Subject: [PATCH 4/6][NETNS]: Tcp-v4 sockets per-net lookup.
Posted by Pavel Emelianov on Thu, 31 Jan 2008 12:38:15 GMT
View Forum Message <> Reply to Message

Add a net argument to inet_lookup and propagate it further
into lookup calls. Plus tune the __inet_check_established.

The dccp and inet_diag, which use that lookup functions
pass the init_net into them.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

---
 include/net/inet_hashtables.h | 48 ++++++++++++++++++++++++++----------------
 net/dccp/ipv4.c | 6 ++--
 net/ipv4/inet_diag.c | 2 +-
 net/ipv4/inet_hashtables.c | 29 ++++++++++++++++++++++++++--------
 net/ipv4/tcp_ipv4.c | 15 ++++++++++++++++++------
 5 files changed, 58 insertions(+), 42 deletions(-)

diff --git a/include/net/inet_hashtables.h b/include/net/inet_hashtables.h
index 55532b9..c23c4ed 100644
--- a/include/net/inet_hashtables.h
+++ b/include/net/inet_hashtables.h
@@ -302,15 +302,17 @@ out:
   wake_up(&hashinfo->lhash_wait);
 }

-extern struct sock *__inet_lookup_listener(struct inet_hashinfo *hashinfo,
+extern struct sock *__inet_lookup_listener(struct net *net,
+   struct inet_hashinfo *hashinfo,
         const __be32 daddr,
         const unsigned short hnum,
         const int dif);

-static inline struct sock *inet_lookup_listener(struct inet_hashinfo *hashinfo,
-      __be32 daddr, __be16 dport, int dif)
+static inline struct sock *inet_lookup_listener(struct net *net,
+ struct inet_hashinfo *hashinfo,
+ __be32 daddr, __be16 dport, int dif)
 {
- return __inet_lookup_listener(hashinfo, daddr, ntohs(dport), dif);
+ return __inet_lookup_listener(net, hashinfo, daddr, ntohs(dport), dif);
 }

 /* Socket demux engine toys. */
@@ -344,26 +346,26 @@ typedef __u64 __bitwise __addrpair;
       (((__force __u64)(__be32)(__daddr)) << 32) | \

```
        ((__force __u64)(__be32)(__saddr)));
 #endif /* __BIG_ENDIAN */
-#define INET_MATCH(__sk, __hash, __cookie, __saddr, __daddr, __ports, __dif)\
- (((__sk)->sk_hash == (__hash))    && \
+#define INET_MATCH(__sk, __net, __hash, __cookie, __saddr, __daddr, __ports, __dif)\
+ (((__sk)->sk_hash == (__hash)) && ((__sk)->sk_net == (__net)) && \
   ((*((__addrpair *)&(inet_sk(__sk)->daddr))) == (__cookie)) && \
   ((*((__portpair *)&(inet_sk(__sk)->dport))) == (__ports)) && \
   (!((__sk)->sk_bound_dev_if) || ((__sk)->sk_bound_dev_if == (__dif))))
-#define INET_TW_MATCH(__sk, __hash, __cookie, __saddr, __daddr, __ports, __dif)\
- (((__sk)->sk_hash == (__hash))    && \
+#define INET_TW_MATCH(__sk, __net, __hash, __cookie, __saddr, __daddr, __ports, __dif)\
+ (((__sk)->sk_hash == (__hash)) && ((__sk)->sk_net == (__net)) && \
   ((*((__addrpair *)&(inet_twsk(__sk)->tw_daddr))) == (__cookie)) && \
   ((*((__portpair *)&(inet_twsk(__sk)->tw_dport))) == (__ports)) && \
   (!((__sk)->sk_bound_dev_if) || ((__sk)->sk_bound_dev_if == (__dif))))
 #else /* 32-bit arch */
 #define INET_ADDR_COOKIE(__name, __saddr, __daddr)
-#define INET_MATCH(__sk, __hash, __cookie, __saddr, __daddr, __ports, __dif) \
- (((__sk)->sk_hash == (__hash))    && \
+#define INET_MATCH(__sk, __net, __hash, __cookie, __saddr, __daddr, __ports, __dif) \
+ (((__sk)->sk_hash == (__hash)) && ((__sk)->sk_net == (__net)) && \
   (inet_sk(__sk)->daddr  == (__saddr))  && \
   (inet_sk(__sk)->rcv_saddr == (__daddr))  && \
   ((*((__portpair *)&(inet_sk(__sk)->dport))) == (__ports)) && \
   (!((__sk)->sk_bound_dev_if) || ((__sk)->sk_bound_dev_if == (__dif))))
-#define INET_TW_MATCH(__sk, __hash,__cookie, __saddr, __daddr, __ports, __dif) \
- (((__sk)->sk_hash == (__hash))    && \
+#define INET_TW_MATCH(__sk, __net, __hash,__cookie, __saddr, __daddr, __ports, __dif) \
+ (((__sk)->sk_hash == (__hash)) && ((__sk)->sk_net == (__net)) && \
   (inet_twsk(__sk)->tw_daddr == (__saddr))  && \
   (inet_twsk(__sk)->tw_rcv_saddr == (__daddr))  && \
   ((*((__portpair *)&(inet_twsk(__sk)->tw_dport))) == (__ports)) && \
@@ -376,32 +378,36 @@ typedef __u64 __bitwise __addrpair;
  *
  * Local BH must be disabled here.
  */
-extern struct sock * __inet_lookup_established(struct inet_hashinfo *hashinfo,
+extern struct sock * __inet_lookup_established(struct net *net,
+  struct inet_hashinfo *hashinfo,
   const __be32 saddr, const __be16 sport,
   const __be32 daddr, const u16 hnum, const int dif);

 static inline struct sock *
- inet_lookup_established(struct inet_hashinfo *hashinfo,
+ inet_lookup_established(struct net *net, struct inet_hashinfo *hashinfo,
     const __be32 saddr, const __be16 sport,
     const __be32 daddr, const __be16 dport,
```

```
     const int dif)
 {
- return __inet_lookup_established(hashinfo, saddr, sport, daddr,
+ return __inet_lookup_established(net, hashinfo, saddr, sport, daddr,
       ntohs(dport), dif);
 }

-static inline struct sock *__inet_lookup(struct inet_hashinfo *hashinfo,
+static inline struct sock *__inet_lookup(struct net *net,
+      struct inet_hashinfo *hashinfo,
       const __be32 saddr, const __be16 sport,
       const __be32 daddr, const __be16 dport,
       const int dif)
 {
  u16 hnum = ntohs(dport);
- struct sock *sk = __inet_lookup_established(hashinfo, saddr, sport, daddr,
-          hnum, dif);
- return sk ? : __inet_lookup_listener(hashinfo, daddr, hnum, dif);
+ struct sock *sk = __inet_lookup_established(net, hashinfo,
+    saddr, sport, daddr, hnum, dif);
+
+ return sk ? : __inet_lookup_listener(net, hashinfo, daddr, hnum, dif);
 }

-static inline struct sock *inet_lookup(struct inet_hashinfo *hashinfo,
+static inline struct sock *inet_lookup(struct net *net,
+           struct inet_hashinfo *hashinfo,
           const __be32 saddr, const __be16 sport,
           const __be32 daddr, const __be16 dport,
           const int dif)
@@ -409,7 +415,7 @@ static inline struct sock *inet_lookup(struct inet_hashinfo *hashinfo,
  struct sock *sk;

  local_bh_disable();
- sk = __inet_lookup(hashinfo, saddr, sport, daddr, dport, dif);
+ sk = __inet_lookup(net, hashinfo, saddr, sport, daddr, dport, dif);
  local_bh_enable();

  return sk;
diff --git a/net/dccp/ipv4.c b/net/dccp/ipv4.c
index 9e38b0d..c982ad8 100644
--- a/net/dccp/ipv4.c
+++ b/net/dccp/ipv4.c
@@ -218,7 +218,7 @@ static void dccp_v4_err(struct sk_buff *skb, u32 info)
   return;
  }

- sk = inet_lookup(&dccp_hashinfo, iph->daddr, dh->dccph_dport,
```

```
+ sk = inet_lookup(&init_net, &dccp_hashinfo, iph->daddr, dh->dccph_dport,
     iph->saddr, dh->dccph_sport, inet_iif(skb));
  if (sk == NULL) {
   ICMP_INC_STATS_BH(ICMP_MIB_INERRORS);
@@ -436,7 +436,7 @@ static struct sock *dccp_v4_hnd_req(struct sock *sk, struct sk_buff *skb)
  if (req != NULL)
   return dccp_check_req(sk, skb, req, prev);

- nsk = inet_lookup_established(&dccp_hashinfo,
+ nsk = inet_lookup_established(&init_net, &dccp_hashinfo,
           iph->saddr, dh->dccph_sport,
           iph->daddr, dh->dccph_dport,
           inet_iif(skb));
@@ -817,7 +817,7 @@ static int dccp_v4_rcv(struct sk_buff *skb)

  /* Step 2:
   * Look up flow ID in table and get corresponding socket */
- sk = __inet_lookup(&dccp_hashinfo,
+ sk = __inet_lookup(&init_net, &dccp_hashinfo,
       iph->saddr, dh->dccph_sport,
       iph->daddr, dh->dccph_dport, inet_iif(skb));
  /*
diff --git a/net/ipv4/inet_diag.c b/net/ipv4/inet_diag.c
index 4cfb15c..95c9f14 100644
--- a/net/ipv4/inet_diag.c
+++ b/net/ipv4/inet_diag.c
@@ -268,7 +268,7 @@ static int inet_diag_get_exact(struct sk_buff *in_skb,
  err = -EINVAL;

  if (req->idiag_family == AF_INET) {
-  sk = inet_lookup(hashinfo, req->id.idiag_dst[0],
+  sk = inet_lookup(&init_net, hashinfo, req->id.idiag_dst[0],
       req->id.idiag_dport, req->id.idiag_src[0],
       req->id.idiag_sport, req->id.idiag_if);
  }
diff --git a/net/ipv4/inet_hashtables.c b/net/ipv4/inet_hashtables.c
index db1e53a..48d4500 100644
--- a/net/ipv4/inet_hashtables.c
+++ b/net/ipv4/inet_hashtables.c
@@ -127,7 +127,8 @@ EXPORT_SYMBOL(inet_listen_wlock);
  * remote address for the connection. So always assume those are both
  * wildcarded during the search since they can never be otherwise.
  */
-static struct sock *inet_lookup_listener_slow(const struct hlist_head *head,
+static struct sock *inet_lookup_listener_slow(struct net *net,
+           const struct hlist_head *head,
           const __be32 daddr,
           const unsigned short hnum,
```

```
          const int dif)
@@ -139,7 +140,8 @@ static struct sock *inet_lookup_listener_slow(const struct hlist_head
*head,
  sk_for_each(sk, node, head) {
   const struct inet_sock *inet = inet_sk(sk);

-  if (inet->num == hnum && !ipv6_only_sock(sk)) {
+  if (sk->sk_net == net && inet->num == hnum &&
+      !ipv6_only_sock(sk)) {
     const __be32 rcv_saddr = inet->rcv_saddr;
     int score = sk->sk_family == PF_INET ? 1 : 0;

@@ -165,7 +167,8 @@ static struct sock *inet_lookup_listener_slow(const struct hlist_head
*head,
 }

 /* Optimize the common listener case. */
-struct sock *__inet_lookup_listener(struct inet_hashinfo *hashinfo,
+struct sock *__inet_lookup_listener(struct net *net,
+        struct inet_hashinfo *hashinfo,
         const __be32 daddr, const unsigned short hnum,
         const int dif)
 {
@@ -180,9 +183,9 @@ struct sock *__inet_lookup_listener(struct inet_hashinfo *hashinfo,
   if (inet->num == hnum && !sk->sk_node.next &&
       (!inet->rcv_saddr || inet->rcv_saddr == daddr) &&
       (sk->sk_family == PF_INET || !ipv6_only_sock(sk)) &&
-      !sk->sk_bound_dev_if)
+      !sk->sk_bound_dev_if && sk->sk_net == net)
     goto sherry_cache;
-  sk = inet_lookup_listener_slow(head, daddr, hnum, dif);
+  sk = inet_lookup_listener_slow(net, head, daddr, hnum, dif);
  }
  if (sk) {
 sherry_cache:
@@ -193,7 +196,8 @@ sherry_cache:
 }
 EXPORT_SYMBOL_GPL(__inet_lookup_listener);

-struct sock * __inet_lookup_established(struct inet_hashinfo *hashinfo,
+struct sock * __inet_lookup_established(struct net *net,
+      struct inet_hashinfo *hashinfo,
       const __be32 saddr, const __be16 sport,
       const __be32 daddr, const u16 hnum,
       const int dif)
@@ -212,13 +216,15 @@ struct sock * __inet_lookup_established(struct inet_hashinfo
*hashinfo,
  prefetch(head->chain.first);
```

```
  read_lock(lock);
  sk_for_each(sk, node, &head->chain) {
-  if (INET_MATCH(sk, hash, acookie, saddr, daddr, ports, dif))
+  if (INET_MATCH(sk, net, hash, acookie,
+      saddr, daddr, ports, dif))
     goto hit; /* You sunk my battleship! */
  }

  /* Must check for a TIME_WAIT'er before going to listener hash. */
  sk_for_each(sk, node, &head->twchain) {
-  if (INET_TW_MATCH(sk, hash, acookie, saddr, daddr, ports, dif))
+  if (INET_TW_MATCH(sk, net, hash, acookie,
+      saddr, daddr, ports, dif))
     goto hit;
  }
  sk = NULL;
@@ -249,6 +255,7 @@ static int __inet_check_established(struct inet_timewait_death_row
*death_row,
  struct sock *sk2;
  const struct hlist_node *node;
  struct inet_timewait_sock *tw;
+ struct net *net = sk->sk_net;

  prefetch(head->chain.first);
  write_lock(lock);
@@ -257,7 +264,8 @@ static int __inet_check_established(struct inet_timewait_death_row
*death_row,
  sk_for_each(sk2, node, &head->twchain) {
   tw = inet_twsk(sk2);

-  if (INET_TW_MATCH(sk2, hash, acookie, saddr, daddr, ports, dif)) {
+  if (INET_TW_MATCH(sk2, net, hash, acookie,
+      saddr, daddr, ports, dif)) {
     if (twsk_unique(sk, sk2, twp))
      goto unique;
     else
@@ -268,7 +276,8 @@ static int __inet_check_established(struct inet_timewait_death_row
*death_row,

  /* And established part... */
  sk_for_each(sk2, node, &head->chain) {
-  if (INET_MATCH(sk2, hash, acookie, saddr, daddr, ports, dif))
+  if (INET_MATCH(sk2, net, hash, acookie,
+      saddr, daddr, ports, dif))
     goto not_unique;
  }

diff --git a/net/ipv4/tcp_ipv4.c b/net/ipv4/tcp_ipv4.c
```

```
index 9aea88b..77c1939 100644
--- a/net/ipv4/tcp_ipv4.c
+++ b/net/ipv4/tcp_ipv4.c
@@ -369,8 +369,8 @@ void tcp_v4_err(struct sk_buff *skb, u32 info)
   return;
  }

- sk = inet_lookup(&tcp_hashinfo, iph->daddr, th->dest, iph->saddr,
-    th->source, inet_iif(skb));
+ sk = inet_lookup(skb->dev->nd_net, &tcp_hashinfo, iph->daddr, th->dest,
+    iph->saddr, th->source, inet_iif(skb));
  if (!sk) {
   ICMP_INC_STATS_BH(ICMP_MIB_INERRORS);
   return;
@@ -1503,8 +1503,8 @@ static struct sock *tcp_v4_hnd_req(struct sock *sk, struct sk_buff *skb)
  if (req)
   return tcp_check_req(sk, skb, req, prev);

- nsk = inet_lookup_established(&tcp_hashinfo, iph->saddr, th->source,
-          iph->daddr, th->dest, inet_iif(skb));
+ nsk = inet_lookup_established(sk->sk_net, &tcp_hashinfo, iph->saddr,
+    th->source, iph->daddr, th->dest, inet_iif(skb));

  if (nsk) {
   if (nsk->sk_state != TCP_TIME_WAIT) {
@@ -1661,8 +1661,8 @@ int tcp_v4_rcv(struct sk_buff *skb)
  TCP_SKB_CB(skb)->flags  = iph->tos;
  TCP_SKB_CB(skb)->sacked  = 0;

- sk = __inet_lookup(&tcp_hashinfo, iph->saddr, th->source,
-       iph->daddr, th->dest, inet_iif(skb));
+ sk = __inet_lookup(skb->dev->nd_net, &tcp_hashinfo, iph->saddr,
+    th->source, iph->daddr, th->dest, inet_iif(skb));
  if (!sk)
   goto no_tcp_socket;

@@ -1735,7 +1735,8 @@ do_time_wait:
  }
  switch (tcp_timewait_state_process(inet_twsk(sk), skb, th)) {
  case TCP_TW_SYN: {
-  struct sock *sk2 = inet_lookup_listener(&tcp_hashinfo,
+  struct sock *sk2 = inet_lookup_listener(skb->dev->nd_net,
+         &tcp_hashinfo,
          iph->daddr, th->dest,
          inet_iif(skb));
   if (sk2) {
--
1.5.3.4
```

Subject: Re: [PATCH 4/6][NETNS]: Tcp-v4 sockets per-net lookup.
Posted by davem on Thu, 31 Jan 2008 13:06:51 GMT
View Forum Message <> Reply to Message

From: Pavel Emelyanov <xemul@openvz.org>
Date: Thu, 31 Jan 2008 15:38:15 +0300

> Add a net argument to inet_lookup and propagate it further
> into lookup calls. Plus tune the __inet_check_established.
>
> The dccp and inet_diag, which use that lookup functions
> pass the init_net into them.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

Applied.