CUMULUS

Netlink Workshop

netdev 0x13 - March 20, 2019

David Ahern

Background

Netlink

Generic messaging between userspace and kernel

- RFC 3549

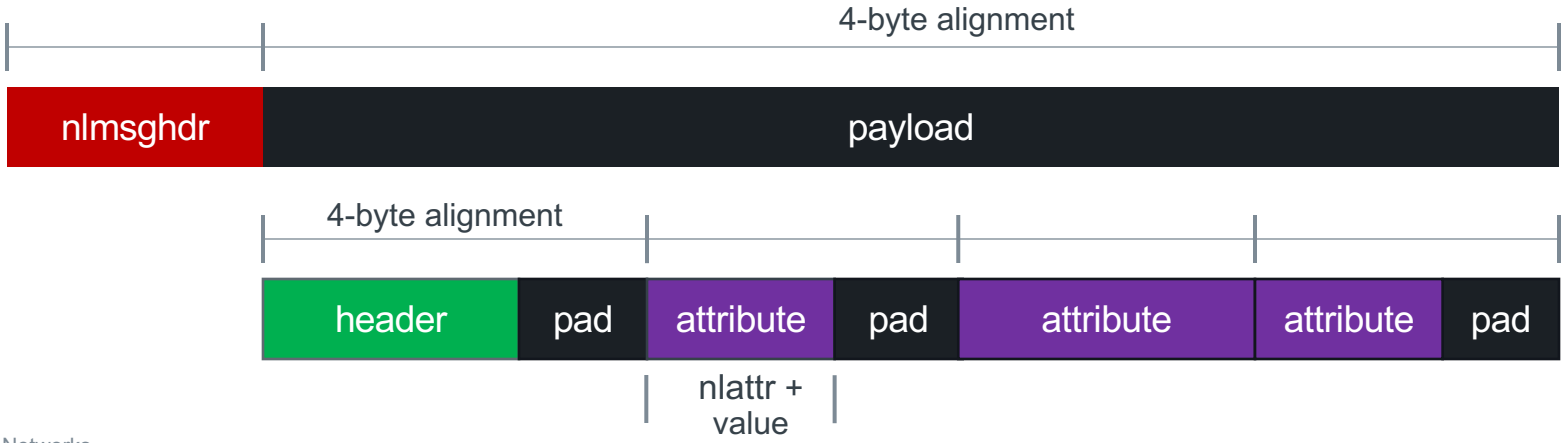
Typical socket API

- socket(AF_NETLINK, SOCK_RAW, protocol);
- protocol is one of NETLINK_*
  eg., NETLINK_ROUTE, NETLINK_GENERIC

Netlink Messages

struct nlmsghdr header followed by payload

- payload typically has a header based on message type
- attributes are TLV: struct nlattr *(type + length)* followed by value
- 4-byte alignment of each segment



Message Types Example - rtnetlink

rtnetlink uses NEW, DEL, GET, SET scheme for each object type

Objects

- LINK, ADDR, ROUTE, NEIGH, RULE, …

Example message types

- RTM_NEWLINK, RTM_DELLINK, RTM_GETLINK, RTM_SETLINK
- RTM_NEWROUTE, RTM_DELROUTE, RTM_GETROUTE, RTM_SETROUTE

Not all actions supported by all objects

Inner Header Example - rtnetlink

Each **OBJECT** has its own header struct

- Address family *must* be first entry
  AF_UNSPEC, AF_INET, AF_INET6
- Needs pad if not 4-byte aligned
- **Should** be the struct used for messages in the set

```
struct ifinfomsg {
        unsigned char   ifi_family;
        unsigned char   __ifi_pad;
        unsigned short  ifi_type;
        int             ifi_index;
        unsigned        ifi_flags;
        unsigned        ifi_change;
};
```

Message type	struct
RTM_*LINK	ifinfomsg
RTM_*ADDR	ifaddrmsg
RTM_*ROUTE	rtmsg
RTM_*NEIGH	ndmsg
RTM_*RULE	fib_rule_hdr
RTM_GETNEIGHTBL	ndtmsg
RTM_*NETCONF	netconfmsg
RTM_*NSID	rtgenmsg
RTM_*STATS	if_stats_msg
RTM_*MDB	br_port_msg

Netlink Notifications

Kernel generated messages to user space about an event

- multicast groups and listening for notifications
  RTNLGRP_<object> for rtnetlink
- API: sockaddr_nl and nl_groups (32-bit mask)
- Groups > 31 need to use setsockopt
  NETLINK_ADD_MEMBERSHIP, NETLINK_DROP_MEMBERSHIP

Past Mistakes and Impacts

Wrong Payload Header

iproute2 historically sent rtgenmsg (< 3.9) and then ifinfomsg (< 4.20) for most rtnetlink dump requests

- **wrong** header for all message types but RTM_*LINK

Kernel allowed this at the time and now always has to allow it

- Most dump handlers did not check the request message, only referenced address family

Affects the ability to add attributes to dump request or use data in proper header struct

- Dump capability has been around for years
- Want to add option to limit amount of data returned

Wrong Payload Header - Examples

fdb dump

- Initial fdb dump commit did not check header; iproute2 was sending rtgenmsg
- Later commit changes iproute2 to send ifinfomsg; kernel allows
- Commits to both code bases for MASTER attribute in fdb dumps
- Kernel allows ifinfomsg for GETNEIGH + attributes appended
- Mistake was propagated to libnl

Route dump

- Proper struct is rtmsg (12 bytes); ifinfomsg (16 bytes) sent by iproute2 and allowed for years
- Desire to add attribute to dump request (e.g., specific table). Can the kernel uniquely determine rtmsg + attribute and ifinfomsg + attribute?

Strict Checking

A lot of the RTM_GET* commands fixed in 4.20 but requires a UAPI

- strict checking - NETLINK_GET_STRICT_CHK socket option
- iproute2 fixed to send correct headers (4.20) and strict checking (5.0)

Awkward

- userspace **opts in** to the kernel doing the right thing

Too many existing dump handlers to fix all at once

- tc still not done

Strict Checking Going Forward

New code should use strict checking by default

- nlmsg_parse_strict and nla_parse_strict

Existing code should be handling parse failures

- check the return code of the parse functions – they fail for a reason

Wrong Size for attributes

Sending wrong size for attribute

- userspace sends u32 instead of u8 or u16

If an attribute is a u32, length should be exactly 4 bytes, not >= 4

Example

- RTA_GATEWAY for IPv4 allows address to > 4 (e.g., an IPv6 address)
- The result is a route with only the bottom 4 bytes of the v6 address!

NEW/SET Commands

Kernel handlers historically not checked every attribute

- should error out if message contains unknown attribute

New attributes added over time

- new command on old kernel – random networking config
- VXLAN is one example

No way for userspace to probe for support

Going forward as new attributes are added ALL existing handlers should be updated

- implement support or return an error

NEW/SET Commands Example

RTA_VIA attribute for routes

- Added for MPLS – a way to specify a gateway from a different address family
- IPv4 and IPv6 not updated to check for attribute – just ignores it

  wrong route inserted – e.g., device only, no gateway

  $ ip ro add 172.16.1.254/32 via inet6 fe80::68d5:56ff:fe07:2f1b dev eth1

  $ ip ro ls

  …

  172.16.1.254 dev eth1
- FRR can not rely on probing for RTA_VIA to know if kernel supports v6 gateway with v4 route

  New feature coming soon

What Needs to Improve

Hacks^Wworkarounds are needed for existing code

- Ugly UAPI and complicated kernel code

Check every bit processing netlink messages

- A bit not checked is a bit that can not be used in the future
- Allows userspace to probe for feature support
- ***strict parsing functions + nla policy***

More thought and review

- Consistency across network layer address families for example
- At a point that IPv6 multipath routes are forever different from ipv4

Scale

Message batching and ACKs

netlink API supports message batching

- one sendmsg call with a buffer containing N-messages
- ‘tc --batch’ uses this
- FRR working on support

Do not set NLM_F_ACK

- ACK message generated only on error
- Recent change to FRR - ~20% speedup

Reducing Notification Storms

What can be done to reduce the number and size of notifications?

- Userspace socket overflows, the app has to re-sync – can be costly
- Can notifications be batched?

Measurable cost to generating notifications

- Can they be skipped completely (no message generation) if no listeners?

Limiting notifications received by app

- BPF socket filter one option

  filter run on message content

  limited scope – e.g., can not limit by VRF (aka MASTER device)

Notification storms

Link events

- Too many messages and large message sizes

Routes

- IPv4 – does not send notifications for routes affected by link events
- IPv6 – recent patch to opt-out (sysctl); makes v6 similar to v4
- MPLS – needs to be fixed

Neighbors

- All neigh entries evicted on a link event
- Patch to opt-out

Notification Examples

ntp - LINKs, ADDR, ROUTE, MROUTE

- time server cares about all networking config? well, sort of
- does not parse message and only looks at header for message type. updates interface_timer
- Clearly does not need every single message in a NOS environment
  e.g., Management VRF limitation

dhcpd - LINK, ROUTE, ADDR, NEIGH

Jamal has more examples later

Kernel Data

Kernel holds the single source of truth for networking configuration data and userspace queries

- e.g., rtnetlink – RTM_GET*, NLM_F_DUMP (return all)

Do not want to push all data to userspace when user only wants a small subset

- Lot of messages between kernel and user
- Wasted cycles on both sides
- Have to start over if a change happens in the middle
  Problem at scale

Coarse Filtering for GET Request

Set fields in payload header

Append attributes to GET request

- Device index, Master device, route table
- e.g., LINK, NEIGH, ROUTE dumps

Should be limited to higher level, coarse grained filtering

- At some point too many checks on each object affects performance

Thank you!

Visit us at cumulusnetworks.com or follow us @cumulusnetworks