Subject: Re: namespace acceptance process. bad news
Posted by ebiederm on Wed, 05 Dec 2007 22:21:30 GMT
View Forum Message <> Reply to Message

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

> Alexey Kuznetsov wrote:
>> Hello!
>>
>>> Alexey seems to disagree with this approach, is it possible to elaborate
>>> a little bit ?
>>
>> My first reaction was exactly the same as David's one. Exactly. :-)
>>
>> flowi structure was invented to be both easily initialized/disposed
>> as a local variable and copied/stored in various caches as a key.
>>
>> If it has some reference inside, it becomes really ugly.
>>
>> But it is the first reaction. I guess you do not have much of choice.
>> The only alternative is to add an additional argument to functions
>> taking flowi, which is even uglier.
>>
>> So, it looks like netns still have to go to flowi, but functions copying
>> flowi (in route.c/flow.c/whatever) should not use raw memcpy to store this
>> and must remember that saving flowi is possible only when refcnt to netns
>> is held somewhere.
>>
>> Alexey
>
> Thanks Alexey for your analysis.
>
> There is no refcount for netns held because it is used as an identifier. We can
> perhaps make it clear by changing the field fl_net by:
>
> struct net *fl_net => unsigned long fl_net_key;

> In this case, we must track all places where we reused fl_net as a pointer to
> retrieve the netns like in route.c, fib_hash.c or fib_rules.c because in this
> case we must held a reference. So the functions will probably take a new netns
> parameter or pick the netns pointer from somewhere else.

I did a quick grep for the places we actually use fl_net, and we barely
examine it so I don't expect there will be to much pain.

Several of the references work against the routing table entry and
we can just put a struct net reference in rtable. (The hold_net and release_net is
just for sanity checking).

```
net/ipv4/icmp.c:                dev = dev_get_by_index(rt->fl.fl_net, rt->fl.iif);
net/ipv4/route.c:       peer = inet_getpeer(rt->fl.fl_net, rt->rt_dst, create);
net/ipv4/route.c:                       hold_net(rt->fl.fl_net);
net/ipv4/route.c:       release_net(rt->fl.fl_net);
net/ipv4/route.c:       rth->fl.fl_net  = hold_net(oldflp->fl_net);
net/ipv4/route.c:               rth->fl.fl_net == flp->fl_net &&
```

The rest look like we will have to examine in detail.

```
net/ipv4/fib_rules.c:   if ((tbl = fib_get_table(flp->fl_net, rule->table)) == NULL)
net/ipv4/fib_rules.c:           if ((tb = fib_get_table(flp->fl_net, res->r->table)) != NULL)
net/ipv4/route.c:               if (r->fl.fl_net != st->p.net)
include/net/ip_fib.h:   struct net *net = flp->fl_net;
include/net/ip_fib.h:   struct net *net = flp->fl_net;
net/ipv4/fib_hash.c:    struct net *net = flp->fl_net;
net/ipv4/fib_rules.c:   struct net *net = flp->fl_net;
net/ipv4/fib_trie.c:    struct net *net = flp->fl_net;
net/ipv4/icmp.c:        net = rt->fl.fl_net;
net/ipv4/route.c:               fl1->fl_net == fl2->fl_net;
net/ipv4/route.c:       struct net *net = oldflp->fl_net;
```

But it is a small enough list it shouldn't take an insanely long time to look at.

Eric