Netlink APIs to Expose Netdev Objects

Oct 2023, Netdev 0x17

Amritha Nambiar

Sridhar Samudrala

Agenda

- Configuring Netdevices
- Netdev Configuration Interfaces
- YAML netlink
- Netdev-genl (YAML Netlink for Netdev)
- Queue Configuration
- NAPI Configuration
- Page Pool Configuration
- Generic Queue Management
- Summary

Configuring Netdevices

- Netdevs are becoming complex – more functionality, more configurability
- Multitude of netdevice features
- Multiple interfaces to configure
- Requires a stable, generic and scalable API
  - To keep up with such complexity
  - For user controllability
  - Stronger generic models in the kernel abstract away driver specific implementations (avoids code duplication and noticeable differences between vendors)

Netdev Properties are Growing

Device configuration from Netdev for multiple properties such as:

- Queue management and queue properties
- NAPI parameters
- Statistics
- RSS
- Interrupt Moderation
- Traffic/flow steering
- Traffic shaping, rate limiting
- Stateless offloads (Checksum, GRO, TSO, USO, Vlan strip/insertion etc .)
- XDP
- Zerocopy and header data split per queue/queue-group configurations
- HW/netdev/driver dumps and attestation
- Scaling - aRFS/RFS/RPS/XPS
- Tunneling
- Link information

Netdev Configuration Interfaces

- Standards based upstream APIs used by multiple vendors:
  - Functionality (enable/disable)
  - Performance (parameters values, ranges, user specific demands per feature)
  - Debugging (report device diagnostic data, dumps, device attestation)

- TC
- Devlink
- Ethtool
- Sysfs

- Netfilter
- Ip
- Bridge
- Socket options

One size does not fit all

- High configurability issues are not new
  - Kubernetes: Container Network Interface (CNI) Specification plugins
    - plugin is a program that applies a specified network configuration
  - OpenStack
    - “OpenStack SDK is implemented as an extensible core, upon which vendor extensions can be plugged in”
- Targeted APIs. Which the orchestration layer in user space would have to combine
- For new features, netlink (netdev-genl) can be a useful candidate
  - As attributes of different netdev objects

Netlink protocol specifications (in YAML)

- Reduces developer involvement in netlink coding
- Truly generic netlink libraries (no changes to support a new family or a new operation)
- Netlink messaging and attributes in YAML, codegen outputs user and kernel code (parsing, validating, documentation etc.)
- devlink, ethtool, netdev, DPLL, FOU, TLS handshake, OVS, address/route/link configuration

•Kernel uses the YAML specs to generate:

  - the C uAPI header
  - documentation of the protocol as a ReST file
  - policy tables for input attribute validation
  - operation tables

•Compatibility – Genetlink

Using Netlink protocol specifications

- Python CLI tool in kernel
- YAML specification to issue netlink requests to the kernel (Documentation/netlink/specs/)
- CLI arguments:
  - --spec - point to the spec file
  - --do $name / --dump $name - issue request $name
  - --json $attrs - provide attributes for the request
  - --subscribe $group - receive notifications from $group
- JSON input arguments, output Python-pretty-printed
- Generating kernel code - tools/net/ynl/ynl-regen.sh
- Generating user code - YNL lib (libmnl based C library) integrates with python code generator to create netlink wrappers)

Netdev-genl (YAML Netlink for Netdev)

- Generic netlink interface for configuring netdevice features (uses netdev.yaml spec)
- Request types:
  - GET, SET (notifiers)
- Operation types:
  - Do
  - Dump (all netdevs or filtered dump for single netdev)
- Currently, netdev-genl exposes XDP features, XDP Rx metadata etc.
- Other Netdev objects (APIs) are WIP:
  - Queue
  - NAPI
  - Page pool

Queue Configuration

- Goals:
  - Queue object for an abstract queue configuration model in the core
  - Simplify modifications, reconfiguration for drivers, validations in core, export configurations to user
- Basic queue parameters:
  - Queue index
  - Queue type (Rx, Tx, XDP-Tx)
  - Interface index of netdevice to which the queue belongs
  - NAPI id (NAPI instance servicing the queue)

Queue Configuration

- More per queue attributes possible:
  - Statistics
  - Ring size (descriptor number)
  - Header-data split for Rx zerocopy (per queue configs, disable for AF_XDP queue)
  - Page pool id (Associate a page pool to a queue)
    - Memory model (dma-buf, kernel vs user buf, NUMA node binding for queue, device NUMA node vs app NUMA node etc.)
  - RSS context handling (map queue to RSS context for container partitioning)
  - XDP – expose XDP_Tx queues

NAPI Configuration

- NAPI instance as netlink object
- Exposing NAPI attributes
  - NAPI id
  - Interface index of netdevice to which NAPI instance belongs
  - Interrupt number associated with the NAPI instance
  - PID of the NAPI thread
- Add/Extend:
  - Poller timeout as NAPI attribute
- Usecases:
  - Adjusting the NAPI thread priorities and SMP affinity
  - Configure NAPI pollers to queues from the userspace (limit the number of NAPI instances, 1 poller <-> queues (on the same interrupt vector)

Page Pool Configuration

- Expose page pool information via netlink
  - Unique ID of a Page Pool instance
  - Interface index of the netdev to which the pool belongs
  - NAPI id of the instance using this page pool
  - Memory use (number of references to this page pool, bytes/amount of memory held by pages, when page pool was destroyed)
  - Page pool statistics
- Add/Extend:
  - dma-buf (kernel vs user buf)
  - NUMA node binding for pages, device NUMA node vs app NUMA node etc.

Queue management

- Goal:
  - Generic dynamic queue management APIs (create, delete, start, stop)
  - Allow creation of queues without full device reset
    - Existing queues continue Tx/Rx, additional queues created
    - Queue-ids sequential or from userspace?
  - Align multiple models:
    - Process specific queues without zerocopy (repurpose existing queues)
    - Process specific queues from userspace with zerocopy (memory alloc/free from process)
    - Default driver queues for generic workload
- Requirements :
    - Queue object maintains full expected configuration and driver creates queues
    - Page pool configuration (pp created by driver, userspace update/configure parameters)
    - Associate dma-buf (or host memory buffer) with a page-pool
    - Associate queue with page-pool

Example APIs

GET:

- queue-get

```
{'ifindex': 12, 'napi-id': 593, 'queue-id': 0, 'queue-type': 'tx’}
```

- napi-get

```
{'ifindex': 12, 'irq': 291, 'napi-id': 593, 'pid': 3817}
```

- page-pool-get

```
{'id': 10, 'ifindex’: 12, 'napi-id’: 593}
```

SET:

- queue-set:

```
queue-set Q_IDX_1 napi-id NAPI_ID_1
queue-set Q_IDX_2 napi-id NAPI_ID_1
```

CREATE:

- queue-create:

```
queue-create $Q_IDX napi-id $NAPI_ID
```

Summary

Netdevs are becoming complex with more functionality, more configurability and requires a stable, generic and scalable API

Targeted feature-specific APIs, Netlink for Netdev configuration (netdev-genl)

Queue, NAPI, Page pool as netlink objects for Netdev configuration


intel