Subject: [patch 0/2][NETNS][RFD] moving fl_net to dst_entry
Posted by Daniel Lezcano on Tue, 11 Dec 2007 13:12:31 GMT
View Forum Message <> Reply to Message

Actually the fl_net field from the struct flowi is not acceptable.

The patchset is a example in how we can use the dst_entry/rtable with flowi and how we can retrieve netns with container_of. The interesting part is we can use flowi to retrieve the netns without modifying the structure itself. The drawback is we have to ensure in the network code flowi is not called without being encaspulated in a rtable structure. As far as I see there are a few places where we use flowi directly coming from a local variable in a function.

If this approach is acceptable, I can move forward and post something for ipv4.

--

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

Subject: [patch 1/2][NETNS][RFD] store the network namespace pointer in the dst_entry structure
Posted by Daniel Lezcano on Tue, 11 Dec 2007 13:12:32 GMT
View Forum Message <> Reply to Message

Store the network namespace pointer in the dst_entry structure when it is allocated.
The different protocols redefine the route object as a derivate object from dst_entry. So using the dst_entry to store the network namespace pointer will allow to take into account the ipv4, ipv6, dccp protocols in one shot through the different route objects, rtable, rt6_info, ...

---
 include/net/dst.h | 3 ++-
 net/core/dst.c | 3 ++-
 net/decnet/dn_route.c | 4 ++--
 net/ipv4/route.c | 14 +++++++-------
 net/ipv6/route.c | 18 ++++++++++--------
 net/xfrm/xfrm_policy.c | 2 +-
 6 files changed, 24 insertions(+), 20 deletions(-)

Index: linux-2.6-netns/include/net/dst.h
===
--- linux-2.6-netns.orig/include/net/dst.h
+++ linux-2.6-netns/include/net/dst.h
@@ -81,6 +81,7 @@ struct dst_entry

```
   struct dn_route  *dn_next;
  };
  char   info[0];
+ struct net  *net;
 };


@@ -181,7 +182,7 @@ static inline struct dst_entry *dst_pop(
 }

 extern int dst_discard(struct sk_buff *skb);
-extern void * dst_alloc(struct dst_ops * ops);
+extern void * dst_alloc(struct dst_ops * ops, struct net *net);
 extern void __dst_free(struct dst_entry * dst);
 extern struct dst_entry *dst_destroy(struct dst_entry * dst);

Index: linux-2.6-netns/net/core/dst.c
===================================================================
--- linux-2.6-netns.orig/net/core/dst.c
+++ linux-2.6-netns/net/core/dst.c
@@ -160,7 +160,7 @@ int dst_discard(struct sk_buff *skb)
 }
 EXPORT_SYMBOL(dst_discard);

-void * dst_alloc(struct dst_ops * ops)
+void * dst_alloc(struct dst_ops * ops, struct net *net)
 {
  struct dst_entry * dst;

@@ -176,6 +176,7 @@ void * dst_alloc(struct dst_ops * ops)
  dst->lastuse = jiffies;
  dst->path = dst;
  dst->input = dst->output = dst_discard;
+ dst->net = net;
 #if RT_CACHE_DEBUG >= 2
  atomic_inc(&dst_total);
 #endif
Index: linux-2.6-netns/net/decnet/dn_route.c
===================================================================
--- linux-2.6-netns.orig/net/decnet/dn_route.c
+++ linux-2.6-netns/net/decnet/dn_route.c
@@ -1086,7 +1086,7 @@ make_route:
  if (dev_out->flags & IFF_LOOPBACK)
   flags |= RTCF_LOCAL;

- rt = dst_alloc(&dn_dst_ops);
+ rt = dst_alloc(&dn_dst_ops, &init_net);
  if (rt == NULL)
```

```
   goto e_nobufs;

@@ -1350,7 +1350,7 @@ static int dn_route_input_slow(struct sk
  }

 make_route:
- rt = dst_alloc(&dn_dst_ops);
+ rt = dst_alloc(&dn_dst_ops, &init_net);
  if (rt == NULL)
   goto e_nobufs;

Index: linux-2.6-netns/net/ipv4/route.c
===================================================================
--- linux-2.6-netns.orig/net/ipv4/route.c
+++ linux-2.6-netns/net/ipv4/route.c
@@ -1115,7 +1115,7 @@ void ip_rt_redirect(__be32 old_gw, __be3
     dst_hold(&rth->u.dst);
     rcu_read_unlock();

-    rt = dst_alloc(&ipv4_dst_ops);
+    rt = dst_alloc(&ipv4_dst_ops, dev->nd_net);
     if (rt == NULL) {
      ip_rt_put(rth);
      in_dev_put(in_dev);
@@ -1565,7 +1565,7 @@ static int ip_route_input_mc(struct sk_b
      dev, &spec_dst, &itag) < 0)
   goto e_inval;

- rth = dst_alloc(&ipv4_dst_ops);
+ rth = dst_alloc(&ipv4_dst_ops, dev->nd_net);
  if (!rth)
   goto e_nobufs;

@@ -1704,7 +1704,7 @@ static inline int __mkroute_input(struct
  }


- rth = dst_alloc(&ipv4_dst_ops);
+ rth = dst_alloc(&ipv4_dst_ops, in_dev->dev->nd_net);
  if (!rth) {
   err = -ENOBUFS;
   goto cleanup;
@@ -1888,7 +1888,7 @@ brd_input:
  RT_CACHE_STAT_INC(in_brd);

 local_input:
- rth = dst_alloc(&ipv4_dst_ops);
+ rth = dst_alloc(&ipv4_dst_ops, net);
```

```
  if (!rth)
   goto e_nobufs;

@@ -2079,7 +2079,7 @@ static inline int __mkroute_output(struc
  }


- rth = dst_alloc(&ipv4_dst_ops);
+ rth = dst_alloc(&ipv4_dst_ops, dev_out->nd_net);
  if (!rth) {
   err = -ENOBUFS;
   goto cleanup;
@@ -2413,9 +2413,9 @@ static struct dst_ops ipv4_dst_blackhole
 static int ipv4_dst_blackhole(struct rtable **rp, struct flowi *flp, struct sock *sk)
 {
  struct rtable *ort = *rp;
- struct rtable *rt = (struct rtable *)
-  dst_alloc(&ipv4_dst_blackhole_ops);
+ struct rtable *rt;

+ rt = (struct rtable *) dst_alloc(&ipv4_dst_blackhole_ops, &init_net);
  if (rt) {
   struct dst_entry *new = &rt->u.dst;

Index: linux-2.6-netns/net/ipv6/route.c
===================================================================
--- linux-2.6-netns.orig/net/ipv6/route.c
+++ linux-2.6-netns/net/ipv6/route.c
@@ -195,9 +195,9 @@ struct rt6_info ip6_blk_hole_entry = {
 #endif

 /* allocate dst with ip6_dst_ops */
-static __inline__ struct rt6_info *ip6_dst_alloc(void)
+static __inline__ struct rt6_info *ip6_dst_alloc(struct net *net)
 {
- return (struct rt6_info *)dst_alloc(&ip6_dst_ops);
+ return (struct rt6_info *)dst_alloc(&ip6_dst_ops, net);
 }

 static void ip6_dst_destroy(struct dst_entry *dst)
@@ -790,10 +790,11 @@ EXPORT_SYMBOL(ip6_route_output);
 int ip6_dst_blackhole(struct sock *sk, struct dst_entry **dstp, struct flowi *fl)
 {
  struct rt6_info *ort = (struct rt6_info *) *dstp;
- struct rt6_info *rt = (struct rt6_info *)
-  dst_alloc(&ip6_dst_blackhole_ops);
  struct dst_entry *new = NULL;
+ struct rt6_info *rt;
```

```
+ struct net *net = &init_net;

+ rt = (struct rt6_info *) dst_alloc(&ip6_dst_blackhole_ops, net);
  if (rt) {
   new = &rt->u.dst;

@@ -923,7 +924,7 @@ struct dst_entry *ndisc_dst_alloc(struct
  if (unlikely(idev == NULL))
   return NULL;

- rt = ip6_dst_alloc();
+ rt = ip6_dst_alloc(dev->nd_net);
  if (unlikely(rt == NULL)) {
   in6_dev_put(idev);
   goto out;
@@ -1054,6 +1055,7 @@ int ip6_route_add(struct fib6_config *cf
  struct net_device *dev = NULL;
  struct inet6_dev *idev = NULL;
  struct fib6_table *table;
+ struct net *net = &init_net;
  int addr_type;

  if (cfg->fc_dst_len > 128 || cfg->fc_src_len > 128)
@@ -1081,7 +1083,7 @@ int ip6_route_add(struct fib6_config *cf
   goto out;
  }

- rt = ip6_dst_alloc();
+ rt = ip6_dst_alloc(net);

  if (rt == NULL) {
   err = -ENOMEM;
@@ -1560,7 +1562,7 @@ out:

 static struct rt6_info * ip6_rt_copy(struct rt6_info *ort)
 {
- struct rt6_info *rt = ip6_dst_alloc();
+ struct rt6_info *rt = ip6_dst_alloc(ort->u.dst.net);

  if (rt) {
   rt->u.dst.input = ort->u.dst.input;
@@ -1828,7 +1830,7 @@ struct rt6_info *addrconf_dst_alloc(stru
         const struct in6_addr *addr,
         int anycast)
 {
- struct rt6_info *rt = ip6_dst_alloc();
+ struct rt6_info *rt = ip6_dst_alloc(idev->dev->nd_net);
```

```
  if (rt == NULL)
   return ERR_PTR(-ENOMEM);
Index: linux-2.6-netns/net/xfrm/xfrm_policy.c
===================================================================
--- linux-2.6-netns.orig/net/xfrm/xfrm_policy.c
+++ linux-2.6-netns/net/xfrm/xfrm_policy.c
@@ -1259,7 +1259,7 @@ static inline struct xfrm_dst *xfrm_allo
  if (!afinfo)
   return ERR_PTR(-EINVAL);

- xdst = dst_alloc(afinfo->dst_ops) ?: ERR_PTR(-ENOBUFS);
+ xdst = dst_alloc(afinfo->dst_ops, &init_net) ?: ERR_PTR(-ENOBUFS);

  xfrm_policy_put_afinfo(afinfo);


--
```

Subject: [patch 2/2][NETNS][RFD] use dst_entries to retrieve the network namespace pointer

Posted by Daniel Lezcano on Tue, 11 Dec 2007 13:12:33 GMT

View Forum Message <> Reply to Message

The objective we have is to remove the fl_net field from the struct flowi.

In the previous patch, we make the network namespace to go up to the dst_entry.
This patch makes an example in how we can use the dst_entry/rtable combined with
a container_of to retrieve the network namespace when we have the flowi parameter.

One restriction is we should pass always a flowi parameter coming from a struct
dst_entry.

```
---
 include/net/ip_fib.h |    3 -
 net/ipv4/route.c     |  111 ++++++++++++++++++++++++++++++++++++++-----------
 2 files changed, 60 insertions(+), 54 deletions(-)

Index: linux-2.6-netns/net/ipv4/route.c
===================================================================
--- linux-2.6-netns.orig/net/ipv4/route.c
+++ linux-2.6-netns/net/ipv4/route.c
@@ -1788,15 +1788,18 @@ static int ip_route_input_slow(struct sk
  struct net *net = dev->nd_net;
```

```
  struct fib_result res;
  struct in_device *in_dev = in_dev_get(dev);
- struct flowi fl = { .fl_net = net,
-       .nl_u = { .ip4_u =
-         { .daddr = daddr,
-     .saddr = saddr,
-     .tos = tos,
-     .scope = RT_SCOPE_UNIVERSE,
-         } },
-       .mark = skb->mark,
-       .iif = dev->ifindex };
+ struct rtable rt = {
+  .u.dst.net = net,
+  .fl = { .fl_net = net,
+   .nl_u = { .ip4_u =
+     { .daddr = daddr,
+       .saddr = saddr,
+       .tos = tos,
+       .scope = RT_SCOPE_UNIVERSE,
+     } },
+   .mark = skb->mark,
+   .iif = dev->ifindex },
+ };
  unsigned flags = 0;
  u32  itag = 0;
  struct rtable * rth;
@@ -1832,7 +1835,7 @@ static int ip_route_input_slow(struct sk
  /*
   * Now we are ready to route packet.
   */
- if ((err = fib_lookup(&fl, &res)) != 0) {
+ if ((err = fib_lookup(&rt.fl, &res)) != 0) {
   if (!IN_DEV_FORWARD(in_dev))
    goto e_hostunreach;
   goto no_route;
@@ -1862,7 +1865,7 @@ static int ip_route_input_slow(struct sk
  if (res.type != RTN_UNICAST)
   goto martian_destination;

- err = ip_mkroute_input(skb, &res, &fl, in_dev, daddr, saddr, tos);
+ err = ip_mkroute_input(skb, &res, &rt.fl, in_dev, daddr, saddr, tos);
 done:
  in_dev_put(in_dev);
  if (free_res)
@@ -1898,7 +1901,6 @@ local_input:
  rth->u.dst.flags= DST_HOST;
  if (IN_DEV_CONF_GET(in_dev, NOPOLICY))
   rth->u.dst.flags |= DST_NOPOLICY;
```

```
- rth->fl.fl_net = net;
  rth->fl.fl4_dst = daddr;
  rth->rt_dst = daddr;
  rth->fl.fl4_tos = tos;
@@ -1923,7 +1925,7 @@ local_input:
   rth->rt_flags  &= ~RTCF_LOCAL;
  }
  rth->rt_type = res.type;
- hash = rt_hash(daddr, saddr, fl.iif);
+ hash = rt_hash(daddr, saddr, rt.fl.iif);
  err = rt_intern_hash(hash, rth, (struct rtable**)&skb->dst);
  goto done;

@@ -2173,18 +2175,21 @@ static int ip_route_output_slow(struct r
 {
  u32 tos = RT_FL_TOS(oldflp);
  struct net *net = oldflp->fl_net;
- struct flowi fl = { .fl_net = net,
-       .nl_u = { .ip4_u =
-         { .daddr = oldflp->fl4_dst,
-     .saddr = oldflp->fl4_src,
-     .tos = tos & IPTOS_RT_MASK,
-     .scope = ((tos & RTO_ONLINK) ?
-        RT_SCOPE_LINK :
-        RT_SCOPE_UNIVERSE),
-         } },
-       .mark = oldflp->mark,
-       .iif = net->loopback_dev->ifindex,
-       .oif = oldflp->oif };
+ struct rtable rt = {
+  .u.dst.net = net,
+  .fl = { .fl_net = net,
+    .nl_u = { .ip4_u =
+      { .daddr = oldflp->fl4_dst,
+        .saddr = oldflp->fl4_src,
+        .tos = tos & IPTOS_RT_MASK,
+        .scope = ((tos & RTO_ONLINK) ?
+          RT_SCOPE_LINK :
+          RT_SCOPE_UNIVERSE),
+      } },
+    .mark = oldflp->mark,
+    .iif = net->loopback_dev->ifindex,
+    .oif = oldflp->oif },
+ };
  struct fib_result res;
  unsigned flags = 0;
  struct net_device *dev_out = NULL;
@@ -2234,7 +2239,7 @@ static int ip_route_output_slow(struct r
```

```
      Luckily, this hack is good workaround.
     */

-   fl.oif = dev_out->ifindex;
+   rt.fl.oif = dev_out->ifindex;
    goto make_route;
   }
   if (dev_out)
@@ -2256,36 +2261,36 @@ static int ip_route_output_slow(struct r
   }

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

- if (!fl.fl4_dst) {
-  fl.fl4_dst = fl.fl4_src;
-  if (!fl.fl4_dst)
-   fl.fl4_dst = fl.fl4_src = htonl(INADDR_LOOPBACK);
+ if (!rt.fl.fl4_dst) {
+  rt.fl.fl4_dst = rt.fl.fl4_src;
+  if (!rt.fl.fl4_dst)
+   rt.fl.fl4_dst = rt.fl.fl4_src = htonl(INADDR_LOOPBACK);
   if (dev_out)
    dev_put(dev_out);
   dev_out = net->loopback_dev;
   dev_hold(dev_out);
-  fl.oif = net->loopback_dev->ifindex;
```

```
+  rt.fl.oif = net->loopback_dev->ifindex;
   res.type = RTN_LOCAL;
   flags |= RTCF_LOCAL;
   goto make_route;
  }

- if (fib_lookup(&fl, &res)) {
+ if (fib_lookup(&rt.fl, &res)) {
   res.fi = NULL;
   if (oldflp->oif) {
    /* Apparently, routing tables are wrong. Assume,
@@ -2306,9 +2311,9 @@ static int ip_route_output_slow(struct r
       likely IPv6, but we do not.
     */

-   if (fl.fl4_src == 0)
-    fl.fl4_src = inet_select_addr(dev_out, 0,
-           RT_SCOPE_LINK);
+   if (rt.fl.fl4_src == 0)
+    rt.fl.fl4_src = inet_select_addr(dev_out, 0,
+         RT_SCOPE_LINK);
    res.type = RTN_UNICAST;
    goto make_route;
   }
@@ -2320,13 +2325,13 @@ static int ip_route_output_slow(struct r
  free_res = 1;

  if (res.type == RTN_LOCAL) {
-  if (!fl.fl4_src)
-   fl.fl4_src = fl.fl4_dst;
+  if (!rt.fl.fl4_src)
+   rt.fl.fl4_src = rt.fl.fl4_dst;
   if (dev_out)
    dev_put(dev_out);
   dev_out = net->loopback_dev;
   dev_hold(dev_out);
-  fl.oif = dev_out->ifindex;
+  rt.fl.oif = dev_out->ifindex;
   if (res.fi)
    fib_info_put(res.fi);
   res.fi = NULL;
@@ -2336,24 +2341,24 @@ static int ip_route_output_slow(struct r

 #ifdef CONFIG_IP_ROUTE_MULTIPATH
  if (res.fi->fib_nhs > 1 && fl.oif == 0)
-  fib_select_multipath(&fl, &res);
+  fib_select_multipath(&rt.fl, &res);
  else
```

```
 #endif
- if (!res.prefixlen && res.type == RTN_UNICAST && !fl.oif)
-  fib_select_default(&fl, &res);
+ if (!res.prefixlen && res.type == RTN_UNICAST && !rt.fl.oif)
+  fib_select_default(&rt.fl, &res);

- if (!fl.fl4_src)
-  fl.fl4_src = FIB_RES_PREFSRC(res);
+ if (!rt.fl.fl4_src)
+  rt.fl.fl4_src = FIB_RES_PREFSRC(res);

  if (dev_out)
   dev_put(dev_out);
  dev_out = FIB_RES_DEV(res);
  dev_hold(dev_out);
- fl.oif = dev_out->ifindex;
+ rt.fl.oif = dev_out->ifindex;


 make_route:
- err = ip_mkroute_output(rp, &res, &fl, oldflp, dev_out, flags);
+ err = ip_mkroute_output(rp, &res, &rt.fl, oldflp, dev_out, flags);


  if (free_res)
Index: linux-2.6-netns/include/net/ip_fib.h
===================================================================
--- linux-2.6-netns.orig/include/net/ip_fib.h
+++ linux-2.6-netns/include/net/ip_fib.h
@@ -173,7 +173,8 @@ static inline struct fib_table *fib_new_

 static inline int fib_lookup(const struct flowi *flp, struct fib_result *res)
 {
- struct net *net = flp->fl_net;
+ struct rtable *rt = container_of(flp, struct rtable, fl);
+ struct net *net = rt->u.dst.net;
  struct fib_table *local_table = net->ip_fib_local_table;
  struct fib_table *main_table = net->ip_fib_main_table;
  if (local_table->tb_lookup(local_table, flp, res) &&

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

Subject: Re: [patch 1/2][NETNS][RFD] store the network namespace pointer in the dst_entry structure
Posted by ebiederm on Tue, 11 Dec 2007 15:52:24 GMT
View Forum Message <> Reply to Message

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

> Store the network namespace pointer in the dst_entry structure when it is
> allocated.
> The different protocols redefine the route object as a derivate object from
> dst_entry. So using the dst_entry to store the network namespace pointer will
> allow to take into account the ipv4, ipv6, dccp protocols in one shot through
> the different route objects, rtable, rt6_info, ...
>
> ---
> include/net/dst.h | 3 ++-
> net/core/dst.c | 3 ++-
> net/decnet/dn_route.c | 4 ++--
> net/ipv4/route.c | 14 +++++++-------
> net/ipv6/route.c | 18 ++++++++++--------
> net/xfrm/xfrm_policy.c | 2 +-
> 6 files changed, 24 insertions(+), 20 deletions(-)
>
> Index: linux-2.6-netns/include/net/dst.h
> ===
> --- linux-2.6-netns.orig/include/net/dst.h
> +++ linux-2.6-netns/include/net/dst.h
> @@ -81,6 +81,7 @@ struct dst_entry
> struct dn_route *dn_next;
> };
> char info[0];
> + struct net *net;

Unless I'm missing something you just place that net pointer in
the middle of a variable length array. Weird I don't see us
using that array.

Could you please place the struct net *net pointer up by the
network device pointer.
> };

I know we need a net pointer in struct rt_table, because it
is a hash table that we can't dynamically allocate so we need
to place a network namespace pointer as part of the hash key.

For the ipv6 fib tables I don't recall needing a net pointer
as we didn't have a hash table and could instead have separate
roots for different namespaces.

I find this slightly odd as I didn't wind up needing to add
a struct net pointer in struct dst in my proof of concept tree
and struct dst doesn't have a struct flowi so that would not
have prevented it.

Eric

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

---

Subject: Re: [patch 1/2][NETNS][RFD] store the network namespace pointer in the dst_entry structure
Posted by Daniel Lezcano on Tue, 11 Dec 2007 16:14:47 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> Daniel Lezcano <dlezcano@fr.ibm.com> writes:
>
>> Store the network namespace pointer in the dst_entry structure when it is
>> allocated.
>> The different protocols redefine the route object as a derivate object from
>> dst_entry. So using the dst_entry to store the network namespace pointer will
>> allow to take into account the ipv4, ipv6, dccp protocols in one shot through
>> the different route objects, rtable, rt6_info, ...
>>
>> ---
>> include/net/dst.h | 3 ++-
>> net/core/dst.c | 3 ++-
>> net/decnet/dn_route.c | 4 ++--
>> net/ipv4/route.c | 14 +++++++-------
>> net/ipv6/route.c | 18 ++++++++++--------
>> net/xfrm/xfrm_policy.c | 2 +-
>> 6 files changed, 24 insertions(+), 20 deletions(-)
>>
>> Index: linux-2.6-netns/include/net/dst.h
>> ===
>> --- linux-2.6-netns.orig/include/net/dst.h
>> +++ linux-2.6-netns/include/net/dst.h
>> @@ -81,6 +81,7 @@ struct dst_entry
>>   struct dn_route *dn_next;
>>   };

>>   char   info[0];
>> + struct net *net;
>
> Unless I'm missing something you just place that net pointer in
> the middle of a variable length array. Weird I don't see us
> using that array.

yep, right, thanks.

> Could you please place the struct net *net pointer up by the
> network device pointer.
>> };
>
> I know we need a net pointer in struct rt_table, because it
> is a hash table that we can't dynamically allocate so we need
> to place a network namespace pointer as part of the hash key.
>
> For the ipv6 fib tables I don't recall needing a net pointer
> as we didn't have a hash table and could instead have separate
> roots for different namespaces.

Yes don't need for the hash table but we used it to pass the network
namespace parameter to the underlying function which need the net parameter.

We are facing two problems when removing the fl_net field from flowi:

  * The first one is the fl_net is used as a key. This problem can be
handled simply in moving the netns to the rtable.

  * The second one is the usage made by the fl_net to pass through the
different function calls the network namespace pointer without changing
all functions signature. This problem can be solved if we put the netns
pointer in the dst_entry structure, so when we are in ipv4, we use
container_of on rtable and when we are in ipv6, we use the container_of
on rt6_info. So everywhere with the flowi, we can retrieve the netns.

Here is a example for ipv4:

```
static inline int fib_lookup(const struct flowi *flp, struct fib_result
*res)
{
 struct rtable *rt = container_of(flp, struct rtable, fl);
 struct net *net = rt->u.dst.net;
 struct fib_table *local_table = net->ip_fib_local_table;
 struct fib_table *main_table = net->ip_fib_main_table;
 if (local_table->tb_lookup(local_table, flp, res) &&
     main_table->tb_lookup(main_table, flp, res))
  return -ENETUNREACH;
```

```
 return 0;
}
```

Other one for ipv6:

```
static struct rt6_info *ip6_pol_route_lookup(struct fib6_table *table,
        struct flowi *fl, int flags)
{
 struct rt6_info *rt  = container_of(flp, struct rt6_info, fl);
 struct net *net = rt.u.dst.net;
 struct fib6_node *fn;
 struct rt6_info *rt;

 read_lock_bh(&table->tb6_lock);
 fn = fib6_lookup(&table->tb6_root, &fl->fl6_dst, &fl->fl6_src);
restart:
 rt = fn->leaf;
 rt = rt6_device_match(net, rt, fl->oif, flags);
 BACKTRACK(net, &fl->fl6_src);
out:
 dst_use(&rt->u.dst, jiffies);
 read_unlock_bh(&table->tb6_lock);
 return rt;

}
```

> I find this slightly odd as I didn't wind up needing to add
> a struct net pointer in struct dst in my proof of concept tree
> and struct dst doesn't have a struct flowi so that would not
> have prevented it.

The idea is to put the net in the dst_entry because it is accessible
from rtable or rt6_info and these ones contain a flowi field.

Subject: Re: [patch 2/2][NETNS][RFD] use dst_entries to retrieve the network namespace pointer
Posted by den on Tue, 11 Dec 2007 17:00:25 GMT
View Forum Message <> Reply to Message

Daniel Lezcano wrote:
> The objective we have is to remove the fl_net field from the struct flowi.

>
> In the previous patch, we make the network namespace to go up to the dst_entry.
> This patch makes an example in how we can use the dst_entry/rtable combined with
> a container_of to retrieve the network namespace when we have the flowi parameter.
>
> One restriction is we should pass always a flowi parameter coming from a struct
> dst_entry.
this is an obfuscation IMHO. You use rtable instead of flowi, so
- rtable obtain flowi meaning and keeps original meaning
- stack usage in increased

So, for the case, I think it is better to have an additional parameter
on IP route output path. (There is a device on the input one).

Regards,
 Den

>
> ---
> include/net/ip_fib.h | 3 -
> net/ipv4/route.c | 111 ++++++++++++++++++++++++++++++----------------------
> 2 files changed, 60 insertions(+), 54 deletions(-)
>
> Index: linux-2.6-netns/net/ipv4/route.c
> ===
> --- linux-2.6-netns.orig/net/ipv4/route.c
> +++ linux-2.6-netns/net/ipv4/route.c
> @@ -1788,15 +1788,18 @@ static int ip_route_input_slow(struct sk
>   struct net *net = dev->nd_net;
>   struct fib_result res;
>   struct in_device *in_dev = in_dev_get(dev);
> - struct flowi fl = { .fl_net = net,
> -   .nl_u = { .ip4_u =
> -   { .daddr = daddr,
> -   .saddr = saddr,
> -   .tos = tos,
> -   .scope = RT_SCOPE_UNIVERSE,
> -   } },
> -   .mark = skb->mark,
> -   .iif = dev->ifindex };
> + struct rtable rt = {
> + .u.dst.net = net,
> + .fl = { .fl_net = net,
> +   .nl_u = { .ip4_u =
> +   { .daddr = daddr,
> +   .saddr = saddr,
> +   .tos = tos,
> +   .scope = RT_SCOPE_UNIVERSE,

```
> +      } },
> +    .mark = skb->mark,
> +    .iif = dev->ifindex },
> + };
>    unsigned flags = 0;
>    u32  itag = 0;
>    struct rtable * rth;
> @@ -1832,7 +1835,7 @@ static int ip_route_input_slow(struct sk
>    /*
>     * Now we are ready to route packet.
>     */
> - if ((err = fib_lookup(&fl, &res)) != 0) {
> + if ((err = fib_lookup(&rt.fl, &res)) != 0) {
>     if (!IN_DEV_FORWARD(in_dev))
>      goto e_hostunreach;
>     goto no_route;
> @@ -1862,7 +1865,7 @@ static int ip_route_input_slow(struct sk
>    if (res.type != RTN_UNICAST)
>     goto martian_destination;
>
> - err = ip_mkroute_input(skb, &res, &fl, in_dev, daddr, saddr, tos);
> + err = ip_mkroute_input(skb, &res, &rt.fl, in_dev, daddr, saddr, tos);
>  done:
>    in_dev_put(in_dev);
>    if (free_res)
> @@ -1898,7 +1901,6 @@ local_input:
>    rth->u.dst.flags= DST_HOST;
>    if (IN_DEV_CONF_GET(in_dev, NOPOLICY))
>     rth->u.dst.flags |= DST_NOPOLICY;
> - rth->fl.fl_net = net;
>    rth->fl.fl4_dst = daddr;
>    rth->rt_dst = daddr;
>    rth->fl.fl4_tos = tos;
> @@ -1923,7 +1925,7 @@ local_input:
>     rth->rt_flags  &= ~RTCF_LOCAL;
>    }
>    rth->rt_type = res.type;
> - hash = rt_hash(daddr, saddr, fl.iif);
> + hash = rt_hash(daddr, saddr, rt.fl.iif);
>    err = rt_intern_hash(hash, rth, (struct rtable**)&skb->dst);
>    goto done;
>
> @@ -2173,18 +2175,21 @@ static int ip_route_output_slow(struct r
>  {
>    u32 tos = RT_FL_TOS(oldflp);
>    struct net *net = oldflp->fl_net;
> - struct flowi fl = { .fl_net = net,
> -       .nl_u = { .ip4_u =
```

```
> -         { .daddr = oldflp->fl4_dst,
> -     .saddr = oldflp->fl4_src,
> -     .tos = tos & IPTOS_RT_MASK,
> -     .scope = ((tos & RTO_ONLINK) ?
> -       RT_SCOPE_LINK :
> -       RT_SCOPE_UNIVERSE),
> -         } },
> -       .mark = oldflp->mark,
> -       .iif = net->loopback_dev->ifindex,
> -       .oif = oldflp->oif };
> + struct rtable rt = {
> +  .u.dst.net = net,
> +  .fl = { .fl_net = net,
> +    .nl_u = { .ip4_u =
> +      { .daddr = oldflp->fl4_dst,
> +        .saddr = oldflp->fl4_src,
> +        .tos = tos & IPTOS_RT_MASK,
> +        .scope = ((tos & RTO_ONLINK) ?
> +          RT_SCOPE_LINK :
> +          RT_SCOPE_UNIVERSE),
> +      } },
> +    .mark = oldflp->mark,
> +    .iif = net->loopback_dev->ifindex,
> +    .oif = oldflp->oif },
> + };
>   struct fib_result res;
>   unsigned flags = 0;
>   struct net_device *dev_out = NULL;
> @@ -2234,7 +2239,7 @@ static int ip_route_output_slow(struct r
>         Luckily, this hack is good workaround.
>       */
>
> -   fl.oif = dev_out->ifindex;
> +    rt.fl.oif = dev_out->ifindex;
>     goto make_route;
>    }
>    if (dev_out)
> @@ -2256,36 +2261,36 @@ static int ip_route_output_slow(struct r
>    }
>
>    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);
> +    if (!rt.fl.fl4_src)
> +     rt.fl.fl4_src = inet_select_addr(dev_out, 0,
> +         RT_SCOPE_LINK);
>     goto make_route;
```

```
>    }
> -  if (!fl.fl4_src) {
> +  if (!rt.fl.fl4_src) {
>     if (MULTICAST(oldflp->fl4_dst))
> -    fl.fl4_src = inet_select_addr(dev_out, 0,
> -               fl.fl4_scope);
> +    rt.fl.fl4_src = inet_select_addr(dev_out, 0,
> +           rt.fl.fl4_scope);
>     else if (!oldflp->fl4_dst)
> -    fl.fl4_src = inet_select_addr(dev_out, 0,
> -               RT_SCOPE_HOST);
> +    rt.fl.fl4_src = inet_select_addr(dev_out, 0,
> +           RT_SCOPE_HOST);
>    }
>   }
>
> - if (!fl.fl4_dst) {
> -  fl.fl4_dst = fl.fl4_src;
> -  if (!fl.fl4_dst)
> -   fl.fl4_dst = fl.fl4_src = htonl(INADDR_LOOPBACK);
> + if (!rt.fl.fl4_dst) {
> +  rt.fl.fl4_dst = rt.fl.fl4_src;
> +  if (!rt.fl.fl4_dst)
> +   rt.fl.fl4_dst = rt.fl.fl4_src = htonl(INADDR_LOOPBACK);
>    if (dev_out)
>     dev_put(dev_out);
>    dev_out = net->loopback_dev;
>    dev_hold(dev_out);
> -  fl.oif = net->loopback_dev->ifindex;
> +  rt.fl.oif = net->loopback_dev->ifindex;
>    res.type = RTN_LOCAL;
>    flags |= RTCF_LOCAL;
>    goto make_route;
>   }
>
> - if (fib_lookup(&fl, &res)) {
> + if (fib_lookup(&rt.fl, &res)) {
>    res.fi = NULL;
>    if (oldflp->oif) {
>     /* Apparently, routing tables are wrong. Assume,
> @@ -2306,9 +2311,9 @@ static int ip_route_output_slow(struct r
>        likely IPv6, but we do not.
>      */
>
> -   if (fl.fl4_src == 0)
> -    fl.fl4_src = inet_select_addr(dev_out, 0,
> -               RT_SCOPE_LINK);
> +   if (rt.fl.fl4_src == 0)
```

```
> +    rt.fl.fl4_src = inet_select_addr(dev_out, 0,
> +          RT_SCOPE_LINK);
>      res.type = RTN_UNICAST;
>      goto make_route;
>    }
> @@ -2320,13 +2325,13 @@ static int ip_route_output_slow(struct r
>    free_res = 1;
>
>    if (res.type == RTN_LOCAL) {
> -    if (!fl.fl4_src)
> -      fl.fl4_src = fl.fl4_dst;
> +    if (!rt.fl.fl4_src)
> +      rt.fl.fl4_src = rt.fl.fl4_dst;
>      if (dev_out)
>        dev_put(dev_out);
>      dev_out = net->loopback_dev;
>      dev_hold(dev_out);
> -    fl.oif = dev_out->ifindex;
> +    rt.fl.oif = dev_out->ifindex;
>      if (res.fi)
>        fib_info_put(res.fi);
>      res.fi = NULL;
> @@ -2336,24 +2341,24 @@ static int ip_route_output_slow(struct r
>
>    #ifdef CONFIG_IP_ROUTE_MULTIPATH
>    if (res.fi->fib_nhs > 1 && fl.oif == 0)
> -    fib_select_multipath(&fl, &res);
> +    fib_select_multipath(&rt.fl, &res);
>    else
>    #endif
> - if (!res.prefixlen && res.type == RTN_UNICAST && !fl.oif)
> -    fib_select_default(&fl, &res);
> + if (!res.prefixlen && res.type == RTN_UNICAST && !rt.fl.oif)
> +    fib_select_default(&rt.fl, &res);
>
> - if (!fl.fl4_src)
> -    fl.fl4_src = FIB_RES_PREFSRC(res);
> + if (!rt.fl.fl4_src)
> +    rt.fl.fl4_src = FIB_RES_PREFSRC(res);
>
>    if (dev_out)
>      dev_put(dev_out);
>    dev_out = FIB_RES_DEV(res);
>    dev_hold(dev_out);
> - fl.oif = dev_out->ifindex;
> + rt.fl.oif = dev_out->ifindex;
>
>
```

> make_route:
> - err = ip_mkroute_output(rp, &res, &fl, oldflp, dev_out, flags);
> + err = ip_mkroute_output(rp, &res, &rt.fl, oldflp, dev_out, flags);
>
>
>   if (free_res)
> Index: linux-2.6-netns/include/net/ip_fib.h
> ===
> --- linux-2.6-netns.orig/include/net/ip_fib.h
> +++ linux-2.6-netns/include/net/ip_fib.h
> @@ -173,7 +173,8 @@ static inline struct fib_table *fib_new_
>
> static inline int fib_lookup(const struct flowi *flp, struct fib_result *res)
> {
> - struct net *net = flp->fl_net;
> + struct rtable *rt = container_of(flp, struct rtable, fl);
> + struct net *net = rt->u.dst.net;
>   struct fib_table *local_table = net->ip_fib_local_table;
>   struct fib_table *main_table = net->ip_fib_main_table;
>   if (local_table->tb_lookup(local_table, flp, res) &&
>

Subject: Re: [patch 1/2][NETNS][RFD] store the network namespace pointer in the dst_entry structure
Posted by ebiederm on Tue, 11 Dec 2007 17:07:18 GMT
View Forum Message <> Reply to Message

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

>> Could you please place the struct net *net pointer up by the
>> network device pointer.
>>> };
>>
>> I know we need a net pointer in struct rt_table, because it
>> is a hash table that we can't dynamically allocate so we need
>> to place a network namespace pointer as part of the hash key.
>>
>> For the ipv6 fib tables I don't recall needing a net pointer as we didn't have
>> a hash table and could instead have separate
>> roots for different namespaces.
>
> Yes don't need for the hash table but we used it to pass the network namespace

> parameter to the underlying function which need the net parameter.
>
> We are facing two problems when removing the fl_net field from flowi:
>
> * The first one is the fl_net is used as a key. This problem can be handled
> simply in moving the netns to the rtable.

Yes.

> * The second one is the usage made by the fl_net to pass through the different
> function calls the network namespace pointer without changing all functions
> signature. This problem can be solved if we put the netns pointer in the
> dst_entry structure, so when we are in ipv4, we use container_of on rtable and
> when we are in ipv6, we use the container_of on rt6_info. So everywhere with the
> flowi, we can retrieve the netns.

That doesn't work as rt6_info does not currently hold a struct flowi.

>> I find this slightly odd as I didn't wind up needing to add
>> a struct net pointer in struct dst in my proof of concept tree
>> and struct dst doesn't have a struct flowi so that would not
>> have prevented it.
>
> The idea is to put the net in the dst_entry because it is accessible from rtable
> or rt6_info and these ones contain a flowi field.

And since that isn't true, the idea seems to fall flat on it's face.

I expect most of the instances of struct flowi that we would be
looking things up with would be on the stack so the earlier concerns
raised would likely still need to be addressed.

Eric

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

---

Subject: Re: [patch 1/2][NETNS][RFD] store the network namespace pointer in the dst_entry structure
Posted by Benjamin Thery on Tue, 11 Dec 2007 17:24:07 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> Daniel Lezcano <dlezcano@fr.ibm.com> writes:
>
>>> Could you please place the struct net *net pointer up by the

>>> network device pointer.
>>>> };
>>> I know we need a net pointer in struct rt_table, because it
>>> is a hash table that we can't dynamically allocate so we need
>>> to place a network namespace pointer as part of the hash key.
>>>
>>> For the ipv6 fib tables I don't recall needing a net pointer as we didn't have
>>> a hash table and could instead have separate
>>> roots for different namespaces.
>> Yes don't need for the hash table but we used it to pass the network namespace
>> parameter to the underlying function which need the net parameter.
>>
>> We are facing two problems when removing the fl_net field from flowi:
>>
>> * The first one is the fl_net is used as a key. This problem can be handled
>> simply in moving the netns to the rtable.
>
> Yes.
>
>> * The second one is the usage made by the fl_net to pass through the different
>> function calls the network namespace pointer without changing all functions
>> signature. This problem can be solved if we put the netns pointer in the
>> dst_entry structure, so when we are in ipv4, we use container_of on rtable and
>> when we are in ipv6, we use the container_of on rt6_info. So everywhere with the
>> flowi, we can retrieve the netns.
>
> That doesn't work as rt6_info does not currently hold a struct flowi.
>
>>> I find this slightly odd as I didn't wind up needing to add
>>> a struct net pointer in struct dst in my proof of concept tree
>>> and struct dst doesn't have a struct flowi so that would not
>>> have prevented it.
>> The idea is to put the net in the dst_entry because it is accessible from rtable
>> or rt6_info and these ones contain a flowi field.
>
> And since that isn't true, the idea seems to fall flat on it's face.

My fault.
While talking with Daniel last week I suggested to put the net in
dst_entry instead of rtable, because dst_entry was common to rtable
and rt6_info. I thought we could factorize some code this way. I
wrongly assumed IPv6 was pretty similar to IPv4 in the way it handles
flowi. I should have checked that more carefully. Crap ;)

Benjamin

>
> I expect most of the instances of struct flowi that we would be

> looking things up with would be on the stack so the earlier concerns
> raised would likely still need to be addressed.
>
> Eric
>

--
B e n j a m i n T h e r y - BULL/DT/Open Software R&D

http://www.bull.com

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

---

Subject: Re: [patch 2/2][NETNS][RFD] use dst_entries to retrieve the network namespace pointer
Posted by ebiederm on Tue, 11 Dec 2007 17:31:34 GMT
View Forum Message <> Reply to Message

"Denis V. Lunev" <den@sw.ru> writes:

> Daniel Lezcano wrote:
>> The objective we have is to remove the fl_net field from the struct flowi.
>>
>> In the previous patch, we make the network namespace to go up to the
> dst_entry.
>> This patch makes an example in how we can use the dst_entry/rtable combined
> with
>> a container_of to retrieve the network namespace when we have the flowi
> parameter.
>>
>> One restriction is we should pass always a flowi parameter coming from a
> struct
>> dst_entry.
> this is an obfuscation IMHO. You use rtable instead of flowi, so
> - rtable obtain flowi meaning and keeps original meaning
> - stack usage in increased
>
> So, for the case, I think it is better to have an additional parameter
> on IP route output path. (There is a device on the input one).

Sounds right, and for shared code as well.

The only reason I used flowi in my proof of concept implementation
is that it seemed to be a prebuilt structure for passing that kind of

thing so I could save a bit of pain by extending what already existed.

Eric

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

---

Subject: Re: [patch 1/2][NETNS][RFD] store the network namespace pointer in the dst_entry structure
Posted by Daniel Lezcano on Tue, 11 Dec 2007 17:36:14 GMT
View Forum Message <> Reply to Message

Benjamin Thery wrote:
> Eric W. Biederman wrote:
>> Daniel Lezcano <dlezcano@fr.ibm.com> writes:
>>
>>>> Could you please place the struct net *net pointer up by the
>>>> network device pointer.
>>>> };
>>>> I know we need a net pointer in struct rt_table, because it
>>>> is a hash table that we can't dynamically allocate so we need
>>>> to place a network namespace pointer as part of the hash key.
>>>>
>>>> For the ipv6 fib tables I don't recall needing a net pointer as we didn't have
>>>> a hash table and could instead have separate
>>>> roots for different namespaces.
>>> Yes don't need for the hash table but we used it to pass the network namespace
>>> parameter to the underlying function which need the net parameter.
>>>
>>> We are facing two problems when removing the fl_net field from flowi:
>>>
>>> * The first one is the fl_net is used as a key. This problem can be handled
>>> simply in moving the netns to the rtable.
>> Yes.
>>
>>> * The second one is the usage made by the fl_net to pass through the different
>>> function calls the network namespace pointer without changing all functions
>>> signature. This problem can be solved if we put the netns pointer in the
>>> dst_entry structure, so when we are in ipv4, we use container_of on rtable and
>>> when we are in ipv6, we use the container_of on rt6_info. So everywhere with the
>>> flowi, we can retrieve the netns.
>> That doesn't work as rt6_info does not currently hold a struct flowi.
>>
>>>> I find this slightly odd as I didn't wind up needing to add
>>>> a struct net pointer in struct dst in my proof of concept tree
>>>> and struct dst doesn't have a struct flowi so that would not

>>>> have prevented it.
>>> The idea is to put the net in the dst_entry because it is accessible from rtable
>>> or rt6_info and these ones contain a flowi field.
>> And since that isn't true, the idea seems to fall flat on it's face.
>
> My fault.
> While talking with Daniel last week I suggested to put the net in
> dst_entry instead of rtable, because dst_entry was common to rtable
> and rt6_info. I thought we could factorize some code this way. I
> wrongly assumed IPv6 was pretty similar to IPv4 in the way it handles
> flowi. I should have checked that more carefully. Crap ;)

Me too :)

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

---

Subject: Re: [patch 2/2][NETNS][RFD] use dst_entries to retrieve the network namespace pointer
Posted by Daniel Lezcano on Wed, 12 Dec 2007 09:03:17 GMT
View Forum Message <> Reply to Message

Denis V. Lunev wrote:
> Daniel Lezcano wrote:
>> The objective we have is to remove the fl_net field from the struct flowi.
>>
>> In the previous patch, we make the network namespace to go up to the dst_entry.
>> This patch makes an example in how we can use the dst_entry/rtable combined with
>> a container_of to retrieve the network namespace when we have the flowi parameter.
>>
>> One restriction is we should pass always a flowi parameter coming from a struct
>> dst_entry.
> this is an obfuscation IMHO. You use rtable instead of flowi, so
> - rtable obtain flowi meaning and keeps original meaning
> - stack usage in increased

Yes, you are right. Using container_of is not a good idea finally.

> So, for the case, I think it is better to have an additional parameter
> on IP route output path. (There is a device on the input one).

I thought that at the beginning, but I was not inclined to change
exported functions API, so I looked a way to avoid that but I failed
miserably :)
Obviously, the ip route output path extra parameter is the cleanest way
to introduce the network namespace pointer in the routing code. So why not ?

Subject: Re: [patch 2/2][NETNS][RFD] use dst_entries to retrieve the network namespace pointer
Posted by den on Wed, 12 Dec 2007 09:26:57 GMT
View Forum Message <> Reply to Message

Daniel Lezcano wrote:
> Denis V. Lunev wrote:
>> Daniel Lezcano wrote:
>>> The objective we have is to remove the fl_net field from the struct
>>> flowi.
>>>
>>> In the previous patch, we make the network namespace to go up to the
>>> dst_entry.
>>> This patch makes an example in how we can use the dst_entry/rtable
>>> combined with
>>> a container_of to retrieve the network namespace when we have the
>>> flowi parameter.
>>>
>>> One restriction is we should pass always a flowi parameter coming
>>> from a struct
>>> dst_entry.
>> this is an obfuscation IMHO. You use rtable instead of flowi, so
>> - rtable obtain flowi meaning and keeps original meaning
>> - stack usage in increased
>
> Yes, you are right. Using container_of is not a good idea finally.
>
>> So, for the case, I think it is better to have an additional parameter
>> on IP route output path. (There is a device on the input one).
>
> I thought that at the beginning, but I was not inclined to change
> exported functions API, so I looked a way to avoid that but I failed
> miserably :)
> Obviously, the ip route output path extra parameter is the cleanest way
> to introduce the network namespace pointer in the routing code. So why
> not ?
>

Daniel,

I am in somewhere near the end of routing back-end namespacing, so could
you wait a bit till I send it to Dave. I think this could be done just

after Pavel's set will be commited (netns_ipv4 is needed :)

Regards,
Den

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

---

Subject: Re: [patch 2/2][NETNS][RFD] use dst_entries to retrieve the network namespace pointer
Posted by Daniel Lezcano on Wed, 12 Dec 2007 11:18:39 GMT
View Forum Message <> Reply to Message

Denis V. Lunev wrote:
> Daniel Lezcano wrote:
>> Denis V. Lunev wrote:
>>> Daniel Lezcano wrote:
>>>> The objective we have is to remove the fl_net field from the struct
>>>> flowi.
>>>>
>>>> In the previous patch, we make the network namespace to go up to the
>>>> dst_entry.
>>>> This patch makes an example in how we can use the dst_entry/rtable
>>>> combined with
>>>> a container_of to retrieve the network namespace when we have the
>>>> flowi parameter.
>>>>
>>>> One restriction is we should pass always a flowi parameter coming
>>>> from a struct
>>>> dst_entry.
>>> this is an obfuscation IMHO. You use rtable instead of flowi, so
>>> - rtable obtain flowi meaning and keeps original meaning
>>> - stack usage in increased
>> Yes, you are right. Using container_of is not a good idea finally.
>>
>>> So, for the case, I think it is better to have an additional parameter
>>> on IP route output path. (There is a device on the input one).
>> I thought that at the beginning, but I was not inclined to change
>> exported functions API, so I looked a way to avoid that but I failed
>> miserably :)
>> Obviously, the ip route output path extra parameter is the cleanest way
>> to introduce the network namespace pointer in the routing code. So why
>> not ?
>>
>
> Daniel,

>
> I am in somewhere near the end of routing back-end namespacing, so could
> you wait a bit till I send it to Dave. I think this could be done just
> after Pavel's set will be commited (netns_ipv4 is needed :)
>
> Regards,
> Den

No problem.

Benjamin and I will change the ipv6 routing then.

Thanks.
 -- Daniel