Making Networking Queues a First Class Citizen in the Kernel

Magnus Karlsson & Björn Töpel, Intel
Jesper Dangaard Brouer & Toke Høiland-Jørgensen, RedHat
Jakub Kicinski, Netronome
Maxim Mikityanskiy, Mellanox
Andy Gospodarek, Broadcom
work in progress
Motivation

- Users in the kernel:
  - The Linux stack
  - XDP_TX and XDP_REDIRECT actions (hidden)
  - AF_XDP (hidden)
  - Qdisc with HW offload (hidden)

- User-space APIs:
  - AF_XDP: bind(ifindex, qid)
  - Ethtool
  - /sys/class/net/<dev>/queues/{Rx,Tx}-N/

**Problem:** What queue id to provide?
Outline

- Problem scoping and queue definition
- Interface proposal
- Usage examples
- Design proposal and implementation plan
- Challenges and open questions
Queue HW Basics

Two problems:
- Splitting up queues between PFs and VFs in a device
- Allocating and freeing queues within a netdev
Our Focus

Two problems:
- Splitting up queues between PFs and VFs in a device
- Allocating and freeing queues within a netdev
Queue Definition

- Unidirectional: Rx or Tx, not both
- Tied to a HW device
- Referenced by ifindex,qid. Qid unique within a device
- Belongs to a single netdev => single namespace
- Always refers to a real HW queue (for physical devices)
- The queue id (qid) is opaque in user space
Interface Proposal

- Netlink interface

- NETLINK_CMD_QUEUES_LIST - List all used queues of an ifindex/netdev
- NETLINK_CMD_QUEUE_ALLOC - Allocate a queue
- NETLINK_CMD_QUEUE_GET - Get an attribute of a queue
- NETLINK_CMD_QUEUE_SET - Set an attribute of a queue
- NETLINK_CMD_QUEUE_FREE - Free a queue
NETLINK_CMD_QUEUE_ALLOC

W = written, input data
R = read, output data

- W ifindex: the interface this queue should be allocated from
- W name: an optional name for this queue
- RW irq = if not provided, associate this queue with an unused irq and return the irq number. If provided, associate this queue with this irq.
- W type = tx | rx: should the queue be Tx or Rx. Both not allowed.
- R qid: Returns the qid of the queue
- R error
NETLINK_CMD_QUEUES_LIST

W = written, input data
R = read, output data

- W ifindex: the interface this queue should be allocated from
- W name, qid, or irq: search by name, qid, irq or type
- R name, qid, irq and type
- R error
Example Usage: AF_XDP

- Allocate queue pair affinitized to a specific core and bind an AF_XDP socket to it

  NETLINK_CMD_QUEUE_ALLOC ifindex1, name_rx, rx => qid_rx, irq_rx
  NETLINK_CMD_QUEUE_ALLOC ifindex1, name_tx, tx, irq_rx => qid_tx
  echo "2" > /proc/irq/<irq_rx>/smp_affinity
  bind(fd, ifindex1, qid = qid_rx qid_tx = qid_tx)

- bind(fd, ifindex1, qid = qid_rx) would pick a Tx queue for you, just like today
Example Usage: Ethtool

- Channel = all queues tied to the same irq
  - Numbered 0...real_num_{rx|tx}_queues - 1
- But queue API produces two qids that are opaque!

Solution proposal:
- Linux stack Rx queues always have qid = 0...real_num_rx_queues – 1
  - Channel N = Linux stack Rx qid N
- Ethtool looks up irq of supplied qid. List all queues using that irq
- Or does ethtool have a better interface?
- New Ethtool interfaces possible
- What to do with Tx only channels?
Example Usage: XDP_REDIRECT

- Allocate a Tx queue for XDP_REDIRECT

/* NOTE: details to be ironed out */
NETLINK_CMD_QUEUEALLOC ifindex1, name_tx, tx => qid_tx
struct queue_target tgt = { .queues[0] = qid_tx, .mode = QUEUE_SINGLE};
bpf_map_update(queue_map, 0, &tgt);

/* In BPF program */
xdp_redirect_map(queue_map, 0, FLAGS);
- Not reusing \_rx and \_tx arrays. New structure needed
- Tie into existing interfaces, e.g. netif\_set\_real\_num\_rx\_queues()
- Qids can be decided by driver
  - For backwards compatibility and encoding queue types
- New alloc and free ndo:s in driver needed
Implementation Plan

- Netlink (+ sysfs) interface
- Queue manager module in the kernel that keeps track of queues, allocations and deallocations
- Show Linux stack queues in netlink interface (netif_alloc_netdev_queues + netif_alloc_rx_queues)
- Show XDP_TX queues in netlink interface
- Show qdisc mqprio HW offloaded queues in netlink interface
- Implement alloc and free queue ndo:s in drivers. Start with a single driver.
- Implement libbpf helpers for creating a new socket tied to a new dedicated AF_XDP queue.
- Update xdpsock app to use this
- iproute2 support for queue manipulation
- Update all three drivers currently supporting AF_XDP zero-copy
- Update macvlan to allocate queues using this new interface
- (Move XDP_TX queue creation policy outside of driver)
- (Move Linux stack queue creation policy outside of driver)
Challenges & Open Questions

- Interactions with changing the number of queues in ethtool
  - Reserved qid space that we do not allocate from, or do we?
- Can ethtool use this interface or do we use ethtool’s one?
- Do we also need a sysfs interface in /sys/class/net/<dev>/queues/?
  - Functionally equivalent or read-only?
- What queue properties should be exposed?
- Exposing napi as well?
- Do we at some later stage support virtual queues?
- What to call the "queue manager"?
Next Steps

- Incorporate all your feedback
- Post interface proposal to mailing list
- Patch of three first steps (netlink + queue manager + show Linux stack queues)