
Subject: [PATCH RFC 29/31] net: Make AF_PACKET handle multiple network namespaces

Posted by [ebiederm](#) on Thu, 25 Jan 2007 19:00:31 GMT

[View Forum Message](#) <> [Reply to Message](#)

From: Eric W. Biederman <ebiederm@xmission.com> - unquoted

This is done by making all of the relevant global variables per network namespace.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

net/packet/af_packet.c | 125 ++++++-----

1 files changed, 81 insertions(+), 44 deletions(-)

diff --git a/net/packet/af_packet.c b/net/packet/af_packet.c

index 4ac9f9f..c772491 100644

--- a/net/packet/af_packet.c

+++ b/net/packet/af_packet.c

@@ -152,8 +152,8 @@ dev->hard_header == NULL (ll header is added by device, we cannot control it)

*/

/* List of all packet sockets. */

-static HLIST_HEAD(packet_sklist);

-static DEFINE_RWLOCK(packet_sklist_lock);

+static DEFINE_PER_NET(rwlock_t, packet_sklist_lock);

+static DEFINE_PER_NET(struct hlist_head, packet_sklist);

static atomic_t packet_socks_nr;

@@ -264,9 +264,6 @@ static int packet_rcv_spkt(struct sk_buff *skb, struct packet_type *pt, struct n

struct sock *sk;

struct sockaddr_pkt *spkt;

- if (!net_eq(dev->nd_net, init_net()))

- goto out;

-

/*

* When we registered the protocol we saved the socket in the data

* field for just this event.

@@ -288,6 +285,9 @@ static int packet_rcv_spkt(struct sk_buff *skb, struct packet_type *pt, struct n

if (skb->pkt_type == PACKET_LOOPBACK)

goto out;

+ if (!net_eq(dev->nd_net, sk->sk_net))

```

+ goto out;
+
if ((skb = skb_share_check(skb, GFP_ATOMIC)) == NULL)
    goto oom;

@@ -359,7 +359,7 @@ static int packet_sendmsg_spkt(struct kiocb *iocb, struct socket *sock,
    */

    saddr->spkt_device[13] = 0;
- dev = dev_get_by_name(init_net(), saddr->spkt_device);
+ dev = dev_get_by_name(sk->sk_net, saddr->spkt_device);
    err = -ENODEV;
    if (dev == NULL)
        goto out_unlock;
@@ -475,15 +475,15 @@ static int packet_rcv(struct sk_buff *skb, struct packet_type *pt, struct
net_de
    int skb_len = skb->len;
    unsigned snaplen;

- if (!net_eq(dev->nd_net, init_net()))
- goto drop;
-
if (skb->pkt_type == PACKET_LOOPBACK)
    goto drop;

    sk = pt->af_packet_priv;
    po = pkt_sk(sk);

+ if (!net_eq(dev->nd_net, sk->sk_net))
+ goto drop;
+
    skb->dev = dev;

    if (dev->hard_header) {
@@ -583,15 +583,15 @@ static int tpacket_rcv(struct sk_buff *skb, struct packet_type *pt, struct
net_d
    unsigned short macoff, netoff;
    struct sk_buff *copy_skb = NULL;

- if (!net_eq(dev->nd_net, init_net()))
- goto drop;
-
if (skb->pkt_type == PACKET_LOOPBACK)
    goto drop;

    sk = pt->af_packet_priv;
    po = pkt_sk(sk);

```

```

+ if (!net_eq(dev->nd_net, sk->sk_net))
+ goto drop;
+
+ if (dev->hard_header) {
+ if (sk->sk_type != SOCK_DGRAM)
+ skb_push(skb, skb->data - skb->mac.raw);
@@ -744,7 +744,7 @@ static int packet_sendmsg(struct kiocb *iocb, struct socket *sock,
+ }

- dev = dev_get_by_index(init_net(), ifindex);
+ dev = dev_get_by_index(sk->sk_net, ifindex);
+ err = -ENXIO;
+ if (dev == NULL)
+ goto out_unlock;
@@ -817,15 +817,17 @@ static int packet_release(struct socket *sock)
+ {
+ struct sock *sk = sock->sk;
+ struct packet_sock *po;
+ net_t net;

+ if (!sk)
+ return 0;

+ net = sk->sk_net;
+ po = pkt_sk(sk);

- write_lock_bh(&packet_sklist_lock);
+ write_lock_bh(&per_net(packet_sklist_lock, net));
+ sk_del_node_init(sk);
- write_unlock_bh(&packet_sklist_lock);
+ write_unlock_bh(&per_net(packet_sklist_lock, net));

+ /*
+  * Unhook packet receive handler.
@@ -943,7 +945,7 @@ static int packet_bind_spkt(struct socket *sock, struct sockaddr *uaddr,
+ int add
+ return -EINVAL;
+ strlcpy(name, uaddr->sa_data, sizeof(name));

- dev = dev_get_by_name(init_net(), name);
+ dev = dev_get_by_name(sk->sk_net, name);
+ if (dev) {
+ err = packet_do_bind(sk, dev, pkt_sk(sk)->num);
+ dev_put(dev);
@@ -971,7 +973,7 @@ static int packet_bind(struct socket *sock, struct sockaddr *uaddr, int
+ addr_len

```

```

if (sll->sll_ifindex) {
    err = -ENODEV;
- dev = dev_get_by_index(init_net(), sll->sll_ifindex);
+ dev = dev_get_by_index(sk->sk_net, sll->sll_ifindex);
    if (dev == NULL)
        goto out;
}
@@ -1000,9 +1002,6 @@ static int packet_create(net_t net, struct socket *sock, int protocol)
    __be16 proto = (__force __be16)protocol; /* weird, but documented */
    int err;

- if (!net_eq(net, init_net()))
- return -EAFNOSUPPORT;
-
    if (!capable(CAP_NET_RAW))
        return -EPERM;
    if (sock->type != SOCK_DGRAM && sock->type != SOCK_RAW)
@@ -1052,9 +1051,9 @@ static int packet_create(net_t net, struct socket *sock, int protocol)
    po->running = 1;
}

- write_lock_bh(&packet_sklist_lock);
- sk_add_node(sk, &packet_sklist);
- write_unlock_bh(&packet_sklist_lock);
+ write_lock_bh(&per_net(packet_sklist_lock, net));
+ sk_add_node(sk, &per_net(packet_sklist, net));
+ write_unlock_bh(&per_net(packet_sklist_lock, net));
    return(0);
out:
    return err;
@@ -1158,7 +1157,7 @@ static int packet_getname_spkt(struct socket *sock, struct sockaddr
*uaddr,
    return -EOPNOTSUPP;

    uaddr->sa_family = AF_PACKET;
- dev = dev_get_by_index(init_net(), pkt_sk(sk)->ifindex);
+ dev = dev_get_by_index(sk->sk_net, pkt_sk(sk)->ifindex);
    if (dev) {
        strlcpy(uaddr->sa_data, dev->name, 15);
        dev_put(dev);
@@ -1184,7 +1183,7 @@ static int packet_getname(struct socket *sock, struct sockaddr *uaddr,
    sll->sll_family = AF_PACKET;
    sll->sll_ifindex = po->ifindex;
    sll->sll_protocol = po->num;
- dev = dev_get_by_index(init_net(), po->ifindex);
+ dev = dev_get_by_index(sk->sk_net, po->ifindex);
    if (dev) {
        sll->sll_hatype = dev->type;

```

```

sll->sll_halen = dev->addr_len;
@@ -1237,7 +1236,7 @@ static int packet_mc_add(struct sock *sk, struct packet_mreq_max
*mreq)
    rtnl_lock();

    err = -ENODEV;
- dev = __dev_get_by_index(init_net(), mreq->mr_ifindex);
+ dev = __dev_get_by_index(sk->sk_net, mreq->mr_ifindex);
    if (!dev)
        goto done;

@@ -1291,7 +1290,7 @@ static int packet_mc_drop(struct sock *sk, struct packet_mreq_max
*mreq)
    if (--ml->count == 0) {
        struct net_device *dev;
        *mlp = ml->next;
- dev = dev_get_by_index(init_net(), ml->ifindex);
+ dev = dev_get_by_index(sk->sk_net, ml->ifindex);
        if (dev) {
            packet_dev_mc(dev, ml, -1);
            dev_put(dev);
@@ -1319,7 +1318,7 @@ static void packet_flush_mclist(struct sock *sk)
    struct net_device *dev;

    po->mclist = ml->next;
- if ((dev = dev_get_by_index(init_net(), ml->ifindex)) != NULL) {
+ if ((dev = dev_get_by_index(sk->sk_net, ml->ifindex)) != NULL) {
    packet_dev_mc(dev, ml, -1);
    dev_put(dev);
}

@@ -1438,12 +1437,10 @@ static int packet_notifier(struct notifier_block *this, unsigned long
msg, void
    struct sock *sk;
    struct hlist_node *node;
    struct net_device *dev = (struct net_device*)data;
+ net_t net = dev->nd_net;

- if (!net_eq(dev->nd_net, init_net()))
- return NOTIFY_DONE;
-
- read_lock(&packet_sklist_lock);
- sk_for_each(sk, node, &packet_sklist) {
+ read_lock(&per_net(packet_sklist_lock, net));
+ sk_for_each(sk, node, &per_net(packet_sklist, net)) {
    struct packet_sock *po = pkt_sk(sk);

    switch (msg) {
@@ -1483,7 +1480,7 @@ static int packet_notifier(struct notifier_block *this, unsigned long msg,

```

```

void
    break;
}
}
- read_unlock(&packet_sklist_lock);
+ read_unlock(&per_net(packet_sklist_lock, net));
    return NOTIFY_DONE;
}

@@ -1851,12 +1848,12 @@ static struct notifier_block packet_netdev_notifier = {
};

#ifdef CONFIG_PROC_FS
-static inline struct sock *packet_seq_idx(loff_t off)
+static inline struct sock *packet_seq_idx(net_t net, loff_t off)
{
    struct sock *s;
    struct hlist_node *node;

- sk_for_each(s, node, &packet_sklist) {
+ sk_for_each(s, node, &per_net(packet_sklist, net)) {
    if (!loff--)
        return s;
    }
@@ -1865,21 +1862,24 @@ static inline struct sock *packet_seq_idx(loff_t off)

static void *packet_seq_start(struct seq_file *seq, loff_t *pos)
{
- read_lock(&packet_sklist_lock);
- return *pos ? packet_seq_idx(*pos - 1) : SEQ_START_TOKEN;
+ net_t net = net_from_voidp(seq->private);
+ read_lock(&per_net(packet_sklist_lock, net));
+ return *pos ? packet_seq_idx(net, *pos - 1) : SEQ_START_TOKEN;
}

static void *packet_seq_next(struct seq_file *seq, void *v, loff_t *pos)
{
+ net_t net = net_from_voidp(seq->private);
    ++*pos;
    return (v == SEQ_START_TOKEN)
- ? sk_head(&packet_sklist)
+ ? sk_head(&per_net(packet_sklist, net))
    : sk_next((struct sock*)v);
}

static void packet_seq_stop(struct seq_file *seq, void *v)
{
- read_unlock(&packet_sklist_lock);

```

```

+ net_t net = net_from_voidp(seq->private);
+ read_unlock(&per_net(packet_sklist_lock, net));
}

static int packet_seq_show(struct seq_file *seq, void *v)
@@ -1915,7 +1915,22 @@ static struct seq_operations packet_seq_ops = {

static int packet_seq_open(struct inode *inode, struct file *file)
{
- return seq_open(file, &packet_seq_ops);
+ struct seq_file *seq;
+ int res;
+ res = seq_open(file, &packet_seq_ops);
+ if (!res) {
+ seq = file->private_data;
+ seq->private = net_to_voidp(get_net(PROC_NET(inode)));
+ }
+ return res;
+}
+
+static int packet_seq_release(struct inode *inode, struct file *file)
+{
+ struct seq_file *seq= file->private_data;
+ net_t net = net_from_voidp(seq->private);
+ put_net(net);
+ return seq_release(inode, file);
}

static struct file_operations packet_seq_fops = {
@@ -1923,15 +1938,37 @@ static struct file_operations packet_seq_fops = {
    .open = packet_seq_open,
    .read = seq_read,
    .llseek = seq_lseek,
- .release = seq_release,
+ .release = packet_seq_release,
};

#endif

+static int packet_net_init(net_t net)
+{
+ rwlock_init(&per_net(packet_sklist_lock, net));
+ INIT_HLIST_HEAD(&per_net(packet_sklist, net));
+
+ if (!proc_net_fops_create(net, "packet", 0, &packet_seq_fops))
+ return -ENOMEM;
+
+ return 0;

```

```

+}
+
+static void packet_net_exit(net_t net)
+{
+ proc_net_remove(net, "packet");
+}
+
+static struct pernet_operations packet_net_ops = {
+ .init = packet_net_init,
+ .exit = packet_net_exit,
+};
+
+static void __exit packet_exit(void)
+{
- proc_net_remove(init_net(), "packet");
  unregister_netdevice_notifier(&packet_netdev_notifier);
+ unregister_pernet_subsys(&packet_net_ops);
  sock_unregister(PF_PACKET);
  proto_unregister(&packet_proto);
}
@@ -1944,8 +1981,8 @@ static int __init packet_init(void)
  goto out;

  sock_register(&packet_family_ops);
+ register_pernet_subsys(&packet_net_ops);
  register_netdevice_notifier(&packet_netdev_notifier);
- proc_net_fops_create(init_net(), "packet", 0, &packet_seq_fops);
out:
  return rc;
}
--
1.4.4.1.g278f

```

Containers mailing list
Containers@lists.osdl.org
<https://lists.osdl.org/mailman/listinfo/containers>
