Subject: [patch 06/10][NETNS][IP6_FIB] pass the network namespace parameter to timer callback
Posted by Daniel Lezcano on Thu, 15 Nov 2007 14:01:49 GMT
View Forum Message <> Reply to Message

The fib tables are now relative to the namespace. When the garbage collector timer expires, we must have a network namespace in order to retrieve the tables. For now this is the init_net, but we should be able to have a timer per namespace and use the timer callback parameter to pass the network namespace from the expired timer.

The timer callback, fib6_run_gc, is actually used to be called synchronously by some functions and asynchronously when the timer expires.

When the timer expires, the delay specified for fib6_run_gc parameter is alway zero. So, I changed fib6_run_gc to not be a timer callback but a function called by the timer callback and I added a timer callback where its work is just to retrieve from the data arg of the timer the network namespace and call fib6_run_gc with zero expiring time and the network namespace parameters. That makes the code cleaner for the fib6_run_gc callers.

Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>
Signed-off-by: Benjamin Thery <benjamin.thery@bull.net>
---
 include/net/ip6_fib.h | 2 +-
 net/ipv6/ip6_fib.c | 17 ++++++++++++-----
 net/ipv6/ndisc.c | 5 ++++-
 net/ipv6/route.c | 7 +++++--
 4 files changed, 21 insertions(+), 10 deletions(-)

Index: linux-2.6-netns/net/ipv6/ip6_fib.c
===
--- linux-2.6-netns.orig/net/ipv6/ip6_fib.c
+++ linux-2.6-netns/net/ipv6/ip6_fib.c
@@ -93,7 +93,9 @@ static int fib6_walk_continue(struct fib

 static __u32 rt_sernum;

-static DEFINE_TIMER(ip6_fib_timer, fib6_run_gc, 0, 0);
+static void fib6_gc_timer_cb(unsigned long arg);
+
+static DEFINE_TIMER(ip6_fib_timer, fib6_gc_timer_cb, 0, (unsigned long)&init_net);

 static struct fib6_walker_t fib6_walker_list = {
  .prev = &fib6_walker_list,
@@ -1431,11 +1433,11 @@ static int fib6_age(struct rt6_info *rt,

```
 static DEFINE_SPINLOCK(fib6_gc_lock);

-void fib6_run_gc(unsigned long dummy)
+void fib6_run_gc(unsigned long expires, struct net *net)
 {
- if (dummy != ~0UL) {
+ if (expires != ~0UL) {
   spin_lock_bh(&fib6_gc_lock);
-  gc_args.timeout = dummy ? (int)dummy : ip6_rt_gc_interval;
+  gc_args.timeout = expires ? (int)expires : ip6_rt_gc_interval;
  } else {
   local_bh_disable();
   if (!spin_trylock(&fib6_gc_lock)) {
@@ -1449,7 +1451,7 @@ void fib6_run_gc(unsigned long dummy)

  ndisc_dst_gc(&gc_args.more);

- fib6_clean_all(&init_net, fib6_age, 0, NULL);
+ fib6_clean_all(net, fib6_age, 0, NULL);

  if (gc_args.more)
   mod_timer(&ip6_fib_timer, jiffies + ip6_rt_gc_interval);
@@ -1460,6 +1462,11 @@ void fib6_run_gc(unsigned long dummy)
  spin_unlock_bh(&fib6_gc_lock);
 }

+static void fib6_gc_timer_cb(unsigned long arg)
+{
+ fib6_run_gc(0, (struct net *)arg);
+}
+
 static int fib6_net_init(struct net *net)
 {
  int ret;
Index: linux-2.6-netns/include/net/ip6_fib.h
===================================================================
--- linux-2.6-netns.orig/include/net/ip6_fib.h
+++ linux-2.6-netns/include/net/ip6_fib.h
@@ -221,7 +221,7 @@ extern int   fib6_del(struct rt6_info *r
 extern void   inet6_rt_notify(int event, struct rt6_info *rt,
       struct nl_info *info);

-extern void   fib6_run_gc(unsigned long dummy);
+extern void   fib6_run_gc(unsigned long expires, struct net *net);

 extern void   fib6_gc_cleanup(void);
```

```
Index: linux-2.6-netns/net/ipv6/ndisc.c
===================================================================
--- linux-2.6-netns.orig/net/ipv6/ndisc.c
+++ linux-2.6-netns/net/ipv6/ndisc.c
@@ -1612,6 +1612,7 @@ int ndisc_rcv(struct sk_buff *skb)
 static int ndisc_netdev_event(struct notifier_block *this, unsigned long event, void *ptr)
 {
  struct net_device *dev = ptr;
+ struct net *net = dev->nd_net;

  if (dev->nd_net != &init_net)
   return NOTIFY_DONE;
@@ -1619,11 +1620,11 @@ static int ndisc_netdev_event(struct not
  switch (event) {
  case NETDEV_CHANGEADDR:
   neigh_changeaddr(&nd_tbl, dev);
-  fib6_run_gc(~0UL);
+  fib6_run_gc(~0UL, net);
   break;
  case NETDEV_DOWN:
   neigh_ifdown(&nd_tbl, dev);
-  fib6_run_gc(~0UL);
+  fib6_run_gc(~0UL, net);
   break;
  default:
   break;
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
@@ -991,7 +991,7 @@ static int ip6_dst_gc(void)
   goto out;

  expire++;
- fib6_run_gc(expire);
+ fib6_run_gc(expire, &init_net);
  last_gc = now;
  if (atomic_read(&ip6_dst_ops.entries) < ip6_dst_ops.gc_thresh)
   expire = ip6_rt_gc_timeout>>1;
@@ -2360,9 +2360,12 @@ static
 int ipv6_sysctl_rtcache_flush(ctl_table *ctl, int write, struct file * filp,
         void __user *buffer, size_t *lenp, loff_t *ppos)
 {
+ unsigned long expires;
+
  if (write) {
+  expires = flush_delay <= 0 ? ~0UL : (unsigned long)flush_delay;
   proc_dointvec(ctl, write, filp, buffer, lenp, ppos);
```

```
-  fib6_run_gc(flush_delay <= 0 ? ~0UL : (unsigned long)flush_delay);
+  fib6_run_gc(expires, &init_net);
   return 0;
  } else
   return -EINVAL;

--
```

_____

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