Subject: [PATCH v3] tun: don't hold network namespace by tun sockets
Posted by Stanislav Kinsbursky on Mon, 12 Mar 2012 12:59:41 GMT
View Forum Message <> Reply to Message

v3: added previously removed sock_put() to the tun_release() callback, because sk_release_kernel() doesn't drop the socket reference.

v2: sk_release_kernel() used for socket release. Dummy tun_release() is required for sk_release_kernel() ---> sock_release() ---> sock->ops->release() call.

TUN was designed to destroy it's socket on network namesapce shutdown. But this will never happen for persistent device, because it's socket holds network namespace.
This patch removes of holding network namespace by TUN socket and replaces it by creating socket in init_net and then changing it's net it to desired one. On shutdown socket is moved back to init_net prior to final put.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>

---
 drivers/net/tun.c | 15 ++++++++++++---
 1 files changed, 12 insertions(+), 3 deletions(-)

```
diff --git a/drivers/net/tun.c b/drivers/net/tun.c
index 2c5d349..74d7f76 100644
--- a/drivers/net/tun.c
+++ b/drivers/net/tun.c
@@ -359,7 +359,7 @@ static void tun_free_netdev(struct net_device *dev)
 {
  struct tun_struct *tun = netdev_priv(dev);

- sock_put(tun->socket.sk);
+ sk_release_kernel(tun->socket.sk);
 }

 /* Net device open. */
@@ -980,10 +980,18 @@ static int tun_recvmsg(struct kiocb *iocb, struct socket *sock,
  return ret;
 }

+static int tun_release(struct socket *sock)
+{
+ if (sock->sk)
+  sock_put(sock->sk);
+ return 0;
+}
+
```

```
 /* Ops structure to mimic raw sockets with tun */
 static const struct proto_ops tun_socket_ops = {
  .sendmsg = tun_sendmsg,
  .recvmsg = tun_recvmsg,
+ .release = tun_release,
 };

 static struct proto tun_proto = {
@@ -1110,10 +1118,11 @@ static int tun_set_iff(struct net *net, struct file *file, struct ifreq *ifr)
   tun->vnet_hdr_sz = sizeof(struct virtio_net_hdr);

   err = -ENOMEM;
-  sk = sk_alloc(net, AF_UNSPEC, GFP_KERNEL, &tun_proto);
+  sk = sk_alloc(&init_net, AF_UNSPEC, GFP_KERNEL, &tun_proto);
   if (!sk)
    goto err_free_dev;

+  sk_change_net(sk, net);
   tun->socket.wq = &tun->wq;
   init_waitqueue_head(&tun->wq.wait);
   tun->socket.ops = &tun_socket_ops;
@@ -1174,7 +1183,7 @@ static int tun_set_iff(struct net *net, struct file *file, struct ifreq *ifr)
  return 0;

  err_free_sk:
- sock_put(sk);
+ tun_free_netdev(dev);
  err_free_dev:
  free_netdev(dev);
  failed:
```