Subject: [PATCH 0/13 net-2.6.26] UDP/ICMP/TCP for a namespace v2
Posted by den on Mon, 31 Mar 2008 13:16:07 GMT
View Forum Message <> Reply to Message

Hello, Dave!

This patch set finally enables TCP/UDP and ICMP inside a namespace.
In order to do this we fix ARP processing, IP options processing and
allow packets to flow to namespace aware protocols.

Finally, this makes namespace functional and alive :)

Changed from v1:
- inet_csk_ctl_sock_create is renamed to inet_ctl_sock_create. Thanks
  to Arnaldo Carvalho de Melo <acme@redhat.com and Vlad Yasevich
  <vladislav.yasevich@hp.com>
- dccp_ctl_make_reset now accept sock rather than socket. Thanks to
  Arnaldo Carvalho de Melo <acme@redhat.com>
- added inet_ctl_sock_destroy

Signed-off-by: Denis V. Lunev <den@openvz.org>

Subject: [PATCH 1/13 net-2.6.26] [TCP]: Replace socket with sock for reset sending.
Posted by den on Mon, 31 Mar 2008 13:16:41 GMT
View Forum Message <> Reply to Message

Replace tcp_socket with tcp_sock. This is more effective (less derefferences
on fast paths). Additionally, the approach is unified to one used in ICMP.

Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 net/ipv4/tcp_ipv4.c | 10 ++++++----
 1 files changed, 6 insertions(+), 4 deletions(-)

diff --git a/net/ipv4/tcp_ipv4.c b/net/ipv4/tcp_ipv4.c
index ef141b8..1d77f37 100644
--- a/net/ipv4/tcp_ipv4.c
+++ b/net/ipv4/tcp_ipv4.c
@@ -89,7 +89,7 @@ int sysctl_tcp_low_latency __read_mostly;
 #define ICMP_MIN_LENGTH 8

 /* Socket used for sending RSTs */
-static struct socket *tcp_socket __read_mostly;
+static struct sock *tcp_sock __read_mostly;

 void tcp_v4_send_check(struct sock *sk, int len, struct sk_buff *skb);

```
@@ -598,7 +598,7 @@ static void tcp_v4_send_reset(struct sock *sk, struct sk_buff *skb)
          sizeof(struct tcphdr), IPPROTO_TCP, 0);
  arg.csumoffset = offsetof(struct tcphdr, check) / 2;

- ip_send_reply(tcp_socket->sk, skb, &arg, arg.iov[0].iov_len);
+ ip_send_reply(tcp_sock, skb, &arg, arg.iov[0].iov_len);

  TCP_INC_STATS_BH(TCP_MIB_OUTSEGS);
  TCP_INC_STATS_BH(TCP_MIB_OUTRSTS);
@@ -693,7 +693,7 @@ static void tcp_v4_send_ack(struct tcp_timewait_sock *twsk,
  if (twsk)
   arg.bound_dev_if = twsk->tw_sk.tw_bound_dev_if;

- ip_send_reply(tcp_socket->sk, skb, &arg, arg.iov[0].iov_len);
+ ip_send_reply(tcp_sock, skb, &arg, arg.iov[0].iov_len);

  TCP_INC_STATS_BH(TCP_MIB_OUTSEGS);
 }
@@ -2490,9 +2490,11 @@ struct proto tcp_prot = {

 void __init tcp_v4_init(void)
 {
- if (inet_csk_ctl_sock_create(&tcp_socket, PF_INET, SOCK_RAW,
+ struct socket *__tcp_socket;
+ if (inet_csk_ctl_sock_create(&__tcp_socket, PF_INET, SOCK_RAW,
          IPPROTO_TCP) < 0)
   panic("Failed to create the TCP control socket.\n");
+ tcp_sock = __tcp_socket->sk;
 }

 EXPORT_SYMBOL(ipv4_specific);
--
1.5.3.rc5
```

Subject: [PATCH 2/13 net-2.6.26] [DCCP]: Replace socket with sock for reset sending.

Posted by den on Mon, 31 Mar 2008 13:16:42 GMT

View Forum Message <> Reply to Message

Replace dccp_v(4|6)_ctl_socket with sock to unify a code with TCP/ICMP.

```
Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 net/dccp/dccp.h   |    2 +-
 net/dccp/ipv4.c   |   16 ++++++++++-------
 net/dccp/ipv6.c   |   10 ++++++----
 net/dccp/output.c |    6 +++---
 4 files changed, 19 insertions(+), 15 deletions(-)

diff --git a/net/dccp/dccp.h b/net/dccp/dccp.h
index e1b7c9c..fe7726b 100644
--- a/net/dccp/dccp.h
+++ b/net/dccp/dccp.h
@@ -296,7 +296,7 @@ extern unsigned int dccp_poll(struct file *file, struct socket *sock,
 extern int    dccp_v4_connect(struct sock *sk, struct sockaddr *uaddr,
        int addr_len);

-extern struct sk_buff *dccp_ctl_make_reset(struct socket *ctl,
+extern struct sk_buff *dccp_ctl_make_reset(struct sock *sk,
        struct sk_buff *skb);
 extern int    dccp_send_reset(struct sock *sk, enum dccp_reset_codes code);
 extern void    dccp_send_close(struct sock *sk, const int active);
diff --git a/net/dccp/ipv4.c b/net/dccp/ipv4.c
index 7d62f7e..f97049b 100644
--- a/net/dccp/ipv4.c
+++ b/net/dccp/ipv4.c
@@ -36,7 +36,7 @@
  * the Out-of-the-blue (OOTB) packets. A control sock will be created
  * for this socket at the initialization time.
  */
-static struct socket *dccp_v4_ctl_socket;
+static struct sock *dccp_v4_ctl_sk;

 int dccp_v4_connect(struct sock *sk, struct sockaddr *uaddr, int addr_len)
 {
@@ -514,11 +514,11 @@ static void dccp_v4_ctl_send_reset(struct sock *sk, struct sk_buff
*rxskb)
  if (rxskb->rtable->rt_type != RTN_LOCAL)
   return;

- dst = dccp_v4_route_skb(dccp_v4_ctl_socket->sk, rxskb);
+ dst = dccp_v4_route_skb(dccp_v4_ctl_sk, rxskb);
  if (dst == NULL)
   return;

- skb = dccp_ctl_make_reset(dccp_v4_ctl_socket, rxskb);
+ skb = dccp_ctl_make_reset(dccp_v4_ctl_sk, rxskb);
  if (skb == NULL)
   goto out;
```

```
@@ -527,10 +527,10 @@ static void dccp_v4_ctl_send_reset(struct sock *sk, struct sk_buff
*rxskb)
         rxiph->daddr);
  skb->dst = dst_clone(dst);

- bh_lock_sock(dccp_v4_ctl_socket->sk);
- err = ip_build_and_send_pkt(skb, dccp_v4_ctl_socket->sk,
+ bh_lock_sock(dccp_v4_ctl_sk);
+ err = ip_build_and_send_pkt(skb, dccp_v4_ctl_sk,
         rxiph->daddr, rxiph->saddr, NULL);
- bh_unlock_sock(dccp_v4_ctl_socket->sk);
+ bh_unlock_sock(dccp_v4_ctl_sk);

  if (net_xmit_eval(err) == 0) {
   DCCP_INC_STATS_BH(DCCP_MIB_OUTSEGS);
@@ -991,6 +991,7 @@ static struct inet_protosw dccp_v4_protosw = {

 static int __init dccp_v4_init(void)
 {
+ struct socket *socket;
  int err = proto_register(&dccp_v4_prot, 1);

  if (err != 0)
@@ -1002,10 +1003,11 @@ static int __init dccp_v4_init(void)

  inet_register_protosw(&dccp_v4_protosw);

- err = inet_csk_ctl_sock_create(&dccp_v4_ctl_socket, PF_INET,
+ err = inet_csk_ctl_sock_create(&socket, PF_INET,
           SOCK_DCCP, IPPROTO_DCCP);
  if (err)
   goto out_unregister_protosw;
+ dccp_v4_ctl_sk = socket->sk;
 out:
  return err;
 out_unregister_protosw:
diff --git a/net/dccp/ipv6.c b/net/dccp/ipv6.c
index ea3f326..44e8b33 100644
--- a/net/dccp/ipv6.c
+++ b/net/dccp/ipv6.c
@@ -34,7 +34,7 @@
 #include "feat.h"

 /* Socket used for sending RSTs and ACKs */
-static struct socket *dccp_v6_ctl_socket;
+static struct sock *dccp_v6_ctl_sk;
```

```
 static struct inet_connection_sock_af_ops dccp_ipv6_mapped;
 static struct inet_connection_sock_af_ops dccp_ipv6_af_ops;
@@ -303,7 +303,7 @@ static void dccp_v6_ctl_send_reset(struct sock *sk, struct sk_buff *rxskb)
  if (!ipv6_unicast_destination(rxskb))
   return;

- skb = dccp_ctl_make_reset(dccp_v6_ctl_socket, rxskb);
+ skb = dccp_ctl_make_reset(dccp_v6_ctl_sk, rxskb);
  if (skb == NULL)
   return;

@@ -324,7 +324,7 @@ static void dccp_v6_ctl_send_reset(struct sock *sk, struct sk_buff *rxskb)
  /* sk = NULL, but it is safe for now. RST socket required. */
  if (!ip6_dst_lookup(NULL, &skb->dst, &fl)) {
   if (xfrm_lookup(&skb->dst, &fl, NULL, 0) >= 0) {
-   ip6_xmit(dccp_v6_ctl_socket->sk, skb, &fl, NULL, 0);
+   ip6_xmit(dccp_v6_ctl_sk, skb, &fl, NULL, 0);
    DCCP_INC_STATS_BH(DCCP_MIB_OUTSEGS);
    DCCP_INC_STATS_BH(DCCP_MIB_OUTRSTS);
    return;
@@ -1173,6 +1173,7 @@ static struct inet_protosw dccp_v6_protosw = {

 static int __init dccp_v6_init(void)
 {
+ struct socket *socket;
  int err = proto_register(&dccp_v6_prot, 1);

  if (err != 0)
@@ -1184,10 +1185,11 @@ static int __init dccp_v6_init(void)

  inet6_register_protosw(&dccp_v6_protosw);

- err = inet_csk_ctl_sock_create(&dccp_v6_ctl_socket, PF_INET6,
+ err = inet_csk_ctl_sock_create(&socket, PF_INET6,
            SOCK_DCCP, IPPROTO_DCCP);
  if (err != 0)
   goto out_unregister_protosw;
+ dccp_v6_ctl_sk = socket->sk;
 out:
  return err;
 out_unregister_protosw:
diff --git a/net/dccp/output.c b/net/dccp/output.c
index 3b763db..f32a84e 100644
--- a/net/dccp/output.c
+++ b/net/dccp/output.c
@@ -348,7 +348,7 @@ struct sk_buff *dccp_make_response(struct sock *sk, struct dst_entry
*dst,
 EXPORT_SYMBOL_GPL(dccp_make_response);
```

```
 /* answer offending packet in @rcv_skb with Reset from control socket @ctl */
-struct sk_buff *dccp_ctl_make_reset(struct socket *ctl, struct sk_buff *rcv_skb)
+struct sk_buff *dccp_ctl_make_reset(struct sock *sk, struct sk_buff *rcv_skb)
 {
  struct dccp_hdr *rxdh = dccp_hdr(rcv_skb), *dh;
  struct dccp_skb_cb *dcb = DCCP_SKB_CB(rcv_skb);
@@ -358,11 +358,11 @@ struct sk_buff *dccp_ctl_make_reset(struct socket *ctl, struct sk_buff
*rcv_skb)
  struct dccp_hdr_reset *dhr;
  struct sk_buff *skb;

- skb = alloc_skb(ctl->sk->sk_prot->max_header, GFP_ATOMIC);
+ skb = alloc_skb(sk->sk_prot->max_header, GFP_ATOMIC);
  if (skb == NULL)
   return NULL;

- skb_reserve(skb, ctl->sk->sk_prot->max_header);
+ skb_reserve(skb, sk->sk_prot->max_header);

  /* Swap the send and the receive. */
  dh = dccp_zeroed_hdr(skb, dccp_hdr_reset_len);
--
1.5.3.rc5
```


Subject: [PATCH 3/13 net-2.6.26] [DCCP]: dccp_v(4|6)_ctl_socket is leaked.
Posted by den on Mon, 31 Mar 2008 13:16:43 GMT
View Forum Message <> Reply to Message

This seems a purism as module can't be unloaded, but though if cleanup method
is present it should be correct and clean all staff created.

Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 net/dccp/ipv4.c | 1 +
 net/dccp/ipv6.c | 1 +
 2 files changed, 2 insertions(+), 0 deletions(-)

diff --git a/net/dccp/ipv4.c b/net/dccp/ipv4.c
index f97049b..6d8f684 100644
--- a/net/dccp/ipv4.c
+++ b/net/dccp/ipv4.c

```
@@ -1020,6 +1020,7 @@ out_proto_unregister:

 static void __exit dccp_v4_exit(void)
 {
+ sock_release(dccp_v4_ctl_sk->sk_socket);
  inet_unregister_protosw(&dccp_v4_protosw);
  inet_del_protocol(&dccp_v4_protocol, IPPROTO_DCCP);
  proto_unregister(&dccp_v4_prot);
diff --git a/net/dccp/ipv6.c b/net/dccp/ipv6.c
index 44e8b33..c5d9d1b 100644
--- a/net/dccp/ipv6.c
+++ b/net/dccp/ipv6.c
@@ -1202,6 +1202,7 @@ out_unregister_proto:

 static void __exit dccp_v6_exit(void)
 {
+ sock_release(dccp_v6_ctl_sk->sk_socket);
  inet6_del_protocol(&dccp_v6_protocol, IPPROTO_DCCP);
  inet6_unregister_protosw(&dccp_v6_protosw);
  proto_unregister(&dccp_v6_prot);
--
1.5.3.rc5
```

Subject: [PATCH 4/13 net-2.6.26] [INET]: Rename inet_csk_ctl_sock_create to inet_ctl_sock_create.

Posted by den on Mon, 31 Mar 2008 13:16:44 GMT

View Forum Message <> Reply to Message

This call is nothing common with INET connection sockets code. It simply
creates an unhashes kernel sockets for protocol messages.

Move the new call into af_inet.c after the rename.

Signed-off-by: Denis V. Lunev <den@openvz.org>

```
---
 include/net/inet_common.h         |    5 +++++
 include/net/inet_connection_sock.h |    5 -----
 net/dccp/ipv4.c                   |    4 ++--
 net/dccp/ipv6.c                   |    4 ++--
 net/ipv4/af_inet.c                |   19 +++++++++++++++++++
 net/ipv4/inet_connection_sock.c    |   19 -------------------
 net/ipv4/tcp_ipv4.c               |    4 ++--
```

```
 net/ipv6/tcp_ipv6.c               |    3 ++-
 8 files changed, 32 insertions(+), 31 deletions(-)

diff --git a/include/net/inet_common.h b/include/net/inet_common.h
index 38d5a1e..d6238bd 100644
--- a/include/net/inet_common.h
+++ b/include/net/inet_common.h
@@ -39,6 +39,11 @@ extern int   inet_getname(struct socket *sock,
 extern int   inet_ioctl(struct socket *sock,
         unsigned int cmd, unsigned long arg);

+extern int   inet_ctl_sock_create(struct socket **sock,
+           unsigned short family,
+           unsigned short type,
+           unsigned char protocol);
+
 #endif


diff --git a/include/net/inet_connection_sock.h b/include/net/inet_connection_sock.h
index f00f057..2ff545a 100644
--- a/include/net/inet_connection_sock.h
+++ b/include/net/inet_connection_sock.h
@@ -327,11 +327,6 @@ extern void inet_csk_listen_stop(struct sock *sk);

 extern void inet_csk_addr2sockaddr(struct sock *sk, struct sockaddr *uaddr);

-extern int inet_csk_ctl_sock_create(struct socket **sock,
-        unsigned short family,
-        unsigned short type,
-        unsigned char protocol);
-
 extern int inet_csk_compat_getsockopt(struct sock *sk, int level, int optname,
           char __user *optval, int __user *optlen);
 extern int inet_csk_compat_setsockopt(struct sock *sk, int level, int optname,
diff --git a/net/dccp/ipv4.c b/net/dccp/ipv4.c
index 6d8f684..feb3fa5 100644
--- a/net/dccp/ipv4.c
+++ b/net/dccp/ipv4.c
@@ -1003,8 +1003,8 @@ static int __init dccp_v4_init(void)

  inet_register_protosw(&dccp_v4_protosw);

- err = inet_csk_ctl_sock_create(&socket, PF_INET,
-           SOCK_DCCP, IPPROTO_DCCP);
+ err = inet_ctl_sock_create(&socket, PF_INET,
+       SOCK_DCCP, IPPROTO_DCCP);
  if (err)
```

```
   goto out_unregister_protosw;
  dccp_v4_ctl_sk = socket->sk;
diff --git a/net/dccp/ipv6.c b/net/dccp/ipv6.c
index c5d9d1b..5690fbd 100644
--- a/net/dccp/ipv6.c
+++ b/net/dccp/ipv6.c
@@ -1185,8 +1185,8 @@ static int __init dccp_v6_init(void)

  inet6_register_protosw(&dccp_v6_protosw);

- err = inet_csk_ctl_sock_create(&socket, PF_INET6,
-            SOCK_DCCP, IPPROTO_DCCP);
+ err = inet_ctl_sock_create(&socket, PF_INET6,
+        SOCK_DCCP, IPPROTO_DCCP);
  if (err != 0)
   goto out_unregister_protosw;
  dccp_v6_ctl_sk = socket->sk;
diff --git a/net/ipv4/af_inet.c b/net/ipv4/af_inet.c
index 5882a13..7ab0bd6 100644
--- a/net/ipv4/af_inet.c
+++ b/net/ipv4/af_inet.c
@@ -1250,6 +1250,25 @@ out:
  return segs;
 }

+int inet_ctl_sock_create(struct socket **sock, unsigned short family,
+     unsigned short type, unsigned char protocol)
+{
+ int rc = sock_create_kern(family, type, protocol, sock);
+
+ if (rc == 0) {
+  (*sock)->sk->sk_allocation = GFP_ATOMIC;
+  inet_sk((*sock)->sk)->uc_ttl = -1;
+  /*
+   * Unhash it so that IP input processing does not even see it,
+   * we do not wish this socket to see incoming packets.
+   */
+  (*sock)->sk->sk_prot->unhash((*sock)->sk);
+ }
+ return rc;
+}
+
+EXPORT_SYMBOL_GPL(inet_ctl_sock_create);
+
 unsigned long snmp_fold_field(void *mib[], int offt)
 {
  unsigned long res = 0;
diff --git a/net/ipv4/inet_connection_sock.c b/net/ipv4/inet_connection_sock.c
```

```
index a7fcaf2..ee55678 100644
--- a/net/ipv4/inet_connection_sock.c
+++ b/net/ipv4/inet_connection_sock.c
@@ -651,25 +651,6 @@ void inet_csk_addr2sockaddr(struct sock *sk, struct sockaddr *uaddr)

 EXPORT_SYMBOL_GPL(inet_csk_addr2sockaddr);

-int inet_csk_ctl_sock_create(struct socket **sock, unsigned short family,
-        unsigned short type, unsigned char protocol)
-{
- int rc = sock_create_kern(family, type, protocol, sock);
-
- if (rc == 0) {
-  (*sock)->sk->sk_allocation = GFP_ATOMIC;
-  inet_sk((*sock)->sk)->uc_ttl = -1;
-  /*
-   * Unhash it so that IP input processing does not even see it,
-   * we do not wish this socket to see incoming packets.
-   */
-  (*sock)->sk->sk_prot->unhash((*sock)->sk);
- }
- return rc;
-}
-
-EXPORT_SYMBOL_GPL(inet_csk_ctl_sock_create);
-
 #ifdef CONFIG_COMPAT
 int inet_csk_compat_getsockopt(struct sock *sk, int level, int optname,
           char __user *optval, int __user *optlen)
diff --git a/net/ipv4/tcp_ipv4.c b/net/ipv4/tcp_ipv4.c
index 1d77f37..edf5a37 100644
--- a/net/ipv4/tcp_ipv4.c
+++ b/net/ipv4/tcp_ipv4.c
@@ -2491,8 +2491,8 @@ struct proto tcp_prot = {
 void __init tcp_v4_init(void)
 {
  struct socket *__tcp_socket;
- if (inet_csk_ctl_sock_create(&__tcp_socket, PF_INET, SOCK_RAW,
-         IPPROTO_TCP) < 0)
+ if (inet_ctl_sock_create(&__tcp_socket, PF_INET, SOCK_RAW,
+     IPPROTO_TCP) < 0)
   panic("Failed to create the TCP control socket.\n");
  tcp_sock = __tcp_socket->sk;
 }
diff --git a/net/ipv6/tcp_ipv6.c b/net/ipv6/tcp_ipv6.c
index 6d851c3..d98222f 100644
--- a/net/ipv6/tcp_ipv6.c
+++ b/net/ipv6/tcp_ipv6.c
```

```
@@ -60,6 +60,7 @@
 #include <net/dsfield.h>
 #include <net/timewait_sock.h>
 #include <net/netdma.h>
+#include <net/inet_common.h>

 #include <asm/uaccess.h>

@@ -2202,7 +2203,7 @@ static int tcpv6_net_init(struct net *net)
  struct socket *sock;
  struct sock *sk;

- err = inet_csk_ctl_sock_create(&sock, PF_INET6, SOCK_RAW, IPPROTO_TCP);
+ err = inet_ctl_sock_create(&sock, PF_INET6, SOCK_RAW, IPPROTO_TCP);
  if (err)
   return err;

--
1.5.3.rc5
```

Subject: [PATCH 5/13 net-2.6.26] [SCTP]: Use inet_ctl_sock_create for control socket creation.

Posted by den on Mon, 31 Mar 2008 13:16:45 GMT

View Forum Message <> Reply to Message

sk->sk_proc->(un)hash is noop right now, so the unification is correct.

Signed-off-by: Denis V. Lunev <den@openvz.org>

```
---
 net/sctp/protocol.c |    7 ++-----
 1 files changed, 2 insertions(+), 5 deletions(-)

diff --git a/net/sctp/protocol.c b/net/sctp/protocol.c
index 5aea911..6a3be58 100644
--- a/net/sctp/protocol.c
+++ b/net/sctp/protocol.c
@@ -680,16 +680,13 @@ static int sctp_ctl_sock_init(void)
  else
   family = PF_INET;

- err = sock_create_kern(family, SOCK_SEQPACKET, IPPROTO_SCTP,
-        &sctp_ctl_socket);
```

```
+ err = inet_ctl_sock_create(&sctp_ctl_socket, family,
+       SOCK_SEQPACKET, IPPROTO_SCTP);
  if (err < 0) {
   printk(KERN_ERR
         "SCTP: Failed to create the SCTP control socket.\n");
   return err;
  }
- sctp_ctl_socket->sk->sk_allocation = GFP_ATOMIC;
- inet_sk(sctp_ctl_socket->sk)->uc_ttl = -1;
-
  return 0;
 }

--
1.5.3.rc5
```

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

---

Subject: [PATCH 6/13 net-2.6.26] [SCTP]: Replace socket with sock for SCTP control socket.
Posted by den on Mon, 31 Mar 2008 13:16:46 GMT
View Forum Message <> Reply to Message

```
Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 net/sctp/protocol.c |   12 +++++++-----
 1 files changed, 7 insertions(+), 5 deletions(-)

diff --git a/net/sctp/protocol.c b/net/sctp/protocol.c
index 6a3be58..ac0833c 100644
--- a/net/sctp/protocol.c
+++ b/net/sctp/protocol.c
@@ -74,7 +74,7 @@ DEFINE_SPINLOCK(sctp_assocs_id_lock);
  * the Out-of-the-blue (OOTB) packets.  A control sock will be created
  * for this socket at the initialization time.
  */
-static struct socket *sctp_ctl_socket;
+static struct sock *sctp_ctl_sock;

 static struct sctp_pf *sctp_pf_inet6_specific;
 static struct sctp_pf *sctp_pf_inet_specific;
@@ -91,7 +91,7 @@ int sysctl_sctp_wmem[3];
 /* Return the address of the control sock. */
 struct sock *sctp_get_ctl_sock(void)
```

```
 {
- return sctp_ctl_socket->sk;
+ return sctp_ctl_sock;
 }

 /* Set up the proc fs entry for the SCTP protocol. */
@@ -674,19 +674,21 @@ static int sctp_ctl_sock_init(void)
 {
  int err;
  sa_family_t family;
+ struct socket *socket;

  if (sctp_get_pf_specific(PF_INET6))
   family = PF_INET6;
  else
   family = PF_INET;

- err = inet_ctl_sock_create(&sctp_ctl_socket, family,
+ err = inet_ctl_sock_create(&socket, family,
        SOCK_SEQPACKET, IPPROTO_SCTP);
  if (err < 0) {
   printk(KERN_ERR
          "SCTP: Failed to create the SCTP control socket.\n");
   return err;
  }
+ sctp_ctl_sock = socket->sk;
  return 0;
 }

@@ -1284,7 +1286,7 @@ err_v6_add_protocol:
  sctp_v6_del_protocol();
 err_add_protocol:
  sctp_v4_del_protocol();
- sock_release(sctp_ctl_socket);
+ sock_release(sctp_ctl_sock->sk_socket);
 err_ctl_sock_init:
  sctp_v6_protosw_exit();
 err_v6_protosw_init:
@@ -1328,7 +1330,7 @@ SCTP_STATIC __exit void sctp_exit(void)
  sctp_v4_del_protocol();

  /* Free the control endpoint.  */
- sock_release(sctp_ctl_socket);
+ sock_release(sctp_ctl_sock->sk_socket);

  /* Free protosw registrations */
  sctp_v6_protosw_exit();
--
```

1.5.3.rc5

_____

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

---

Subject: [PATCH 7/13 net-2.6.26] [INET]: Let inet_ctl_sock_create return sock rather than socket.
Posted by den on Mon, 31 Mar 2008 13:16:47 GMT
View Forum Message <> Reply to Message

All upper protocol layers are already use sock internally.

Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 include/net/inet_common.h | 2 +-
 net/dccp/ipv4.c | 4 +---
 net/dccp/ipv6.c | 4 +---
 net/ipv4/af_inet.c | 12 +++++++-----
 net/ipv4/tcp_ipv4.c | 4 +---
 net/ipv6/tcp_ipv6.c | 5 ++---
 net/sctp/protocol.c | 4 +---
 7 files changed, 14 insertions(+), 21 deletions(-)

```
diff --git a/include/net/inet_common.h b/include/net/inet_common.h
index d6238bd..4bfcf3f 100644
--- a/include/net/inet_common.h
+++ b/include/net/inet_common.h
@@ -39,7 +39,7 @@ extern int   inet_getname(struct socket *sock,
 extern int   inet_ioctl(struct socket *sock,
         unsigned int cmd, unsigned long arg);

-extern int   inet_ctl_sock_create(struct socket **sock,
+extern int   inet_ctl_sock_create(struct sock **sk,
            unsigned short family,
            unsigned short type,
            unsigned char protocol);
diff --git a/net/dccp/ipv4.c b/net/dccp/ipv4.c
index feb3fa5..5669c89 100644
--- a/net/dccp/ipv4.c
+++ b/net/dccp/ipv4.c
@@ -991,7 +991,6 @@ static struct inet_protosw dccp_v4_protosw = {

 static int __init dccp_v4_init(void)
 {
- struct socket *socket;
```

```
  int err = proto_register(&dccp_v4_prot, 1);

  if (err != 0)
@@ -1003,11 +1002,10 @@ static int __init dccp_v4_init(void)

  inet_register_protosw(&dccp_v4_protosw);

- err = inet_ctl_sock_create(&socket, PF_INET,
+ err = inet_ctl_sock_create(&dccp_v4_ctl_sk, PF_INET,
        SOCK_DCCP, IPPROTO_DCCP);
  if (err)
   goto out_unregister_protosw;
- dccp_v4_ctl_sk = socket->sk;
 out:
  return err;
 out_unregister_protosw:
diff --git a/net/dccp/ipv6.c b/net/dccp/ipv6.c
index 5690fbd..cf598bf 100644
--- a/net/dccp/ipv6.c
+++ b/net/dccp/ipv6.c
@@ -1173,7 +1173,6 @@ static struct inet_protosw dccp_v6_protosw = {

 static int __init dccp_v6_init(void)
 {
- struct socket *socket;
  int err = proto_register(&dccp_v6_prot, 1);

  if (err != 0)
@@ -1185,11 +1184,10 @@ static int __init dccp_v6_init(void)

  inet6_register_protosw(&dccp_v6_protosw);

- err = inet_ctl_sock_create(&socket, PF_INET6,
+ err = inet_ctl_sock_create(&dccp_v6_ctl_sk, PF_INET6,
        SOCK_DCCP, IPPROTO_DCCP);
  if (err != 0)
   goto out_unregister_protosw;
- dccp_v6_ctl_sk = socket->sk;
 out:
  return err;
 out_unregister_protosw:
diff --git a/net/ipv4/af_inet.c b/net/ipv4/af_inet.c
index 7ab0bd6..cad664b 100644
--- a/net/ipv4/af_inet.c
+++ b/net/ipv4/af_inet.c
@@ -1250,19 +1250,21 @@ out:
  return segs;
 }
```

```
-int inet_ctl_sock_create(struct socket **sock, unsigned short family,
+int inet_ctl_sock_create(struct sock **sk, unsigned short family,
     unsigned short type, unsigned char protocol)
 {
- int rc = sock_create_kern(family, type, protocol, sock);
+ struct socket *sock;
+ int rc = sock_create_kern(family, type, protocol, &sock);

  if (rc == 0) {
-  (*sock)->sk->sk_allocation = GFP_ATOMIC;
-  inet_sk((*sock)->sk)->uc_ttl = -1;
+  *sk = sock->sk;
+  (*sk)->sk_allocation = GFP_ATOMIC;
+  inet_sk(*sk)->uc_ttl = -1;
   /*
    * Unhash it so that IP input processing does not even see it,
    * we do not wish this socket to see incoming packets.
    */
-  (*sock)->sk->sk_prot->unhash((*sock)->sk);
+  (*sk)->sk_prot->unhash(*sk);
  }
  return rc;
 }
diff --git a/net/ipv4/tcp_ipv4.c b/net/ipv4/tcp_ipv4.c
index edf5a37..cfe5df7 100644
--- a/net/ipv4/tcp_ipv4.c
+++ b/net/ipv4/tcp_ipv4.c
@@ -2490,11 +2490,9 @@ struct proto tcp_prot = {

 void __init tcp_v4_init(void)
 {
- struct socket *__tcp_socket;
- if (inet_ctl_sock_create(&__tcp_socket, PF_INET, SOCK_RAW,
+ if (inet_ctl_sock_create(&tcp_sock, PF_INET, SOCK_RAW,
      IPPROTO_TCP) < 0)
   panic("Failed to create the TCP control socket.\n");
- tcp_sock = __tcp_socket->sk;
 }

 EXPORT_SYMBOL(ipv4_specific);
diff --git a/net/ipv6/tcp_ipv6.c b/net/ipv6/tcp_ipv6.c
index d98222f..2882cc5 100644
--- a/net/ipv6/tcp_ipv6.c
+++ b/net/ipv6/tcp_ipv6.c
@@ -2200,14 +2200,13 @@ static struct inet_protosw tcpv6_protosw = {
 static int tcpv6_net_init(struct net *net)
 {
```

```
  int err;
- struct socket *sock;
  struct sock *sk;

- err = inet_ctl_sock_create(&sock, PF_INET6, SOCK_RAW, IPPROTO_TCP);
+ err = inet_ctl_sock_create(&sk, PF_INET6, SOCK_RAW, IPPROTO_TCP);
  if (err)
   return err;

- net->ipv6.tcp_sk = sk = sock->sk;
+ net->ipv6.tcp_sk = sk;
  sk_change_net(sk, net);
  return err;
 }
diff --git a/net/sctp/protocol.c b/net/sctp/protocol.c
index ac0833c..3c08d33 100644
--- a/net/sctp/protocol.c
+++ b/net/sctp/protocol.c
@@ -674,21 +674,19 @@ static int sctp_ctl_sock_init(void)
 {
  int err;
  sa_family_t family;
- struct socket *socket;

  if (sctp_get_pf_specific(PF_INET6))
   family = PF_INET6;
  else
   family = PF_INET;

- err = inet_ctl_sock_create(&socket, family,
+ err = inet_ctl_sock_create(&sctp_ctl_sock, family,
        SOCK_SEQPACKET, IPPROTO_SCTP);
  if (err < 0) {
   printk(KERN_ERR
          "SCTP: Failed to create the SCTP control socket.\n");
   return err;
  }
- sctp_ctl_sock = socket->sk;
  return 0;
 }

--
1.5.3.rc5
```

Subject: [PATCH 8/13 net-2.6.26] [NETNS]: Inet control socket should not hold a namespace.
Posted by den on Mon, 31 Mar 2008 13:16:48 GMT
View Forum Message <> Reply to Message

This is a generic requirement, so make inet_ctl_sock_create namespace
aware and create a inet_ctl_sock_destroy wrapper around sk_release_kernel.

Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 include/net/inet_common.h | 8 +++++++-
 net/dccp/ipv4.c | 4 ++--
 net/dccp/ipv6.c | 4 ++--
 net/ipv4/af_inet.c | 5 ++++-
 net/ipv4/tcp_ipv4.c | 2 +-
 net/ipv6/tcp_ipv6.c | 14 +++++----------
 net/sctp/protocol.c | 6 +++---
 7 files changed, 22 insertions(+), 21 deletions(-)

```
diff --git a/include/net/inet_common.h b/include/net/inet_common.h
index 4bfcf3f..18c7732 100644
--- a/include/net/inet_common.h
+++ b/include/net/inet_common.h
@@ -42,7 +42,13 @@ extern int   inet_ioctl(struct socket *sock,
 extern int   inet_ctl_sock_create(struct sock **sk,
             unsigned short family,
             unsigned short type,
-            unsigned char protocol);
+            unsigned char protocol,
+            struct net *net);
+
+static inline void inet_ctl_sock_destroy(struct sock *sk)
+{
+ sk_release_kernel(sk);
+}

 #endif

diff --git a/net/dccp/ipv4.c b/net/dccp/ipv4.c
index 5669c89..b12803b 100644
--- a/net/dccp/ipv4.c
+++ b/net/dccp/ipv4.c
@@ -1003,7 +1003,7 @@ static int __init dccp_v4_init(void)
  inet_register_protosw(&dccp_v4_protosw);

  err = inet_ctl_sock_create(&dccp_v4_ctl_sk, PF_INET,
-       SOCK_DCCP, IPPROTO_DCCP);
+       SOCK_DCCP, IPPROTO_DCCP, &init_net);
  if (err)
```

```
   goto out_unregister_protosw;
 out:
@@ -1018,7 +1018,7 @@ out_proto_unregister:

 static void __exit dccp_v4_exit(void)
 {
- sock_release(dccp_v4_ctl_sk->sk_socket);
+ inet_ctl_sock_destroy(dccp_v4_ctl_sk);
  inet_unregister_protosw(&dccp_v4_protosw);
  inet_del_protocol(&dccp_v4_protocol, IPPROTO_DCCP);
  proto_unregister(&dccp_v4_prot);
diff --git a/net/dccp/ipv6.c b/net/dccp/ipv6.c
index cf598bf..94d749e 100644
--- a/net/dccp/ipv6.c
+++ b/net/dccp/ipv6.c
@@ -1185,7 +1185,7 @@ static int __init dccp_v6_init(void)
  inet6_register_protosw(&dccp_v6_protosw);

  err = inet_ctl_sock_create(&dccp_v6_ctl_sk, PF_INET6,
-       SOCK_DCCP, IPPROTO_DCCP);
+       SOCK_DCCP, IPPROTO_DCCP, &init_net);
  if (err != 0)
   goto out_unregister_protosw;
 out:
@@ -1200,7 +1200,7 @@ out_unregister_proto:

 static void __exit dccp_v6_exit(void)
 {
- sock_release(dccp_v6_ctl_sk->sk_socket);
+ inet_ctl_sock_destroy(dccp_v6_ctl_sk);
  inet6_del_protocol(&dccp_v6_protocol, IPPROTO_DCCP);
  inet6_unregister_protosw(&dccp_v6_protosw);
  proto_unregister(&dccp_v6_prot);
diff --git a/net/ipv4/af_inet.c b/net/ipv4/af_inet.c
index cad664b..cf766ad 100644
--- a/net/ipv4/af_inet.c
+++ b/net/ipv4/af_inet.c
@@ -1251,7 +1251,8 @@ out:
 }

 int inet_ctl_sock_create(struct sock **sk, unsigned short family,
-    unsigned short type, unsigned char protocol)
+    unsigned short type, unsigned char protocol,
+    struct net *net)
 {
  struct socket *sock;
  int rc = sock_create_kern(family, type, protocol, &sock);
@@ -1265,6 +1266,8 @@ int inet_ctl_sock_create(struct sock **sk, unsigned short family,
```

```
     * we do not wish this socket to see incoming packets.
     */
    (*sk)->sk_prot->unhash(*sk);
+
+   sk_change_net(*sk, net);
   }
   return rc;
 }
diff --git a/net/ipv4/tcp_ipv4.c b/net/ipv4/tcp_ipv4.c
index cfe5df7..dc8c3dc 100644
--- a/net/ipv4/tcp_ipv4.c
+++ b/net/ipv4/tcp_ipv4.c
@@ -2491,7 +2491,7 @@ struct proto tcp_prot = {
 void __init tcp_v4_init(void)
 {
   if (inet_ctl_sock_create(&tcp_sock, PF_INET, SOCK_RAW,
-      IPPROTO_TCP) < 0)
+      IPPROTO_TCP, &init_net) < 0)
     panic("Failed to create the TCP control socket.\n");
 }

diff --git a/net/ipv6/tcp_ipv6.c b/net/ipv6/tcp_ipv6.c
index 2882cc5..378cc40 100644
--- a/net/ipv6/tcp_ipv6.c
+++ b/net/ipv6/tcp_ipv6.c
@@ -2199,21 +2199,13 @@ static struct inet_protosw tcpv6_protosw = {

 static int tcpv6_net_init(struct net *net)
 {
- int err;
- struct sock *sk;
-
- err = inet_ctl_sock_create(&sk, PF_INET6, SOCK_RAW, IPPROTO_TCP);
- if (err)
-   return err;
-
- net->ipv6.tcp_sk = sk;
- sk_change_net(sk, net);
- return err;
+ return inet_ctl_sock_create(&net->ipv6.tcp_sk, PF_INET6,
+         SOCK_RAW, IPPROTO_TCP, net);
 }

 static void tcpv6_net_exit(struct net *net)
 {
- sk_release_kernel(net->ipv6.tcp_sk);
+ inet_ctl_sock_destroy(net->ipv6.tcp_sk);
 }
```

```
 static struct pernet_operations tcpv6_net_ops = {
diff --git a/net/sctp/protocol.c b/net/sctp/protocol.c
index 3c08d33..067c8a1 100644
--- a/net/sctp/protocol.c
+++ b/net/sctp/protocol.c
@@ -681,7 +681,7 @@ static int sctp_ctl_sock_init(void)
   family = PF_INET;

  err = inet_ctl_sock_create(&sctp_ctl_sock, family,
-       SOCK_SEQPACKET, IPPROTO_SCTP);
+       SOCK_SEQPACKET, IPPROTO_SCTP, &init_net);
  if (err < 0) {
   printk(KERN_ERR
          "SCTP: Failed to create the SCTP control socket.\n");
@@ -1284,7 +1284,7 @@ err_v6_add_protocol:
  sctp_v6_del_protocol();
 err_add_protocol:
  sctp_v4_del_protocol();
- sock_release(sctp_ctl_sock->sk_socket);
+ inet_ctl_sock_destroy(sctp_ctl_sock);
 err_ctl_sock_init:
  sctp_v6_protosw_exit();
 err_v6_protosw_init:
@@ -1328,7 +1328,7 @@ SCTP_STATIC __exit void sctp_exit(void)
  sctp_v4_del_protocol();

  /* Free the control endpoint.  */
- sock_release(sctp_ctl_sock->sk_socket);
+ inet_ctl_sock_destroy(sctp_ctl_sock);

  /* Free protosw registrations */
  sctp_v6_protosw_exit();
--
1.5.3.rc5
```

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

---

Subject: [PATCH 9/13 net-2.6.26] [ICMP]: Simplify ICMP control socket creation.
Posted by den on Mon, 31 Mar 2008 13:16:49 GMT

View Forum Message <> Reply to Message

Replace sock_create_kern with inet_ctl_sock_create.

```
Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 net/ipv4/icmp.c |   25 +++++++------------------
 1 files changed, 7 insertions(+), 18 deletions(-)

diff --git a/net/ipv4/icmp.c b/net/ipv4/icmp.c
index 803bc9f..efc7cbe 100644
--- a/net/ipv4/icmp.c
+++ b/net/ipv4/icmp.c
@@ -93,6 +93,7 @@
 #include <asm/uaccess.h>
 #include <net/checksum.h>
 #include <net/xfrm.h>
+#include <net/inet_common.h>

 /*
  * Build xmit assembly blocks
@@ -1136,7 +1137,7 @@ static void __net_exit icmp_sk_exit(struct net *net)
  int i;

  for_each_possible_cpu(i)
-  sk_release_kernel(net->ipv4.icmp_sk[i]);
+  inet_ctl_sock_destroy(net->ipv4.icmp_sk[i]);
  kfree(net->ipv4.icmp_sk);
  net->ipv4.icmp_sk = NULL;
 }
@@ -1152,17 +1153,13 @@ int __net_init icmp_sk_init(struct net *net)

  for_each_possible_cpu(i) {
   struct sock *sk;
-  struct socket *sock;
-  struct inet_sock *inet;

-  err = sock_create_kern(PF_INET, SOCK_RAW, IPPROTO_ICMP, &sock);
+  err = inet_ctl_sock_create(&sk, PF_INET,
+        SOCK_RAW, IPPROTO_ICMP, net);
   if (err < 0)
    goto fail;

-  net->ipv4.icmp_sk[i] = sk = sock->sk;
-  sk_change_net(sk, net);
-
-  sk->sk_allocation = GFP_ATOMIC;
+  net->ipv4.icmp_sk[i] = sk;

   /* Enough space for 2 64K ICMP packets, including
    * sk_buff struct overhead.
@@ -1170,15 +1167,7 @@ int __net_init icmp_sk_init(struct net *net)
```

```
   sk->sk_sndbuf =
    (2 * ((64 * 1024) + sizeof(struct sk_buff)));

-  inet = inet_sk(sk);
-  inet->uc_ttl = -1;
-  inet->pmtudisc = IP_PMTUDISC_DONT;
-
-  /* Unhash it so that IP input processing does not even
-   * see it, we do not wish this socket to see incoming
-   * packets.
-   */
-  sk->sk_prot->unhash(sk);
+  inet_sk(sk)->pmtudisc = IP_PMTUDISC_DONT;
  }

  /* Control parameters for ECHO replies. */
@@ -1208,7 +1197,7 @@ int __net_init icmp_sk_init(struct net *net)

 fail:
  for_each_possible_cpu(i)
-  sk_release_kernel(net->ipv4.icmp_sk[i]);
+  inet_ctl_sock_destroy(net->ipv4.icmp_sk[i]);
  kfree(net->ipv4.icmp_sk);
  return err;
 }
--
1.5.3.rc5
```

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

---

Subject: [PATCH 10/13 net-2.6.26] [INET]: uc_ttl assignment in inet_ctl_sock_create is redundant.

Posted by den on Mon, 31 Mar 2008 13:16:50 GMT

View Forum Message <> Reply to Message

uc_ttl is initialized in inet(6)_create and never changed except setsockopt
ioctl. Remove this assignment.

Signed-off-by: Denis V. Lunev <den@openvz.org>

```
---
 net/ipv4/af_inet.c |    1 -
 1 files changed, 0 insertions(+), 1 deletions(-)

diff --git a/net/ipv4/af_inet.c b/net/ipv4/af_inet.c
```

```
index cf766ad..72ae8ed 100644
--- a/net/ipv4/af_inet.c
+++ b/net/ipv4/af_inet.c
@@ -1260,7 +1260,6 @@ int inet_ctl_sock_create(struct sock **sk, unsigned short family,
  if (rc == 0) {
   *sk = sock->sk;
   (*sk)->sk_allocation = GFP_ATOMIC;
-  inet_sk(*sk)->uc_ttl = -1;
   /*
    * Unhash it so that IP input processing does not even see it,
    * we do not wish this socket to see incoming packets.
--
1.5.3.rc5
```

Subject: [PATCH 11/13 net-2.6.26] [IPV6]: Simplify IPv6 control sockets creation.
Posted by den on Mon, 31 Mar 2008 13:16:51 GMT
View Forum Message <> Reply to Message

Do this by replacing sock_create_kern with inet_ctl_sock_create.

Signed-off-by: Denis V. Lunev <den@openvz.org>

```
---
 net/ipv6/icmp.c  |   16 ++++++----------
 net/ipv6/mcast.c |   19 ++++++----------
 net/ipv6/ndisc.c |   12 +++++-------
 3 files changed, 17 insertions(+), 30 deletions(-)

diff --git a/net/ipv6/icmp.c b/net/ipv6/icmp.c
index 63309d1..24a657e 100644
--- a/net/ipv6/icmp.c
+++ b/net/ipv6/icmp.c
@@ -64,6 +64,7 @@
 #include <net/addrconf.h>
 #include <net/icmp.h>
 #include <net/xfrm.h>
+#include <net/inet_common.h>

 #include <asm/uaccess.h>
 #include <asm/system.h>
@@ -808,9 +809,8 @@ static int __net_init icmpv6_sk_init(struct net *net)
   return -ENOMEM;
```

```
  for_each_possible_cpu(i) {
-  struct socket *sock;
-  err = sock_create_kern(PF_INET6, SOCK_RAW, IPPROTO_ICMPV6,
-           &sock);
+  err = inet_ctl_sock_create(&sk, PF_INET6,
+         SOCK_RAW, IPPROTO_ICMPV6, net);
    if (err < 0) {
     printk(KERN_ERR
           "Failed to initialize the ICMP6 control socket "
@@ -819,10 +819,8 @@ static int __net_init icmpv6_sk_init(struct net *net)
     goto fail;
    }

-  net->ipv6.icmp_sk[i] = sk = sock->sk;
-  sk_change_net(sk, net);
+  net->ipv6.icmp_sk[i] = sk;

-  sk->sk_allocation = GFP_ATOMIC;
    /*
     * Split off their lock-class, because sk->sk_dst_lock
     * gets used from softirqs, which is safe for
@@ -837,14 +835,12 @@ static int __net_init icmpv6_sk_init(struct net *net)
     */
    sk->sk_sndbuf =
     (2 * ((64 * 1024) + sizeof(struct sk_buff)));
-
-  sk->sk_prot->unhash(sk);
   }
   return 0;

   fail:
   for (j = 0; j < i; j++)
-  sk_release_kernel(net->ipv6.icmp_sk[j]);
+  inet_ctl_sock_destroy(net->ipv6.icmp_sk[j]);
   kfree(net->ipv6.icmp_sk);
   return err;
  }
@@ -854,7 +850,7 @@ static void __net_exit icmpv6_sk_exit(struct net *net)
   int i;

   for_each_possible_cpu(i) {
-  sk_release_kernel(net->ipv6.icmp_sk[i]);
+  inet_ctl_sock_destroy(net->ipv6.icmp_sk[i]);
   }
   kfree(net->ipv6.icmp_sk);
  }
diff --git a/net/ipv6/mcast.c b/net/ipv6/mcast.c
index d810cff..2e6a53f 100644
```

```
--- a/net/ipv6/mcast.c
+++ b/net/ipv6/mcast.c
@@ -59,6 +59,7 @@
 #include <net/ndisc.h>
 #include <net/addrconf.h>
 #include <net/ip6_route.h>
+#include <net/inet_common.h>

 #include <net/ip6_checksum.h>

@@ -2672,12 +2673,10 @@ static void igmp6_proc_exit(struct net *net)

 static int igmp6_net_init(struct net *net)
 {
- struct ipv6_pinfo *np;
- struct socket *sock;
- struct sock *sk;
  int err;

- err = sock_create_kern(PF_INET6, SOCK_RAW, IPPROTO_ICMPV6, &sock);
+ err = inet_ctl_sock_create(&net->ipv6.igmp_sk, PF_INET6,
+       SOCK_RAW, IPPROTO_ICMPV6, net);
  if (err < 0) {
   printk(KERN_ERR
          "Failed to initialize the IGMP6 control socket (err %d).\n",
@@ -2685,13 +2684,7 @@ static int igmp6_net_init(struct net *net)
   goto out;
  }

- net->ipv6.igmp_sk = sk = sock->sk;
- sk_change_net(sk, net);
- sk->sk_allocation = GFP_ATOMIC;
- sk->sk_prot->unhash(sk);
-
- np = inet6_sk(sk);
- np->hop_limit = 1;
+ inet6_sk(net->ipv6.igmp_sk)->hop_limit = 1;

  err = igmp6_proc_init(net);
  if (err)
@@ -2700,13 +2693,13 @@ out:
  return err;

 out_sock_create:
- sk_release_kernel(net->ipv6.igmp_sk);
+ inet_ctl_sock_destroy(net->ipv6.igmp_sk);
  goto out;
 }
```

```
 static void igmp6_net_exit(struct net *net)
 {
- sk_release_kernel(net->ipv6.igmp_sk);
+ inet_ctl_sock_destroy(net->ipv6.igmp_sk);
  igmp6_proc_exit(net);
 }

diff --git a/net/ipv6/ndisc.c b/net/ipv6/ndisc.c
index 510aa74..06d80c6 100644
--- a/net/ipv6/ndisc.c
+++ b/net/ipv6/ndisc.c
@@ -84,6 +84,7 @@

 #include <net/flow.h>
 #include <net/ip6_checksum.h>
+#include <net/inet_common.h>
 #include <linux/proc_fs.h>

 #include <linux/netfilter.h>
@@ -1731,12 +1732,12 @@ static int ndisc_ifinfo_sysctl_strategy(ctl_table *ctl, int __user
*name,

 static int ndisc_net_init(struct net *net)
 {
- struct socket *sock;
  struct ipv6_pinfo *np;
  struct sock *sk;
  int err;

- err = sock_create_kern(PF_INET6, SOCK_RAW, IPPROTO_ICMPV6, &sock);
+ err = inet_ctl_sock_create(&sk, PF_INET6,
+       SOCK_RAW, IPPROTO_ICMPV6, net);
  if (err < 0) {
   ND_PRINTK0(KERN_ERR
       "ICMPv6 NDISC: Failed to initialize the control socket (err %d).\n",
@@ -1744,22 +1745,19 @@ static int ndisc_net_init(struct net *net)
   return err;
  }

- net->ipv6.ndisc_sk = sk = sock->sk;
- sk_change_net(sk, net);
+ net->ipv6.ndisc_sk = sk;

  np = inet6_sk(sk);
- sk->sk_allocation = GFP_ATOMIC;
  np->hop_limit = 255;
  /* Do not loopback ndisc messages */
```

```
  np->mc_loop = 0;
- sk->sk_prot->unhash(sk);

  return 0;
 }

 static void ndisc_net_exit(struct net *net)
 {
- sk_release_kernel(net->ipv6.ndisc_sk);
+ inet_ctl_sock_destroy(net->ipv6.ndisc_sk);
 }

 static struct pernet_operations ndisc_net_ops = {
--
1.5.3.rc5
```

Subject: [PATCH 12/13 net-2.6.26] [NETNS]: Create tcp control socket in the each namespace.
Posted by den on Mon, 31 Mar 2008 13:16:52 GMT
View Forum Message <> Reply to Message

```
Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 include/net/netns/ipv4.h |    1 +
 net/ipv4/tcp_ipv4.c      |   21 ++++++++++++++++++---
 2 files changed, 20 insertions(+), 2 deletions(-)

diff --git a/include/net/netns/ipv4.h b/include/net/netns/ipv4.h
index af685f7..34ee348 100644
--- a/include/net/netns/ipv4.h
+++ b/include/net/netns/ipv4.h
@@ -28,6 +28,7 @@ struct netns_ipv4 {
  struct sock  *fibnl;

  struct sock  **icmp_sk;
+ struct sock  *tcp_sock;

  struct netns_frags frags;
 #ifdef CONFIG_NETFILTER
diff --git a/net/ipv4/tcp_ipv4.c b/net/ipv4/tcp_ipv4.c
index dc8c3dc..1d4a77a 100644
--- a/net/ipv4/tcp_ipv4.c
```

```
+++ b/net/ipv4/tcp_ipv4.c
@@ -2488,11 +2488,28 @@ struct proto tcp_prot = {
 #endif
 };

+
+static int __net_init tcp_sk_init(struct net *net)
+{
+ return inet_ctl_sock_create(&net->ipv4.tcp_sock,
+         PF_INET, SOCK_RAW, IPPROTO_TCP, net);
+}
+
+static void __net_exit tcp_sk_exit(struct net *net)
+{
+ inet_ctl_sock_destroy(net->ipv4.tcp_sock);
+}
+
+static struct pernet_operations __net_initdata tcp_sk_ops = {
+       .init = tcp_sk_init,
+       .exit = tcp_sk_exit,
+};
+
 void __init tcp_v4_init(void)
 {
- if (inet_ctl_sock_create(&tcp_sock, PF_INET, SOCK_RAW,
-     IPPROTO_TCP, &init_net) < 0)
+ if (register_pernet_device(&tcp_sk_ops))
    panic("Failed to create the TCP control socket.\n");
+ tcp_sock = init_net.ipv4.tcp_sock;
 }

 EXPORT_SYMBOL(ipv4_specific);
--
1.5.3.rc5
```

Subject: [PATCH 13/13 net-2.6.26] [NETNS]: Use TCP control socket from a correct namespace.
Posted by den on Mon, 31 Mar 2008 13:16:53 GMT
View Forum Message <> Reply to Message

Signed-off-by: Denis V.Lunev <den@openvz.org>

```
 net/ipv4/tcp_ipv4.c |   10 ++++------
 1 files changed, 4 insertions(+), 6 deletions(-)

diff --git a/net/ipv4/tcp_ipv4.c b/net/ipv4/tcp_ipv4.c
index 1d4a77a..df89a56 100644
--- a/net/ipv4/tcp_ipv4.c
+++ b/net/ipv4/tcp_ipv4.c
@@ -88,9 +88,6 @@ int sysctl_tcp_low_latency __read_mostly;
 /* Check TCP sequence numbers in ICMP packets. */
 #define ICMP_MIN_LENGTH 8

-/* Socket used for sending RSTs */
-static struct sock *tcp_sock __read_mostly;
-
 void tcp_v4_send_check(struct sock *sk, int len, struct sk_buff *skb);

 #ifdef CONFIG_TCP_MD5SIG
@@ -598,7 +595,8 @@ static void tcp_v4_send_reset(struct sock *sk, struct sk_buff *skb)
          sizeof(struct tcphdr), IPPROTO_TCP, 0);
  arg.csumoffset = offsetof(struct tcphdr, check) / 2;

- ip_send_reply(tcp_sock, skb, &arg, arg.iov[0].iov_len);
+ ip_send_reply(dev_net(skb->dst->dev)->ipv4.tcp_sock, skb,
+       &arg, arg.iov[0].iov_len);

  TCP_INC_STATS_BH(TCP_MIB_OUTSEGS);
  TCP_INC_STATS_BH(TCP_MIB_OUTRSTS);
@@ -693,7 +691,8 @@ static void tcp_v4_send_ack(struct tcp_timewait_sock *twsk,
  if (twsk)
   arg.bound_dev_if = twsk->tw_sk.tw_bound_dev_if;

- ip_send_reply(tcp_sock, skb, &arg, arg.iov[0].iov_len);
+ ip_send_reply(dev_net(skb->dev)->ipv4.tcp_sock, skb,
+       &arg, arg.iov[0].iov_len);

  TCP_INC_STATS_BH(TCP_MIB_OUTSEGS);
 }
@@ -2509,7 +2508,6 @@ void __init tcp_v4_init(void)
 {
  if (register_pernet_device(&tcp_sk_ops))
   panic("Failed to create the TCP control socket.\n");
- tcp_sock = init_net.ipv4.tcp_sock;
 }

 EXPORT_SYMBOL(ipv4_specific);
--
1.5.3.rc5
```

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

Subject: Re: [PATCH 1/13 net-2.6.26] [TCP]: Replace socket with sock for reset sending.
Posted by Arnaldo Carvalho de M on Mon, 31 Mar 2008 13:44:04 GMT
View Forum Message <> Reply to Message

Em Mon, Mar 31, 2008 at 05:16:41PM +0400, Denis V. Lunev escreveu:
> Replace tcp_socket with tcp_sock. This is more effective (less derefferences
> on fast paths). Additionally, the approach is unified to one used in ICMP.
>
> Signed-off-by: Denis V. Lunev <den@openvz.org>

Acked-by: Arnaldo Carvalho de Melo <acme@redhat.com>

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

Subject: Re: [PATCH 2/13 net-2.6.26] [DCCP]: Replace socket with sock for reset sending.
Posted by Arnaldo Carvalho de M on Mon, 31 Mar 2008 13:44:40 GMT
View Forum Message <> Reply to Message

Em Mon, Mar 31, 2008 at 05:16:42PM +0400, Denis V. Lunev escreveu:
> Replace dccp_v(4|6)_ctl_socket with sock to unify a code with TCP/ICMP.
>
> Signed-off-by: Denis V. Lunev <den@openvz.org>

Acked-by: Arnaldo Carvalho de Melo <acme@redhat.com>

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

Subject: Re: [PATCH 3/13 net-2.6.26] [DCCP]: dccp_v(4|6)_ctl_socket is leaked.
Posted by Arnaldo Carvalho de M on Mon, 31 Mar 2008 13:46:08 GMT
View Forum Message <> Reply to Message

Em Mon, Mar 31, 2008 at 05:16:43PM +0400, Denis V. Lunev escreveu:

> This seems a purism as module can't be unloaded, but though if cleanup method
> is present it should be correct and clean all staff created.
>
> Signed-off-by: Denis V. Lunev <den@openvz.org>

Acked-by: Arnaldo Carvalho de Melo <acme@redhat.com>

But please consider adding a inet_ctl_sock_destroy(), even if it only
does this sock_release, as it is the inet_ctl_sock_create() counterpart.

- Arnaldo

Subject: Re: [PATCH 5/13 net-2.6.26] [SCTP]: Use inet_ctl_sock_create for control socket creation.
Posted by Arnaldo Carvalho de M on Mon, 31 Mar 2008 13:46:32 GMT
View Forum Message <> Reply to Message

Em Mon, Mar 31, 2008 at 05:16:45PM +0400, Denis V. Lunev escreveu:
> sk->sk_proc->(un)hash is noop right now, so the unification is correct.
>
> Signed-off-by: Denis V. Lunev <den@openvz.org>

Acked-by: Arnaldo Carvalho de Melo <acme@redhat.com>

Subject: Re: [PATCH 6/13 net-2.6.26] [SCTP]: Replace socket with sock for SCTP control socket.
Posted by Arnaldo Carvalho de M on Mon, 31 Mar 2008 13:47:11 GMT
View Forum Message <> Reply to Message

Em Mon, Mar 31, 2008 at 05:16:46PM +0400, Denis V. Lunev escreveu:
> Signed-off-by: Denis V. Lunev <den@openvz.org>

Acked-by: Arnaldo Carvalho de Melo <acme@redhat.com>

Subject: Re: [PATCH 7/13 net-2.6.26] [INET]: Let inet_ctl_sock_create return sock rather than socket.
Posted by Arnaldo Carvalho de M on Mon, 31 Mar 2008 13:49:17 GMT
View Forum Message <> Reply to Message

Em Mon, Mar 31, 2008 at 05:16:47PM +0400, Denis V. Lunev escreveu:
> All upper protocol layers are already use sock internally.
>
> Signed-off-by: Denis V. Lunev <den@openvz.org>

Acked-by: Arnaldo Carvalho de Melo <acme@redhat.com>

With a nit: if you had done this just after the one that renamed
inet_ctl_sock_create you would avoid touching again the
inet_ctl_sock_create users :-)

- Arnaldo

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

Subject: Re: [PATCH 11/13 net-2.6.26] [IPV6]: Simplify IPv6 control sockets creation.
Posted by Arnaldo Carvalho de M on Mon, 31 Mar 2008 13:52:52 GMT
View Forum Message <> Reply to Message

Em Mon, Mar 31, 2008 at 05:16:51PM +0400, Denis V. Lunev escreveu:
> Do this by replacing sock_create_kern with inet_ctl_sock_create.
>
> Signed-off-by: Denis V. Lunev <den@openvz.org>

Acked-by: Arnaldo Carvalho de Melo <acme@redhat.com>

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

Subject: Re: [PATCH 0/13 net-2.6.26] UDP/ICMP/TCP for a namespace v2
Posted by davem on Thu, 03 Apr 2008 21:58:34 GMT
View Forum Message <> Reply to Message

From: "Denis V. Lunev" <den@openvz.org>
Date: Mon, 31 Mar 2008 17:16:07 +0400

> This patch set finally enables TCP/UDP and ICMP inside a namespace.

> In order to do this we fix ARP processing, IP options processing and
> allow packets to flow to namespace aware protocols.
>
> Finally, this makes namespace functional and alive :)
>
> Changed from v1:
> - inet_csk_ctl_sock_create is renamed to inet_ctl_sock_create. Thanks
>   to Arnaldo Carvalho de Melo <acme@redhat.com and Vlad Yasevich
>   <vladislav.yasevich@hp.com>
> - dccp_ctl_make_reset now accept sock rather than socket. Thanks to
>   Arnaldo Carvalho de Melo <acme@redhat.com>
> - added inet_ctl_sock_destroy
>
> Signed-off-by: Denis V. Lunev <den@openvz.org>

All applied and pushed out to net-2.6.26, thanks!

I also made sure to add all of the ACKs from Arnaldo,
as appropriate.
