

---

Subject: [PATCH 8/12 net-2.6.26] Make netlink\_kernel\_release publically available as sk\_release\_kernel.

Posted by [den](#) on Fri, 29 Feb 2008 13:40:54 GMT

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

---

This staff will be needed for non-netlink kernel sockets, which should also not pin a namespace like tcp\_socket and icmp\_socket.

Signed-off-by: Denis V. Lunev <[den@openvz.org](mailto:den@openvz.org)>

Acked-by: Daniel Lezcano <[dlezcana@fr.ibm.com](mailto:dlezcana@fr.ibm.com)>

---

```
include/net/sock.h      | 13 ++++++++  
net/core/sock.c        | 18 +++++++  
net/netlink/af_netlink.c| 18 ++-----  
3 files changed, 33 insertions(+), 16 deletions(-)
```

```
diff --git a/include/net/sock.h b/include/net/sock.h  
index fd98760..39112e7 100644  
--- a/include/net/sock.h  
+++ b/include/net/sock.h  
@@ -850,6 +850,7 @@ extern struct sock *sk_alloc(struct net *net, int family,  
     gfp_t priority,  
     struct proto *prot);  
extern void sk_free(struct sock *sk);  
+extern void sk_release_kernel(struct sock *sk);  
extern struct sock *sk_clone(const struct sock *sk,  
    const gfp_t priority);  
  
@@ -1333,6 +1334,18 @@ static inline void sk_eat_skb(struct sock *sk, struct sk_buff *skb, int  
copied_e  
}  
#endif  
  
+/*  
+ * Kernel sockets, f.e. rtnl or icmp_socket, are a part of a namespace.  
+ * They should not hold a reference to a namespace in order to allow  
+ * to stop it.  
+ * Sockets after sk_change_net should be released using sk_release_kernel  
+ */  
+static inline void sk_change_net(struct sock *sk, struct net *net)  
{  
+ put_net(sk->sk_net);  
+ sk->sk_net = net;  
+}  
+  
extern void sock_enable_timestamp(struct sock *sk);  
extern int sock_get_timestamp(struct sock *, struct timeval __user *);  
extern int sock_get_timestampns(struct sock *, struct timespec __user *);
```

```

diff --git a/net/core/sock.c b/net/core/sock.c
index 09cb3a7..c71b645 100644
--- a/net/core/sock.c
+++ b/net/core/sock.c
@@ -987,6 +987,24 @@ void sk_free(struct sock *sk)
    sk_prot_free(sk->sk_prot_creator, sk);
}

+/*
+ * Last sock_put should drop reference to sk->sk_net. It has already
+ * been dropped in sk_change_net. Taking reference to stopping namespace
+ * is not an option.
+ * Take reference to a socket to remove it from hash _alive_ and after that
+ * destroy it in the context of init_net.
+ */
+void sk_release_kernel(struct sock *sk)
+{
+ if (sk == NULL || sk->sk_socket == NULL)
+ return;
+
+ sock_hold(sk);
+ sock_release(sk->sk_socket);
+ sk->sk_net = get_net(&init_net);
+ sock_put(sk);
+}
+
struct sock *sk_clone(const struct sock *sk, const gfp_t priority)
{
    struct sock *newsk;
diff --git a/net/netlink/af_netlink.c b/net/netlink/af_netlink.c
index e6b636d..524e826 100644
--- a/net/netlink/af_netlink.c
+++ b/net/netlink/af_netlink.c
@@ -1372,8 +1372,7 @@ netlink_kernel_create(struct net *net, int unit, unsigned int groups,
    goto out_sock_release_nosk;

    sk = sock->sk;
- put_net(sk->sk_net);
- sk->sk_net = net;
+ sk_change_net(sk, net);

    if (groups < 32)
        groups = 32;
@@ -1421,20 +1420,7 @@ EXPORT_SYMBOL(netlink_kernel_create);
void
netlink_kernel_release(struct sock *sk)
{
- /*

```

```
- * Last sock_put should drop reference to sk->sk_net. It has already
- * been dropped in netlink_kernel_create. Taking reference to stopping
- * namespace is not an option.
- * Take reference to a socket to remove it from netlink lookup table
- * _alive_ and after that destroy it in the context of init_net.
- */
- if (sk == NULL || sk->sk_socket == NULL)
- return;
-
- sock_hold(sk);
- sock_release(sk->sk_socket);
- sk->sk_net = get_net(&init_net);
- sock_put(sk);
+ sk_release_kernel(sk);
}
EXPORT_SYMBOL(netlink_kernel_release);
```

--  
1.5.3.rc5

---

Containers mailing list  
Containers@lists.linux-foundation.org  
<https://lists.linux-foundation.org/mailman/listinfo/containers>

---