Hardware switches - the open-source approach

Jiří Pírko
jiri@resnulli.us
Red Hat

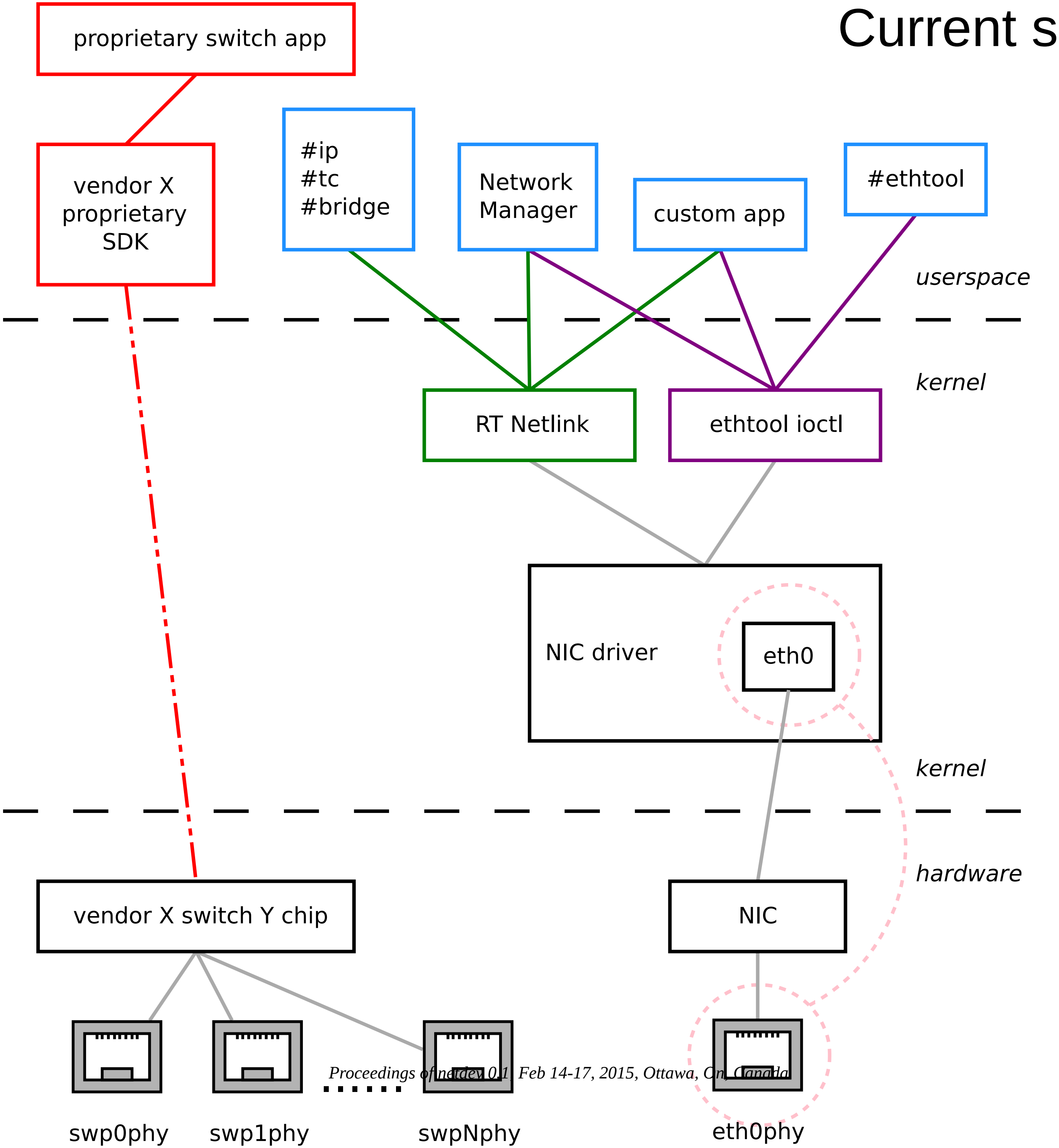
Scope of talk

- Open-source Linux support for various switch and switch-ish chips.
  - Including L2, L3, flow-based forwarding
- TOR (Top-of-rack switch)
- Switch chips in servers
  - Mesh topologies
  - Could replace TORs
- SR-IOV
  - Switch embedded into NIC
  - Used for virtualization purposes
- Home routers
  - e.g. OpenWRT devices
- ~~Custom switch board Linux deployment~~

Current state

- Ice age
- Switch chip vendors
  - Broadcom, Intel, Mellanox, ...
  - They believe they need to protect their “intellectual property”
  - Each has its own “SDK” - userspace binary blob user for accessing HW
    - Vendor lock-in for appliance vendors
- Appliance vendors (boxes)
  - Cisco, Juniper, Brocade, ...
  - They buy chips from others and include them into their products
  - Proprietary tools for switch chip manipulation
    - Vendor lock-in for customers
  - Often use Linux kernel, however not for switch chip manipulation

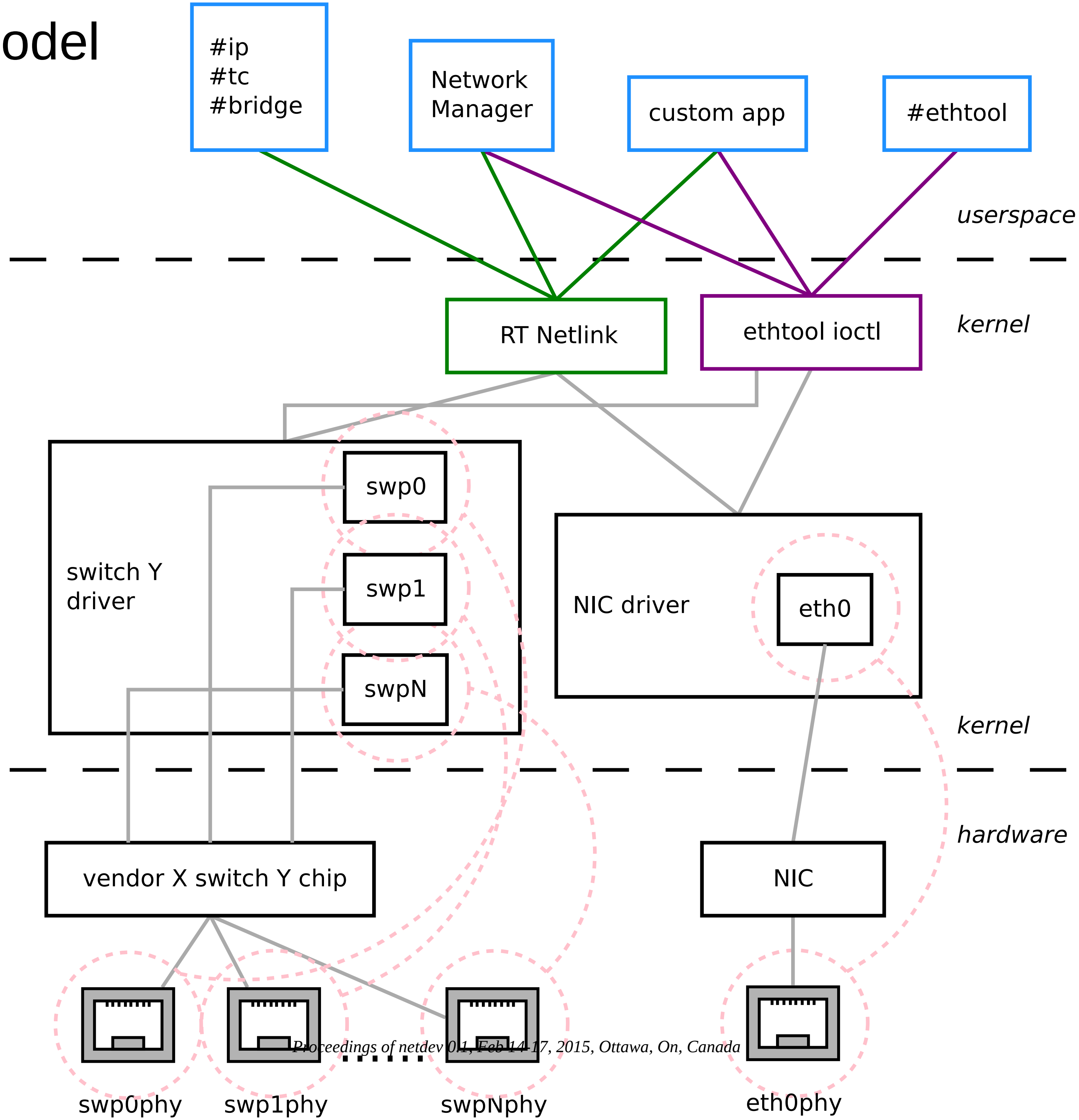
Current state



Desired model

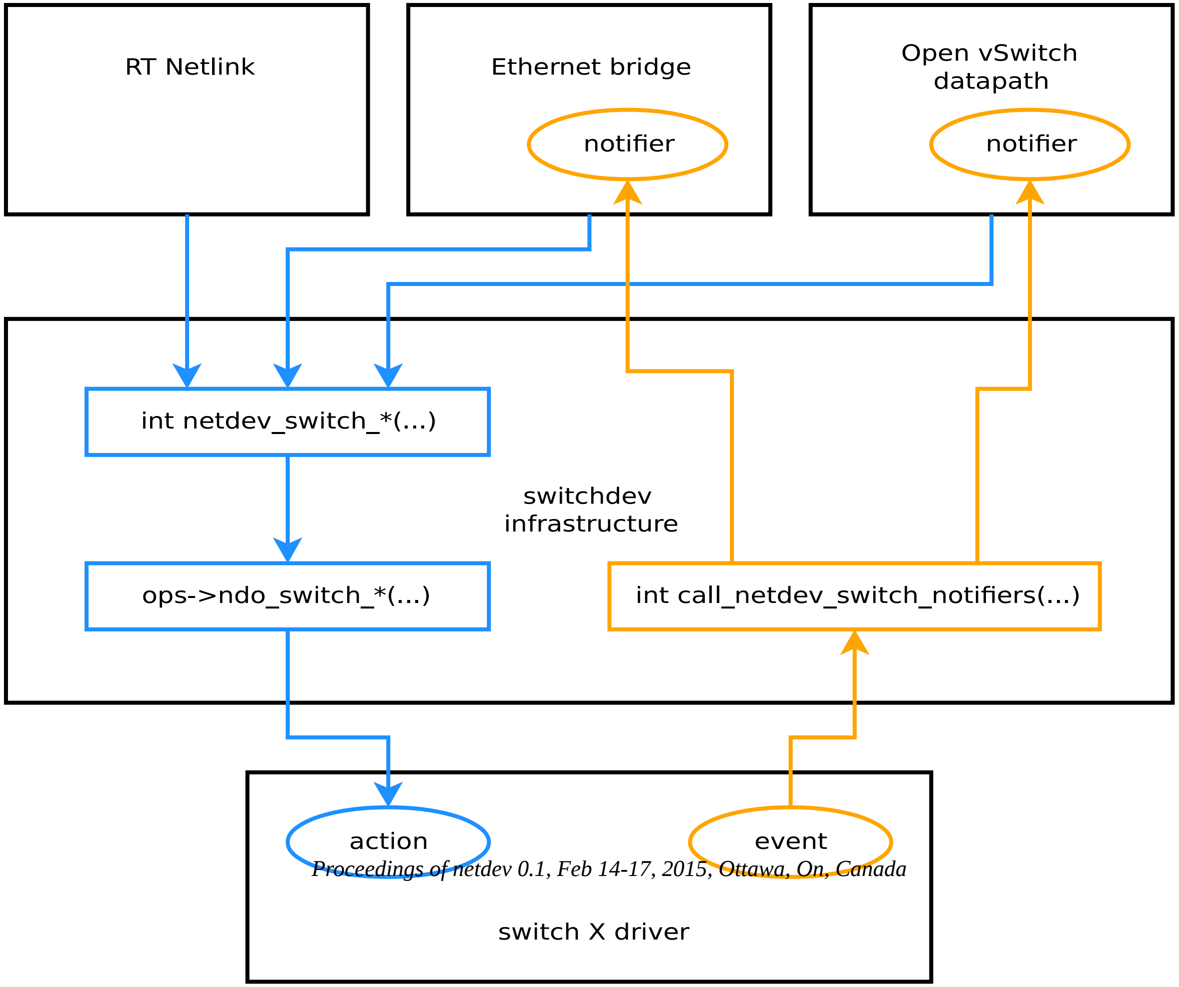
- Possibility to re-use existing network tools for switches
  - *ip*, *ethtool*, *bridge*, *tc*, Network Manager, open vSwitch toolset
- One switch port is represented as one network device (e.g. eth0)
- Port devices should be able to work as independent NICs
  - L3 address assign, packet TX and RX
  - Routing between ports could be offloaded into hardware
- Port devices should work in layered topologies
  - Layered devices: bridge, bonding, Open vSwitch
  - Offload layered devices functionality to hardware if possible
- Ethtool API implementation by driver
- Provide a way to find out if two ports belong to the same switch chip
- Model working name is “switchdev”

Desired model



Linux Switchdev infrastructure

- Switch device specific set of network device operations (ndos)
  - To pass info to switch driver and also to query driver for some information
- Switch device notifier
  - To propagate hardware event to listeners



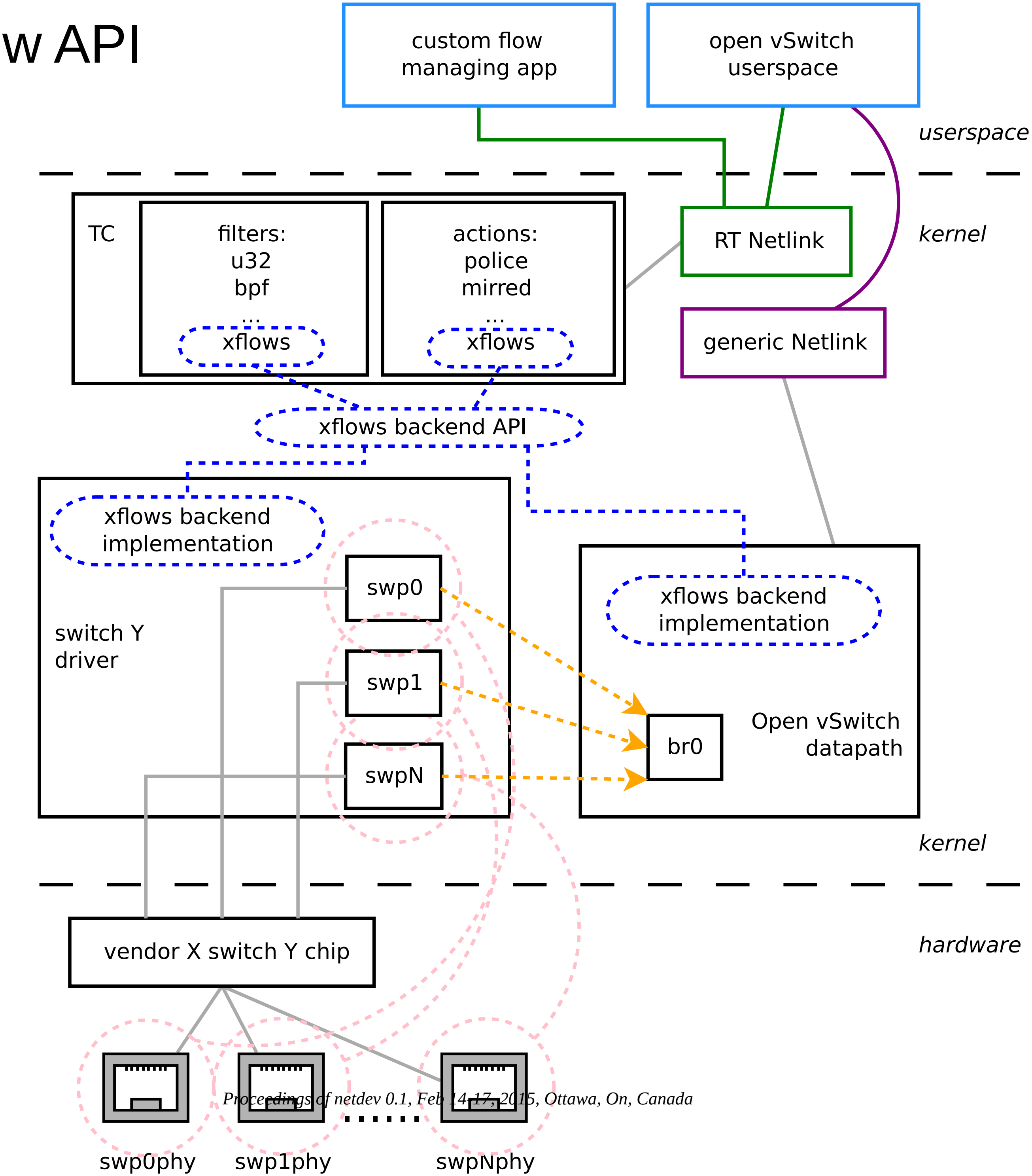
L2 forwarding offload

- Merged into upstream Linux kernel
  - Linux bridge support
  - Rocker switch driver
    - Rocker switch is hardware emulated in QEMU based on OF-DPA model
    - Rocker was created for testing and prototyping purposes
- Two new ndos introduced
  - *ndo_switch_parent_id_get*
    - Called to obtain ID of a switch port parent (switch chip)
  - *ndo_switch_port_stp_update*
    - Called to notify switch driver of a change in STP state of bridge port
- Two new switchdev notifier events introduced
  - *NETDEV_SWITCH_FDB_ADD* and *NETDEV_SWITCH_FDB_DEL*
    - Raised by switch driver in case hardware an FDB entry is added or removed

Future plans

- L3 forwarding offload - an attempt by Scott Feldman
  - Introduction of two new ndos
    - *ndo_switch_fib_ipv4_add* and *ndo_switch_fib_ipv4_del*
      - Called by the core IPv4 FIB code when installing/removing FIB entries to/from the kernel FIB
- Flow-based forwarding offload - an attempt by John Fastabend
  - Called “Flow API”
  - Introduces a new Generic Netlink interface called “net_flow_nl”
    - To be used for offloaded flows maintenance only
  - Userspace app queries hardware capabilities and do the flow insertions accordingly
- TC-based flow offload
  - An alternative to “Flow API”
  - Extends existing TC Netlink API
  - The same interface for software datapath and hardware offload

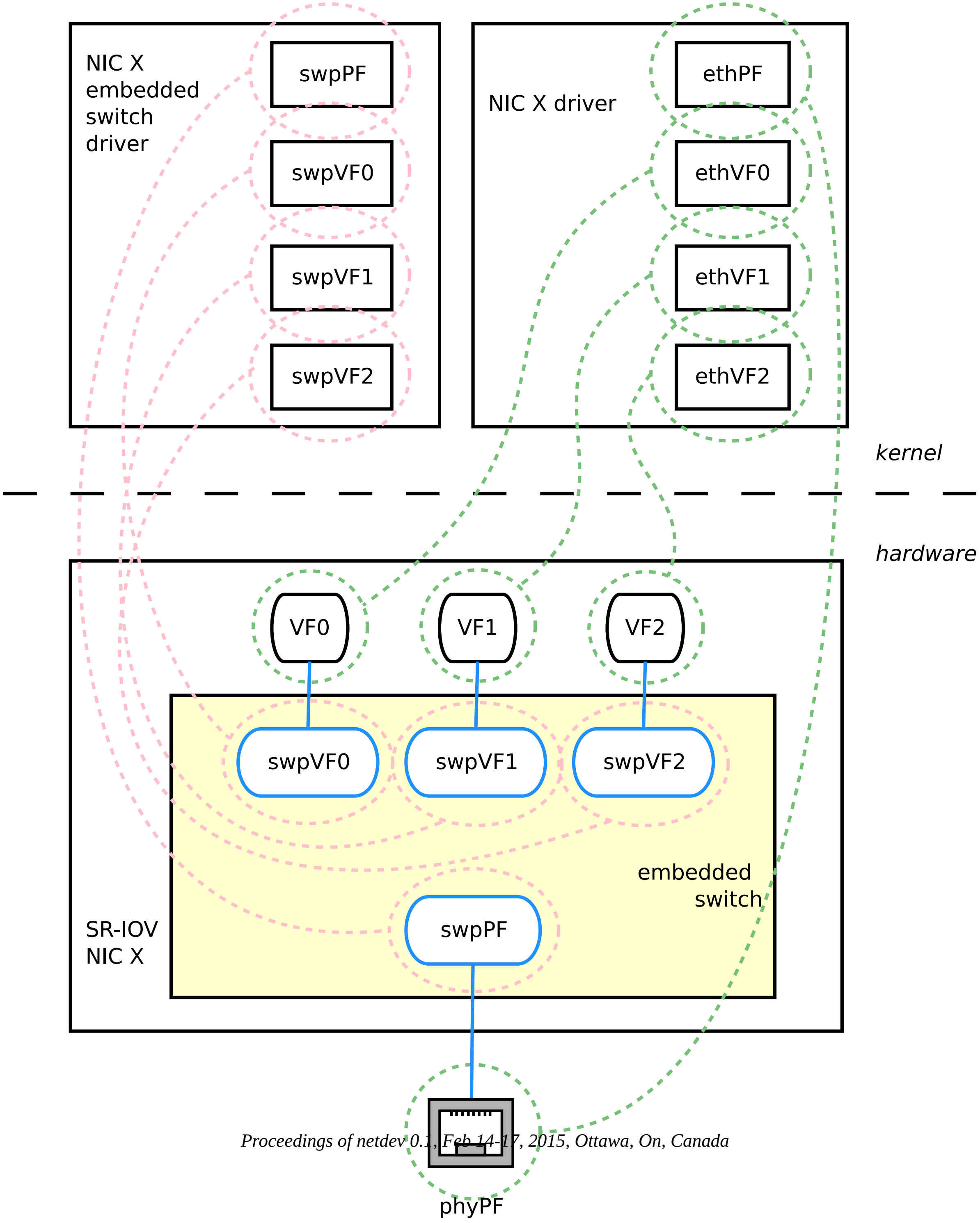
TC-based flow API



SR-IOV use-case

- Embedded switch
  - Interconnects VFs and PF
  - Capabilities differ from NIC to NIC
  - From Linux kernel perspective should be handled like any other switch chip
    - Purpose of switchdev is to provide that abstraction
  - Lot of potential for virtualization use-cases
    - Open vSwitch acceleration
    - Containers, OpenStack

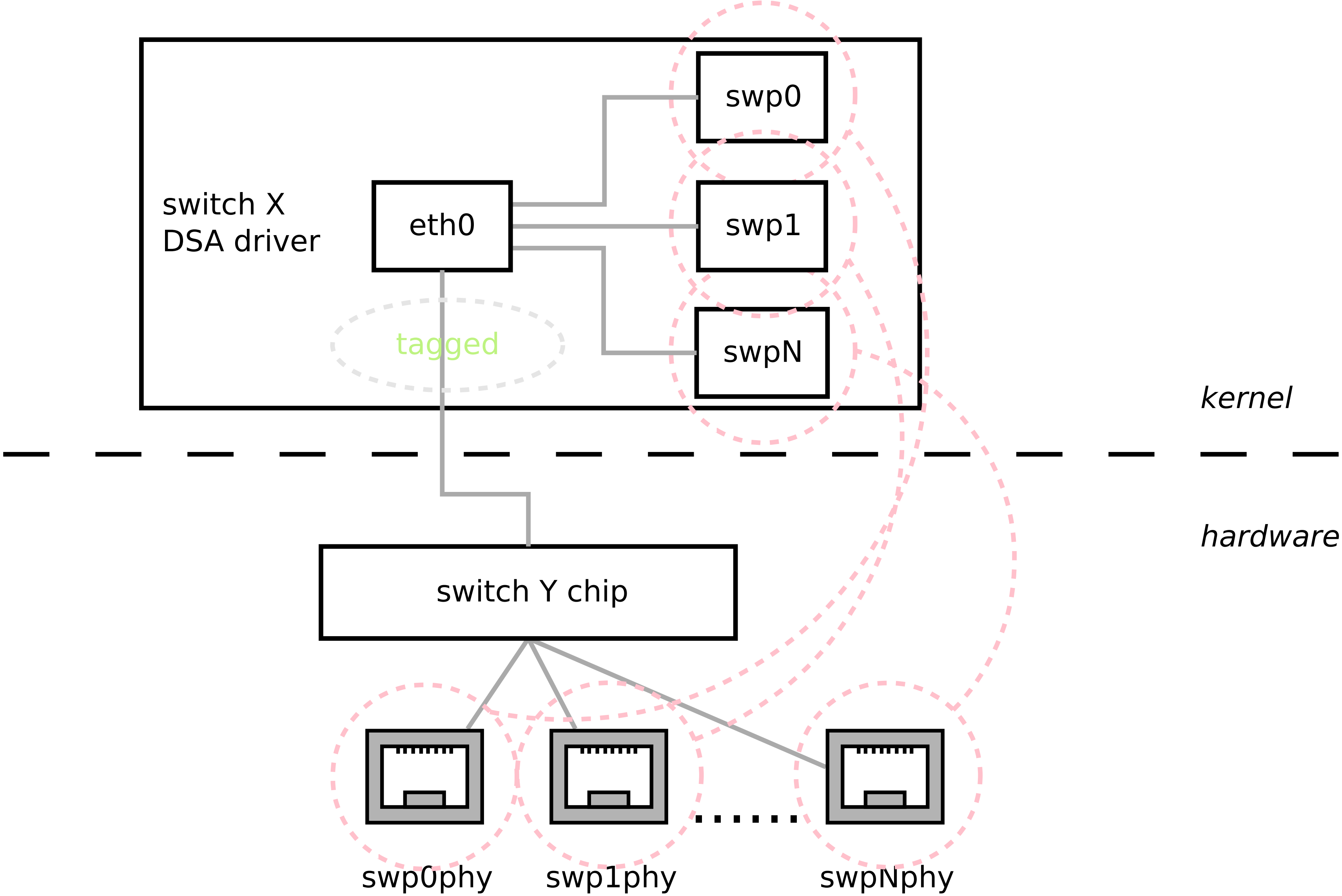
SR-IOV use-case



DSA use-case

- Switch PHY
  - Connected via MII
  - Allows to rx and tx packets via particular ports using “DSA tags”
  - In kernel, for each port there is a netdevice created
  - Fits into the switchdev picture
    - looks like any other switch driver exposing switch ports

DSA use-case


switch X
DSA driver
eth0
swp0
swp1
swpN
tagged
kernel
hardware
switch Y chip
swp0phy
swp1phy
swpNphy

The end

- Questions?