Subject: [patch 07/12] net namespace: set source addresse
Posted by Daniel Lezcano on Fri, 19 Jan 2007 15:47:21 GMT
View Forum Message <> Reply to Message

From: Daniel Lezcano <dlezcano@fr.ibm.com>

When no source address is specified, search from the dev list the
ifaddr allowed to be used as source address.

Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>

---
 include/linux/net_namespace.h | 14 ++++++++
 net/core/net_namespace.c | 68 ++
 net/ipv4/route.c | 28 ++++++++++------
 3 files changed, 100 insertions(+), 10 deletions(-)

Index: 2.6.20-rc4-mm1/net/ipv4/route.c
===
--- 2.6.20-rc4-mm1.orig/net/ipv4/route.c
+++ 2.6.20-rc4-mm1/net/ipv4/route.c
@@ -2475,17 +2475,17 @@

   if (LOCAL_MCAST(oldflp->fl4_dst) || oldflp->fl4_dst == htonl(0xFFFFFFFF)) {
    if (!fl.fl4_src)
-    fl.fl4_src = inet_select_addr(dev_out, 0,
-              RT_SCOPE_LINK);
+   fl.fl4_src = SELECT_SRC_ADDR(dev_out, 0,
+   RT_SCOPE_LINK);
    goto make_route;
   }
   if (!fl.fl4_src) {
    if (MULTICAST(oldflp->fl4_dst))
-    fl.fl4_src = inet_select_addr(dev_out, 0,
-              fl.fl4_scope);
+   fl.fl4_src = SELECT_SRC_ADDR(dev_out, 0,
+   fl.fl4_scope);
    else if (!oldflp->fl4_dst)
-    fl.fl4_src = inet_select_addr(dev_out, 0,
-              RT_SCOPE_HOST);
+   fl.fl4_src = SELECT_SRC_ADDR(dev_out, 0,
+   RT_SCOPE_HOST);
   }
  }

@@ -2525,8 +2525,8 @@
     */

```
    if (fl.fl4_src == 0)
-    fl.fl4_src = inet_select_addr(dev_out, 0,
-            RT_SCOPE_LINK);
+    fl.fl4_src = SELECT_SRC_ADDR(dev_out, 0,
+            RT_SCOPE_LINK);
     res.type = RTN_UNICAST;
     goto make_route;
    }
@@ -2539,7 +2539,13 @@

  if (res.type == RTN_LOCAL) {
   if (!fl.fl4_src)
+#ifdef CONFIG_NET_NS
+    fl.fl4_src = net_ns_select_source_address(dev_out,
+           fl.fl4_dst,
+           RT_SCOPE_LINK);
+#else
     fl.fl4_src = fl.fl4_dst;
+#endif
   if (dev_out)
    dev_put(dev_out);
   dev_out = &loopback_dev;
@@ -2561,8 +2567,10 @@
   fib_select_default(&fl, &res);

  if (!fl.fl4_src)
-  fl.fl4_src = FIB_RES_PREFSRC(res);
-
+  fl.fl4_src = res.fi->fib_prefsrc ? :
+    SELECT_SRC_ADDR(FIB_RES_DEV(res),
+      FIB_RES_GW(res),
+      res.scope);
  if (dev_out)
   dev_put(dev_out);
  dev_out = FIB_RES_DEV(res);
Index: 2.6.20-rc4-mm1/include/linux/net_namespace.h
===================================================================
--- 2.6.20-rc4-mm1.orig/include/linux/net_namespace.h
+++ 2.6.20-rc4-mm1/include/linux/net_namespace.h
@@ -5,6 +5,7 @@
 #include <linux/kref.h>
 #include <linux/nsproxy.h>
 #include <linux/errno.h>
+#include <linux/types.h>

 struct net_namespace {
  struct kref  kref;
@@ -95,6 +96,11 @@
```

```
 extern int net_ns_check_bind(int addr_type, u32 addr);

+extern __be32 net_ns_select_source_address(const struct net_device *dev,
+        u32 dst, int scope);
+
+#define SELECT_SRC_ADDR net_ns_select_source_address
+
 #else /* CONFIG_NET_NS */

 #define INIT_NET_NS(net_ns)
@@ -155,6 +161,14 @@
  return 0;
 }

+static inline __be32 net_ns_select_source_address(struct net_device *dev,
+        u32 dst, int scope)
+{
+ return 0;
+}
+
+#define SELECT_SRC_ADDR inet_select_addr
+
 #endif /* !CONFIG_NET_NS */

 #endif /* _LINUX_NET_NAMESPACE_H */
Index: 2.6.20-rc4-mm1/net/core/net_namespace.c
===================================================================
--- 2.6.20-rc4-mm1.orig/net/core/net_namespace.c
+++ 2.6.20-rc4-mm1/net/core/net_namespace.c
@@ -317,4 +317,72 @@
  return ret;
 }

+/*
+ * This function choose the source address from the network device,
+ * destination and the scope. The function will browse the ifaddr
+ * owned by network namespace and choose the most adapted for the
+ * dst address and dev.
+ * @dev : the network device where the traffic will go
+ * @dst : the destination address
+ * @scope : the scope of the dst address
+ * Returns: a source address
+ */
+__be32 net_ns_select_source_address(const struct net_device *dev,
+        u32 dst, int scope)
+{
+ __be32 addr = 0;
```

```
+ struct in_device *in_dev;
+ struct net_namespace *net_ns = current_net_ns;
+
+ if (LOOPBACK(dst))
+     return htonl(INADDR_LOOPBACK);
+
+ if (!dev)
+  goto no_dev;
+
+ rcu_read_lock();
+ in_dev = __in_dev_get_rcu(dev);
+ if (!in_dev)
+  goto no_in_dev;
+
+ for_ifa(in_dev) {
+  if (ifa->ifa_scope > scope)
+   continue;
+  if (ifa->ifa_net_ns != net_ns)
+   continue;
+  if (!dst || inet_ifa_match(dst, ifa)) {
+   addr = ifa->ifa_local;
+   break;
+  }
+  if (!addr)
+   addr = ifa->ifa_local;
+ } endfor_ifa(in_dev);
+no_in_dev:
+ rcu_read_unlock();
+
+ if (addr)
+  goto out;
+
+no_dev:
+ read_lock(&dev_base_lock);
+ rcu_read_lock();
+ for (dev = dev_base; dev; dev = dev->next) {
+  if ((in_dev = __in_dev_get_rcu(dev)) == NULL)
+   continue;
+
+  for_ifa(in_dev) {
+   if (ifa->ifa_scope != RT_SCOPE_LINK &&
+       ifa->ifa_scope <= scope &&
+       ifa->ifa_net_ns == net_ns) {
+    addr = ifa->ifa_local;
+    goto out_unlock_both;
+   }
+  } endfor_ifa(in_dev);
+ }
```

```
+out_unlock_both:
+ read_unlock(&dev_base_lock);
+ rcu_read_unlock();
+out:
+ return addr;
+}
 #endif /* CONFIG_NET_NS */
```

--