

# Survey of inconsistencies in Linux kernel IPv4/IPv6 UAPI

Roopa Prabhu

# Agenda

- Goals
- Introduction to Kernel Netlink UAPI for IPv4/IPv6
- Introduction to userspace apps relying on the UAPI
- Survey areas of inconsistencies and discuss solutions

# Goals

- Guide to deploy IPv4 and IPv6
- And hope to provide enough motivation to keep the IPv4 and IPv6 UAPI consistent in the future

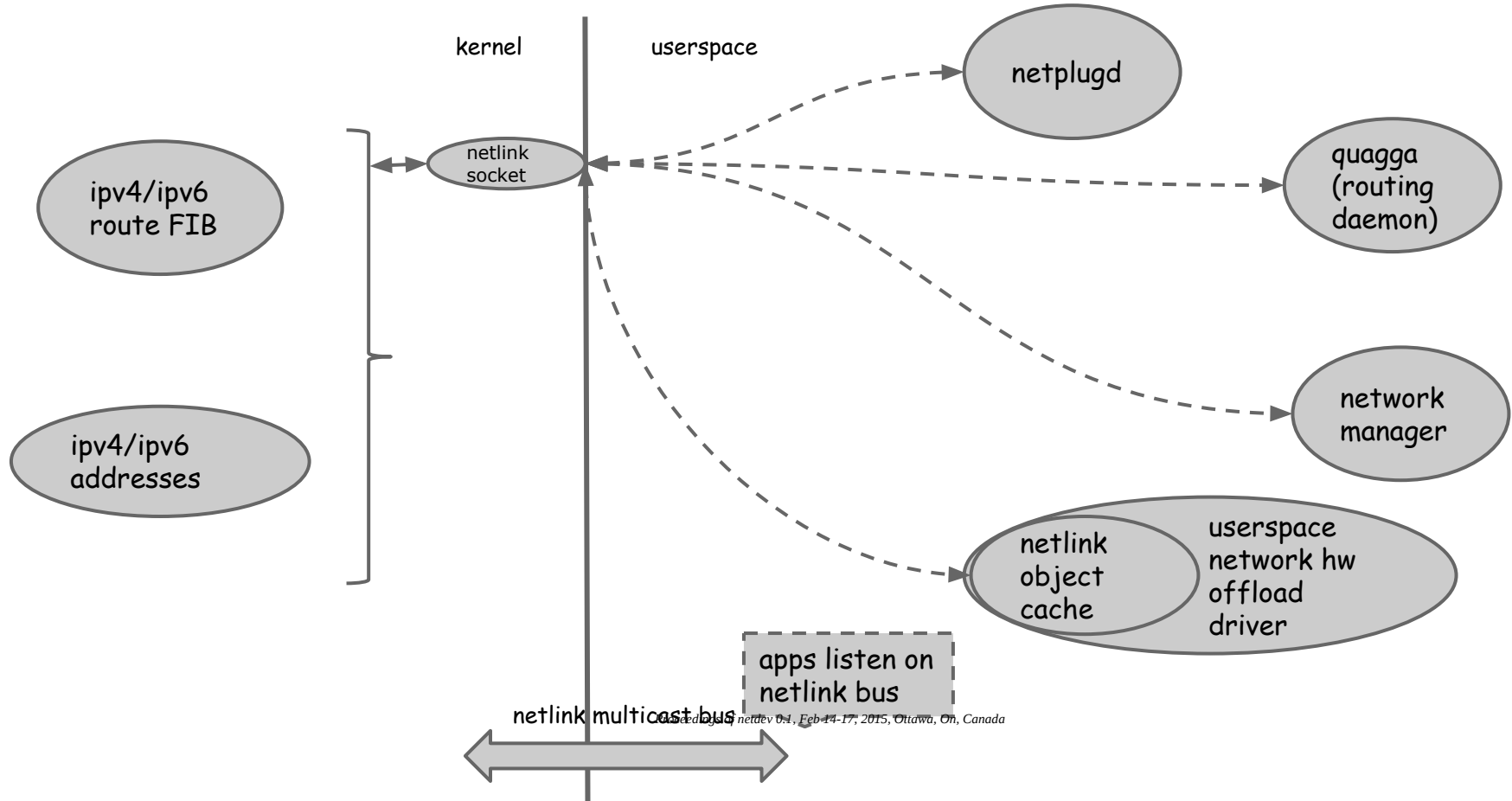
# Introduction

- Kernel provides netlink based UAPI and tools to manage IPv4 and IPv6 (Netlink message types: `RTM_NEWROUTE`, `RTM_DELROUTE` and `RTM_GETROUTE`)
- Kernel notifies user space via Netlink notifications

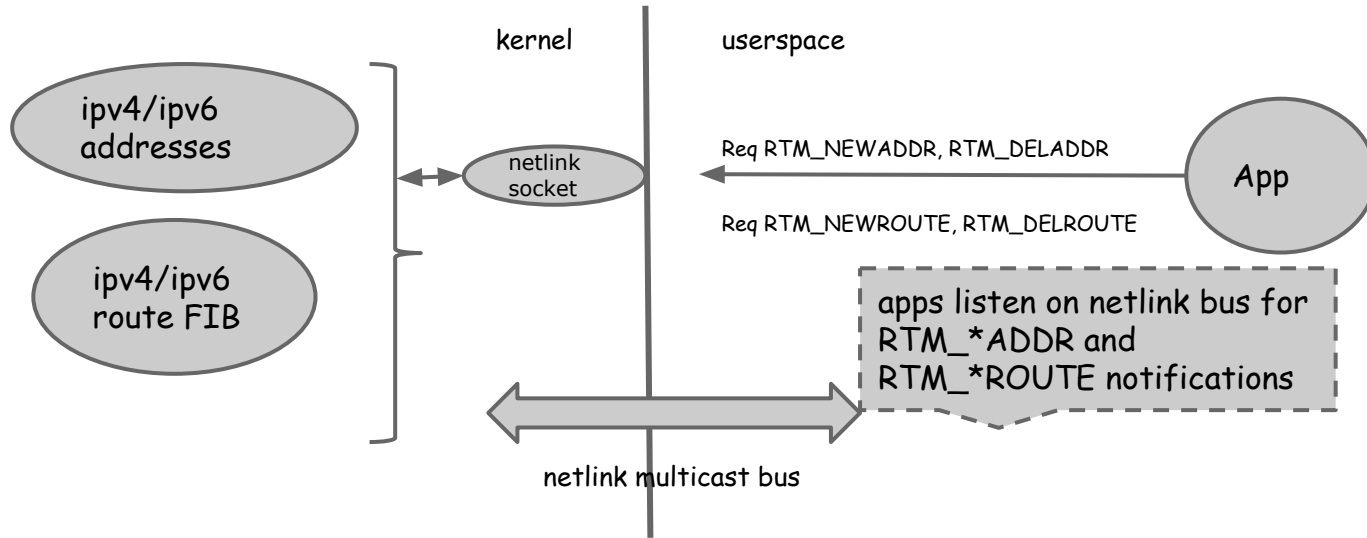
# Example Applications using the API

- Network Managers
- Routing daemons
- Userspace netlink caches
- Userspace hardware offload drivers

# Netlink apps ...



# RTnetlink addr and route messages



# We will discuss the following UAPI's:

- Address handling on interface down
- Route delete notifications on interface down
- Multipath route add/del UAPI
- Multipath route netlink notification
- Multipath route replaces
- Multipath route appends
- Handling un-equal cost multipath routes



# Address handling on interface down

IPv6 global addresses are flushed on ifdown,  
but IPv4 stay

# Example: address handling on ifdown

```
# interface dummy0 below has an ipv4 address, ipv6 global
```

```
# and ipv6 link local address
```

```
ip addr show
```

```
4: dummy0: <BROADCAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc  
noqueue state UNKNOWN group default
```

```
link/ether 12:3f:92:73:f7:1f brd ff:ff:ff:ff:ff:ff
```

```
inet 10.0.13.2/24 scope global dummy0
```

```
valid_lft forever preferred_lft forever
```

```
inet6 2001:20:1::2/64 scope global
```

```
valid_lft forever preferred_lft forever
```

```
inet6 fe80::103f:92ff:fe73:f71f/64 scope link
```

```
valid_lft forever preferred_lft forever
```

```
# down dummy0
```

```
ip link set dev dummy0 down
```

```
ip monitor addr
```

```
Deleted 4: dummy0 inet6 2001:20:1::2/64 scope global
```

```
valid_lft forever preferred_lft forever
```

```
Deleted 4: dummy0 inet6 fe80::103f:92ff:fe73:f71f/64 scope link valid_lft
```

```
forever preferred_lft forever
```

```
# bring interface dummy0 up
```

```
ip link set dev dummy0 up
```

```
# ip monitor output showing ipv6 link local address coming
```

```
# back up
```

```
ip monitor addr
```

```
4: dummy0 inet6 fe80::103f:92ff:fe73:f71f/64 scope link
```

```
valid_lft forever preferred_lft forever
```

```
# ipv6 global scope address 2001:20:1::2/64, never came back # and is lost
```

```
ip addr show
```

```
4: dummy0: <BROADCAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc  
noqueue state UNKNOWN group default
```

```
link/ether 12:3f:92:73:f7:1f brd ff:ff:ff:ff:ff:ff
```

```
inet 10.0.13.2/24 scope global dummy0
```

```
valid_lft forever preferred_lft forever
```

```
inet6 fe80::103f:92ff:fe73:f71f/64 scope link
```

```
valid_lft forever preferred_lft forever
```

# Solutions

**In user-space:** monitor link down messages and re-configure addresses on ifup (netplugd is an option)

**Problems:** This special handling becomes part of multiple applications (problem aggravated with multiple network namespaces: multiple netplugd instances)

**In kernel:** Don't flush IPv6 addresses on Link down  
(Thanks to a recent fix from David Ahern)

# Route delete notifications on interface down

- Kernel notifies user-space of IPv6 dead routes on interface down
- But, user-space is not notified of IPv4 dead routes on interface down

# Example: route delete notifications

```
# interface dummy0 below has an ipv4 address, ipv6 global
# and ipv6 link local address
ip addr show
4: dummy0: <BROADCAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc
noqueue state UNKNOWN group default
link/ether 12:3f:92:73:f7:1f brd ff:ff:ff:ff:ff:ff
inet 10.0.13.2/24 scope global dummy0
valid_lft forever preferred_lft forever
inet6 2001:20:1::2/64 scope global
valid_lft forever preferred_lft forever
inet6 fe80::103f:92ff:fe73:f71f/64 scope link
valid_lft forever preferred_lft forever
# showing IPv4 connected routes installed by the kernel
# for the IPv4 address
ip -4 route show
10.0.13.0/24 dev dummy0 proto kernel scope link src 10.0.13.2
```

```
# showing IPv6 connected routes installed by the kernel
```

```
# for the IPv6 address
```

```
ip -6 route show
```

```
2001:20:1::/64 dev dummy0 proto kernel metric 256
```

```
fe80::/64 dev dummy0 proto kernel metric 256
```

```
# As you can see below, only notifications for IPv6 were
```

```
# generated by the kernel. There were no notifications for
```

```
# IPv4 route delete.
```

```
ip monitor route
```

```
Deleted 2001:20:1::/64 dev dummy0 proto kernel metric 256
```

```
Deleted fe80::/64 dev dummy0 proto kernel metric 256
```

```
Deleted ff00::/8 dev dummy0 table local metric 256
```

```
Deleted local 2001:20:1::2 dev lo table local proto none metric 0
```

```
Deleted local fe80::103f:92ff:fe73:f71f dev lo table local proto none metric 0
```

```
# Both IPv4 and IPv6 connected routes were deleted by
```

```
# the kernel
```

```
ip -4 route show
```

```
ip -6 route show
```

# Solutions

**In user-space:** An application can listen to link notifications and purge all IPv4 dead routes

**Problems:** Handling of route purging gets duplicated in multiple applications

**In kernel:** IPv4 UAPI can be fixed to generate notifications on all dead routes similar to IPv6

**(Note:** Kernel does not generate notifications for dead routes today because user-space can figure this out. Which we believe might be the right thing to do given that this can generate a notification storm on interface down)

# Multipath route add/del api

## IPv4:

```
ip route add 10.0.15.2 \  
    nexthop via 10.0.12.2 dev dummy0 \  
    nexthop via 10.0.13.2 dev dummy1
```

## IPv6: Two ways to add multipath routes

(legacy, currently there for backward compatibility)

```
ip -6 route add 3ffe:304:124:2306::/64 \  
    nexthop via fe80::b077:f0ff:fe23:5cc7 dev dummy0  
ip -6 route add 3ffe:304:124:2306::/64 \  
    nexthop via fe80::d850:e7ff:fe87:cf6a dev dummy1
```

**and**

```
ip -6 route add 3ffe:304:124:2306::/64 \  
    nexthop via fe80::b077:f0ff:fe23:5cc7 dev dummy0 \  
    nexthop via fe80::d850:e7ff:fe87:cf6a dev dummy1
```

# Multipath route notifications

**Ipv4:** Notification contains all nexthop information

ip monitor route

10.0.15.2

nexthop via 10.0.12.2 dev dummy0 weight 1

nexthop via 10.0.13.2 dev dummy1 weight 1

**Ipv6:** One separate notification for each nexthop

ip monitor route

3ffe:304:124:2306::/64 via fe80::b077:f0ff:fe23:5cc7 dev dummy0 metric 1024

3ffe:304:124:2306::/64 via fe80::d850:e7ff:fe87:cf6a dev dummy1 metric 1024



# Solutions

- **In user-space:** Application re-builds a multipath route in userspace from individual notifications
- **Problems:** guessing multipath route in userspace from individual notifications can be error prone
- **In kernel:** IPv6 multipath notification format should be made similar to IPv4

# Multipath route replaces

- Route replace: RTM\_NEWROUTE with NLM\_F\_REPLACE flag
- Unlike IPv4, IPv6 allows replacing a single nexthop in a multipath route

# Route replace example

```
#ipv4
$ip route show
10.0.12.2
    nexthop via 10.0.13.2 dev dummy0 weight 1
    nexthop via 10.0.14.2 dev dummy1 weight 1

$ip route replace 10.0.12.2 nexthop via 10.0.15.2 dev dummy2

$ip monitor route
10.0.12.2 via 10.0.15.2 dev dummy2

$ip route show
10.0.12.2 via 10.0.15.2 dev dummy2

#ipv6
$ ip -6 route show
3ffe:304:124:2306::/64 via fe80::b077:f0ff:fe23:5cc7 dev dummy0
metric 1024
3ffe:304:124:2306::/64 via fe80::d850:e7ff:fe87:cf6a dev dummy1
metric 1024

$ip -6 route replace 3ffe:304:124:2306::/64 nexthop via fe80::c26:
cdf:feca:18f2 dev dummy2

$ip monitor route
3ffe:304:124:2306::/64 via fe80::c26:cdf:feca:18f2 dev dummy2
metric 1024

$ip -6 route show /* replaced a single nexthop of a multipath route */
3ffe:304:124:2306::/64 via fe80::c26:cdf:feca:18f2 dev dummy2
metric 1024
3ffe:304:124:2306::/64 via fe80::d850:e7ff:fe87:cf6a dev dummy1
metric 1024
```

# Solutions

- In user-space: Always replace the first next hop in the list if the notification contained `NLM_F_REPLACE` flag
- Problems: guessing replace sequence in userspace is error prone
- In kernel: IPv6 multipath notification format should be made similar to IPv4 (Additionally, replace notification can contain more info on which route was replaced)

# Multipath route appends

- Route append: `RTM_NEWROUTE` with `NLM_F_APPEND` flag
- Unlike IPv4, IPv6 allows appending a single nexthop to a multipath route

# Example: IPv6 route append

```
#ipv4
ip route show
10.0.12.2
    nexthop via 10.0.13.2 dev dummy0 weight 1
    nexthop via 10.0.14.2 dev dummy1 weight 1

ip route append 10.0.12.2 nexthop via 10.0.15.2 dev dummy2

ip monitor route
10.0.12.2 via 10.0.15.2 dev dummy2

ip route show
10.0.12.2
    nexthop via 10.0.13.2 dev dummy0 weight 1
    nexthop via 10.0.14.2 dev dummy1 weight 1
10.0.12.2 via 10.0.15.2 dev dummy2

#ipv6
ip -6 route show
3ffe:304:124:2306::/64 via fe80::b077:f0ff:fe23:5cc7 dev dummy0
metric 1024
3ffe:304:124:2306::/64 via fe80::d850:e7ff:fe87:cf6a dev dummy1
metric 1024

ip monitor route
3ffe:304:124:2306::/64 via fe80::c26:cdff:feca:18f2 dev dummy2 metric
1024

ip -6 route append 3ffe:304:124:2306::/64 nexthop via fe80::c26:
cdff:feca:18f2 dev dummy2

ip -6 route show
3ffe:304:124:2306::/64 via fe80::b077:f0ff:fe23:5cc7 dev dummy0
metric 1024
3ffe:304:124:2306::/64 via fe80::d850:e7ff:fe87:cf6a dev dummy1
metric 1024
3ffe:304:124:2306::/64 nexthop via fe80::c26:cdff:feca:18f2 dev
dummy2
```

# Solutions

- **In user-space:** Append nexthops learnt from new notification to end of the nexthop list
- **Problems:** guessing append sequence in userspace is error prone
- **In kernel:** IPv6 multipath notification format to be made similar to IPv4

# Unequal cost multipath routes

Two ways to assign weights to nexthops:

1. Repeat nexthop times equal to the weight of the nexthop
2. Use 'weight' attribute to assign weights

IPv4 supports both 1) and 2) today. IPv6 supports only 2)



# Solutions

**In user-space:** weights can be used in the case of both IPv4 and IPv6

**In kernel:** IPv6 can be made consistent with IPv4

(Note: The difference is because of the way IPv6 stores its nexthops in the kernel)

# Conclusions

Keeping the IPv4 and IPv6 kernel API consistent will simplify user space networking apps

# Futures

- Patches to unify IPv4 and IPv6 API (under a `sysctl` if it changes default behaviour)
- Future IPv4 and IPv6 API can keep consistency in mind