Netdev 0x15
2021-07-14

QRATOR LABS

Switchdev: the Good and the Ugly

Alexander Zubkov
green@qrator.net

Variety

- “small” equipment (ex. CPEs)
- high-throughput switches (ex. TOR)
  - our case
  - core/border router
  - Mellanox Spectrum chipset (ex. 16x100G)
- different needs for operations
  - CPE: one-time configuration, no problem to reboot
  - TOR: runtime reconfiguration, reboot not welcome

The Good

- vanilla kernel
- any Linux distro
- standard tools & APIs
  - routing daemons
  - monitoring
  - automation
- for free
- it is worth the effort

The Ugly

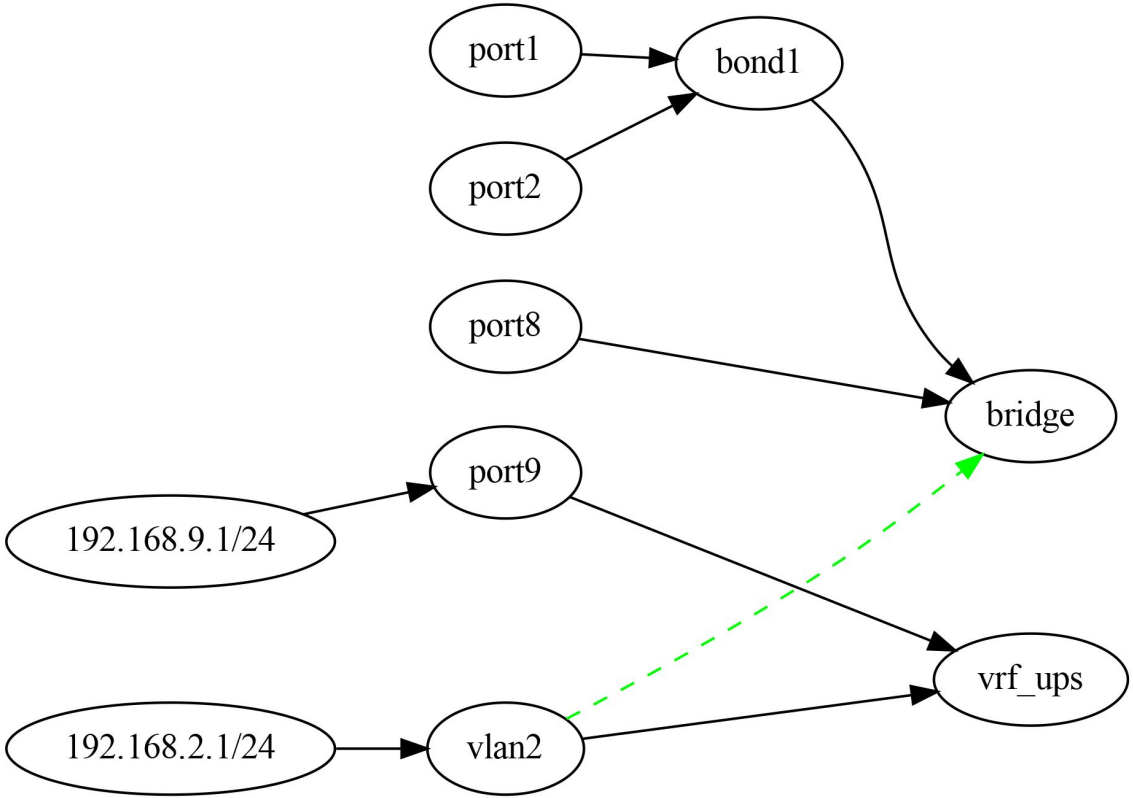
- but here come operations
- configure interfaces & routing
  - port split
  - bonding
  - vlans
  - vrfs
- reconfigure the running switch

Configuration simplified

Tool requirements

- support:
  - port split, bonds, vlans, vrfs, v4/v6, blackhole, ecmp
- human-readable
- apply changes in runtime
- reduce mistakes
- stateless
- show running configuration (running → startup config)
- keep management interface configuration separate

Example setup



bash: example (1/2)

```
ip link add name bond1 type bond lacp_rate fast mode 802.3ad
ip link set dev bond1 down
ip link add name switch type bridge vlan_filtering 1
ip link set dev switch down
ip link add name vrf-ups type vrf table 10
ip link set dev vrf-ups down
ip link set dev port1 master bond1
ip link set dev port1 down
ip link set dev port2 master bond1
ip link set dev port2 down
ip link set dev port8 master switch
ip link set dev port8 down
ip link set dev port9 master vrf-ups
ip link set dev port9 down
ip link set dev bond1 master switch
ip link set dev bond1 down
ip link add link switch name vlan2 type vlan id 2
ip link set dev vlan2 down
```

bash: example (2/2)

```
ip link set dev vlan2 master vrf-ups
ip link set dev vlan2 down
bridge vlan del vid 1 dev bond1
bridge vlan add vid 2 dev bond1 pvid untagged
bridge vlan del vid 1 dev port8
bridge vlan add vid 2 dev port8 pvid untagged
bridge vlan add vid 2 dev switch self
ip link set dev port1 up
ip link set dev port2 up
ip link set dev port8 up
ip link set dev port9 up
ip link set dev bond1 up
ip link set dev switch up
ip link set dev vlan2 up
ip link set dev vrf-ups up
ip -4 address add 192.168.9.1/24 dev port9
ip -4 address add 192.168.2.1/30 dev vlan2
ip -4 route replace blackhole 0.0.0.0/0 metric 4278198272 table 10
```

- startup (initialization) script
  - by hand, generate (templates, netpen.io, …)
- support everything
- can be readable in some degree
- reconfigure by hand
- human mistakes possible
- copy changes to startup by hand

ifupdown-ng: example (1/2)

```
# /etc/network/interfaces:

auto bond1
iface bond1
    use bond
    bond-members port1 port2
    bridge-access 2

auto port8
iface port8
    bridge-access 2

auto port9
inface port9
    vrf vrf_ups
    address 192.168.9.1/24
```

ifupdown-ng: example (2/2)

```
auto switch
iface switch
    use bridge
    bridge-ports bond1 port8
    bridge-stp off
    bridge-vlan-aware yes
    bridge-vids 2

auto vlan2
iface vlan2
    vlan-raw-device switch
    vrf vrf_ups
    address 192.168.2.1/24

auto vrf_ups
iface vrf_ups
    vrf-table 10
```

ifupdown-ng

- https://github.com/ifupdown-ng/ifupdown-ng
- extended with modules, custom commands
- somewhat readable
- saves custom state, may come out of sync
- very bad with applying changes
- can not show running configuration

ifupdown2: example (1/2)

```
# /etc/network/interfaces:

auto bond1
iface bond1
    bond-slaves port1 port2
    bond-mode 802.3ad
    bond-lacp-rate 1
    bond-min-links 1
    bridge-access 2

auto port8
iface port8
    bridge-access 2

auto port9
iface port9
    vrf vrf_ups
    address 192.168.9.1/24
```

ifupdown2: example (2/2)

```
auto switch
iface switch
    bridge-ports bond1 port8
    bridge-vlan-aware yes
    bridge-stp off

auto vlan2
iface vlan2
    vlan-raw-device switch
    vlan-id 2
    vrf vrf_ups
    address 192.168.2.1/24

auto vrf_ups
iface vrf_ups
    vrf-table 1010
```

- https://github.com/CumulusNetworks/ifupdown2
- extended with modules, custom commands
- somewhat readable
- stateless
- can apply changes
- can show running configuration
- made by Cumulus, should have similar requirements

mlxtoolkit: example

```
[bond 1]
slave port 1, port 2

[port 9]
vrf ups
ip 192.168.9.1/24

[vlan 2]
native port 8, bond 1
vrf ups
ip 192.168.2.1/30

[vrf ups]
table 10
```

- https://gitlab.com/qratorlabs/mlxtoolkit/
- fixed set of features and options
  - port split
- more compact config
- stateless
- can apply changes
- can show running configuration with limitations
- our in-house tool

- https://nmstate.io/
- offers declarative YAML config
- uses NetworkManager
- missing vrf, vlan-aware-bridge

netifd (OpenWrt)

- https://openwrt.org/docs/techref/netifd
- promises runtime changes
- almost no documentation
- tied to OpenWrt infrastructure: ubus, uci, procd
- missing vrf, vlan-aware bridge

Other variants

- os-net-config (OpenStack)
  - uses backend provider (ifupdown, …)
- systemd-networkd
- ?

Tool requirements

- …
- atomic changes
  - support in kernel, API, …
- automation
  - Ansible plugins
  - NETCONF/YANG

NETCONF/YANG

- OpenWrt
  - https://openwrt.org/inbox/howto/opencpe
- nmstate
  - concept?
  - https://nmstate.io/devel/design/networking-api.html#appendix-d-netconfyang

Poll (TOR)

NetworkManager	:|	#	Fedora, RedHat
bash	:|	#	Debian
bash	:|	#	Centos
bash	:|	#	Ubuntu
mlxtoolkit	:|	##	Arch, CentOS, Debian, NixOS, Ubuntu
mlxtoolkit	:)	##	Gentoo
proprietary	:(	##	CentOS, RedHat
proprietary	:)	###	Cumulus
proprietary	:)	###	Ubuntu

Poll (CPE)

bash	:)	#	Buildroot
ifupdown	:)	#	Debian, Timesys
uci			OpenWrt

The story so far

- there are tools, but they have their drawbacks
- there are ways to improve
- if you did not find suitable and made an in-house tool
  - you are not alone
  - why not opensource it?
- and that is only one part of the story

- Mellanox Spectrum
  - tc (qdisc, filter)
  - physical ports only
- requirements are mostly the same
- tools?

ACL/bash: example script

```
tc filter add block 100 …

… protocol ip chain 101 pref 1 flower ip_proto icmp action pass
… protocol ip chain 101 pref 2 flower ip_proto tcp action pass
… protocol ip chain 101 pref 3 flower action drop
… protocol ipv6 chain 101 pref 4 flower action drop
… protocol ip chain 100 pref 1 flower ip_proto icmp dst_ip 192.0.2.2 action
pass
… protocol ip chain 100 pref 2 flower src_ip 203.0.113.0/24 action drop
… protocol ip chain 100 pref 3 flower dst_ip 203.0.113.0/24 action goto
chain 101
… protocol ip chain 100 pref 4 flower dst_ip 203.0.113.0/24 action drop
… protocol ip chain 100 pref 5 flower action pass
… protocol ipv6 chain 100 pref 6 flower action drop
… protocol 802.1q chain 0 pref 1000 flower vlan_id 10 action goto chain 100
… protocol 802.1q chain 0 pref 1001 flower action pass
(12 lines)
```

ACL/bash: example show (1/2)

```
filter protocol ip pref 1 flower chain 100
filter protocol ip pref 1 flower chain 100 handle 0x1
  eth_type ipv4
  ip_proto icmp
  in_hw in_hw_count 1
    action order 1: gact action pass
     random type none pass val 0
     index 1 ref 1 bind 1
    used_hw_stats immediate

filter protocol ip pref 2 flower chain 100
filter protocol ip pref 2 flower chain 100 handle 0x1
  eth_type ipv4
  ip_proto tcp
  in_hw in_hw_count 1
    action order 1: gact action pass
     random type none pass val 0
     index 2 ref 1 bind 1
    used_hw_stats immediate
```

ACL/bash: example show (2/2)

```
filter protocol ip pref 3 flower chain 100
filter protocol ip pref 3 flower chain 100 handle 0x1
  eth_type ipv4
  in_hw in_hw_count 1
    action order 1: gact action drop
     random type none pass val 0
     index 3 ref 1 bind 1
    used_hw_stats immediate

...
filter protocol 802.1Q pref 1001 flower chain 0
filter protocol 802.1Q pref 1001 flower chain 0 handle 0x1
  in_hw in_hw_count 1
    action order 1: gact action pass
     random type none pass val 0
     index 12 ref 1 bind 1
    used_hw_stats immediate
(114 lines)
```

ACL/bash

- write only
- changes in runtime by hand
  - good luck

ACL/mlxtoolkit: example

```
[vlan10]
ip_proto icmp dst_ip 192.0.2.2 action pass
src_ip 203.0.113.0/24 action drop
dst_ip 203.0.113.0/24 action goto [ex1]
dst_ip 203.0.113.0/24 action drop
action pass

[ex1]
ip_proto icmp action pass
ip_proto tcp action pass
action drop
```

ACL/mlxtoolkit

- https://gitlab.com/qratorlabs/mlxtoolkit/
- fixed acl model
- compact readable config
- stateless
- can apply changes
- can show running configuration
  - loses auxiliary chain names
- this tool was made in the first place

ACL/mlxtoolkit: model

- one shared acl for all ports
  - tc block
- chain per vlan
- main chain (0): match vlan

- My contacts
  - Alexander Zubkov
  - green@qrator.net