS records from the FW:
  - Need to remain power efficient until the very last intent of streaming
  - API to program that database exists in Android, non-existent in Linux yet

# Proposals

- Continue with ethtool_rx_flow_spec, extended over netlink?

```
ethtool -N eth0 flow-type udp4 dst-ip 224.0.0.251 dst-port 5353 \
    user-def 0x1 action -2 \
    strsearch “_googlecast_tcp.local” \
    strsearch “_amzn-wplay._tcp.local”
```

- Use flower for matching + additional filter for string search?
- Something else entirely?
- Consistency between cfg80211 and ethernet?