Network offloading in OpenWrt

October 2018

Hauke Mehrtens
Who am I

- **Hauke Mehrtens**
  - Working as a software engineer for Intel’s Connected Home Division
  - Active in OpenWrt project since 2009
  - Contributor to upstream Linux kernel
What is Network offloading?

- OpenWrt uses the Linux kernel networking stack

- Normally network packets go through Linux kernel networking subsystem
  - Supports many features and is very flexible and hardware independent
  - everything runs on the CPU
  - Uses only very few special hardware features by default
Hardware features

• Many network SoCs support some packet handling in hardware
  • Layer 2 (Ethernet) switching between different ports
    • Bridge offloading
    • Mac Address learning
    • Multicast forwarding
    • VLAN translation
  • Layer 3 (IP) routing / NAT
    • Network Address Translation (NAT)
    • Static routes
    • PPPoE
• Offloading => Taking advantage of the hardware features
History of Layer 2 Network offloading

• Layer 2 switch configuration supported by OpenWrt swconfig
  • One network device for the switch
  • Multiple VLANs in the switch hardware, add ports to it
  • Never went into upstream kernel

• Upstream kernel has switchdev and DSA
  • Each physical port is one network device
  • Normal network tools are used: ip, brctl, ethtool, …
  • Supported by OpenWrt
DSA (Distributed Switch Architecture)

• Introduced about 10 years ago for Marvell SOHO switches
• Based on switchdev
  • switchdev notified which can be handled by driver
• Common vendor independent device tree binding
  • DSA switch driver only implements callbacks
• Each port is a network device
  • Normal network operations and statistics
• Uses existing standard Linux interfaces
  • ip / ifconfig, brctl, ethtool, …
• Supported functions can be offloaded, others use software path
• Data packets have special tag
DSA (Distributed Switch Architecture)

- Source: https://netdevconf.org/2.1/papers/distributed-switch-architecture.pdf
DSA (Distributed Switch Architecture)

- Bridge offloading
  - Multiple bridges
  - Untagged and single tagged ports
    - VLAN filtering
  - Aging time
  - Spanning Tree Protocol (STP)
- Static flow configuration
- Multicast flow configuration
  - Linux software bridge does IGMP / MLD snooping
- Port mirroring
- Multi chip support
DSA ethtool interface (per port)

- Ethtool statistics (ethtool -S)
- Precision Time Protocol (PTP) and timestamping support
- Energy Efficient Ethernet (EEE)
- Phylib support or direct integrated
- Phylink for SFPs
History of Layer 3 Network offloading

• Each chip vendor has their own drivers, kernel hocks and tools
  • different APIs

• Never support by OpenWrt till 18.06
  • Base added to Linux kernel 4.16
  • SoC specific offloading frames were rejected by OpenWrt community
  • Integrated in OpenWrt 18.06 and later with kernel 4.14 and later
  • Hardware flow offloading supported for some SoC
Flow offloading

- Software and hardware fast path around the Linux networking stack
- Only simple traffic is offloaded, rest uses the Linux networking stack
- Learning is done by the Linux networking stack
- Supports forwarding and Network address translation (NAT)
- Detects flow with netfilter conntrack
Software flow offloading

- New packet is received
- Check if flow is known
  - based on received interface, protocol, source and destination IP address and ports
- If not found forward it to the Linux networking stack
- If found apply NAT translation if needed for this flow
- Send out on destination device
- Integrated in mainline Linux since version 4.16
  - backported in OpenWrt to kernel 4.14
Hardware flow offloading

- Allows to add flows to hardware in addition to software
- Driver implements ndo callback
- If flow not supported by hardware, software path will be used
- Hardware flow offloading not yet in mainline
  - Waiting for first driver to use it
- Included in OpenWrt for kernel 4.14
  - Extended to support bridges, VLANs, PPPoE

- Following UCI config options in firewall default section
  - flow_offloading
  - flow_offloading_hw
Conclusion

• Finally initial part in mainline Linux kernel and OpenWrt
• Vendor neutral implementation
• Integrates with the rest of the Linux kernel
  • No special Interfaces
  • No out of tree patches to add own network hooks

• Future
  • More hardware will be supported
  • Hopefully everything will go into mainline Linux kernel
  • QoS with offloading

• Thank you Pablo, Felix and John for implementing flow offloading
Questions?