Subject: [patch 2/3][NETNS][IPV6] flowlabels - make flowlabels per namespace
Posted by Daniel Lezcano on Wed, 26 Mar 2008 12:28:34 GMT
View Forum Message <> Reply to Message

This patch introduces a new member, fl_net, in struct ip6_flowlabel.
This allows to create labels with the same value in different namespaces.

Signed-off-by: Benjamin Thery <benjamin.thery@bull.net>
Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>
---
 include/net/ipv6.h | 1
 net/ipv6/ip6_flowlabel.c | 72 ++++++++++++++++++++++++++++++++++++----------
 2 files changed, 57 insertions(+), 16 deletions(-)

Index: net-2.6.26/include/net/ipv6.h
===
--- net-2.6.26.orig/include/net/ipv6.h
+++ net-2.6.26/include/net/ipv6.h
@@ -202,6 +202,7 @@ struct ip6_flowlabel
  u32   owner;
  unsigned long lastuse;
  unsigned long expires;
+ struct net *fl_net;
 };

 #define IPV6_FLOWINFO_MASK __constant_htonl(0x0FFFFFFF)
Index: net-2.6.26/net/ipv6/ip6_flowlabel.c
===
--- net-2.6.26.orig/net/ipv6/ip6_flowlabel.c
+++ net-2.6.26/net/ipv6/ip6_flowlabel.c
@@ -62,23 +62,23 @@ static DEFINE_RWLOCK(ip6_fl_lock);
 static DEFINE_RWLOCK(ip6_sk_fl_lock);

-static __inline__ struct ip6_flowlabel * __fl_lookup(__be32 label)
+static inline struct ip6_flowlabel *__fl_lookup(struct net *net, __be32 label)
 {
  struct ip6_flowlabel *fl;

  for (fl=fl_ht[FL_HASH(label)]; fl; fl = fl->next) {
- if (fl->label == label)
+ if (fl->label == label && fl->fl_net == net)
    return fl;
  }
  return NULL;
 }

-static struct ip6_flowlabel * fl_lookup(__be32 label)

```
+static struct ip6_flowlabel *fl_lookup(struct net *net, __be32 label)
 {
  struct ip6_flowlabel *fl;

  read_lock_bh(&ip6_fl_lock);
- fl = __fl_lookup(label);
+ fl = __fl_lookup(net, label);
  if (fl)
   atomic_inc(&fl->users);
  read_unlock_bh(&ip6_fl_lock);
@@ -88,8 +88,10 @@ static struct ip6_flowlabel * fl_lookup(

 static void fl_free(struct ip6_flowlabel *fl)
 {
- if (fl)
+ if (fl) {
+  release_net(fl->fl_net);
   kfree(fl->opt);
+ }
  kfree(fl);
 }

@@ -112,7 +114,6 @@ static void fl_release(struct ip6_flowla
       time_after(ip6_fl_gc_timer.expires, ttd))
    mod_timer(&ip6_fl_gc_timer, ttd);
  }
-
  write_unlock_bh(&ip6_fl_lock);
 }

@@ -148,13 +149,34 @@ static void ip6_fl_gc(unsigned long dumm
  if (!sched && atomic_read(&fl_size))
   sched = now + FL_MAX_LINGER;
  if (sched) {
-  ip6_fl_gc_timer.expires = sched;
-  add_timer(&ip6_fl_gc_timer);
+  mod_timer(&ip6_fl_gc_timer, sched);
  }
  write_unlock(&ip6_fl_lock);
 }

-static struct ip6_flowlabel *fl_intern(struct ip6_flowlabel *fl, __be32 label)
+static void ip6_fl_purge(struct net *net)
+{
+ int i;
+
+ write_lock(&ip6_fl_lock);
+ for (i = 0; i <= FL_HASH_MASK; i++) {
```

```
+  struct ip6_flowlabel *fl, **flp;
+  flp = &fl_ht[i];
+  while ((fl = *flp) != NULL) {
+   if (fl->fl_net == net && atomic_read(&fl->users) == 0) {
+    *flp = fl->next;
+    fl_free(fl);
+    atomic_dec(&fl_size);
+    continue;
+   }
+   flp = &fl->next;
+  }
+ }
+ write_unlock(&ip6_fl_lock);
+}
+
+static struct ip6_flowlabel *fl_intern(struct net *net,
+           struct ip6_flowlabel *fl, __be32 label)
 {
  struct ip6_flowlabel *lfl;

@@ -165,7 +187,7 @@ static struct ip6_flowlabel *fl_intern(s
   for (;;) {
    fl->label = htonl(net_random())&IPV6_FLOWLABEL_MASK;
    if (fl->label) {
-    lfl = __fl_lookup(fl->label);
+    lfl = __fl_lookup(net, fl->label);
     if (lfl == NULL)
      break;
    }
@@ -179,7 +201,7 @@ static struct ip6_flowlabel *fl_intern(s
    * done in ipv6_flowlabel_opt - sock is locked, so new entry
    * with the same label can only appear on another sock
    */
-  lfl = __fl_lookup(fl->label);
+  lfl = __fl_lookup(net, fl->label);
   if (lfl != NULL) {
    atomic_inc(&lfl->users);
    write_unlock_bh(&ip6_fl_lock);
@@ -298,7 +320,8 @@ static int fl6_renew(struct ip6_flowlabe
 }

 static struct ip6_flowlabel *
-fl_create(struct in6_flowlabel_req *freq, char __user *optval, int optlen, int *err_p)
+fl_create(struct net *net, struct in6_flowlabel_req *freq, char __user *optval,
+   int optlen, int *err_p)
 {
  struct ip6_flowlabel *fl;
  int olen;
```

```
@@ -343,6 +366,7 @@ fl_create(struct in6_flowlabel_req *freq
   }
  }

+ fl->fl_net = hold_net(net);
  fl->expires = jiffies;
  err = fl6_renew(fl, freq->flr_linger, freq->flr_expires);
  if (err)
@@ -441,6 +465,7 @@ static inline void fl_link(struct ipv6_p
 int ipv6_flowlabel_opt(struct sock *sk, char __user *optval, int optlen)
 {
  int err;
+ struct net *net = sock_net(sk);
  struct ipv6_pinfo *np = inet6_sk(sk);
  struct in6_flowlabel_req freq;
  struct ipv6_fl_socklist *sfl1=NULL;
@@ -483,7 +508,7 @@ int ipv6_flowlabel_opt(struct sock *sk,
   read_unlock_bh(&ip6_sk_fl_lock);

   if (freq.flr_share == IPV6_FL_S_NONE && capable(CAP_NET_ADMIN)) {
-   fl = fl_lookup(freq.flr_label);
+   fl = fl_lookup(net, freq.flr_label);
    if (fl) {
     err = fl6_renew(fl, freq.flr_linger, freq.flr_expires);
     fl_release(fl);
@@ -496,7 +521,7 @@ int ipv6_flowlabel_opt(struct sock *sk,
   if (freq.flr_label & ~IPV6_FLOWLABEL_MASK)
    return -EINVAL;

-  fl = fl_create(&freq, optval, optlen, &err);
+  fl = fl_create(net, &freq, optval, optlen, &err);
   if (fl == NULL)
    return err;
   sfl1 = kmalloc(sizeof(*sfl1), GFP_KERNEL);
@@ -518,7 +543,7 @@ int ipv6_flowlabel_opt(struct sock *sk,
    read_unlock_bh(&ip6_sk_fl_lock);

    if (fl1 == NULL)
-    fl1 = fl_lookup(freq.flr_label);
+    fl1 = fl_lookup(net, freq.flr_label);
    if (fl1) {
 recheck:
     err = -EEXIST;
@@ -559,7 +584,7 @@ release:
   if (sfl1 == NULL || (err = mem_check(sk)) != 0)
    goto done;

-  fl1 = fl_intern(fl, freq.flr_label);
```

```
+  fl1 = fl_intern(net, fl, freq.flr_label);
   if (fl1 != NULL)
    goto recheck;

@@ -717,13 +742,28 @@ static inline void ip6_flowlabel_proc_fi
 }
 #endif

+static inline void ip6_flowlabel_net_exit(struct net *net)
+{
+ ip6_fl_purge(net);
+}
+
+static struct pernet_operations ip6_flowlabel_net_ops = {
+ .exit = ip6_flowlabel_net_exit,
+};
+
 int ip6_flowlabel_init(void)
 {
+ int err;
+
+ err = register_pernet_subsys(&ip6_flowlabel_net_ops);
+ if (err)
+  return err;
  return ip6_flowlabel_proc_init(&init_net);
 }

 void ip6_flowlabel_cleanup(void)
 {
  del_timer(&ip6_fl_gc_timer);
+ unregister_pernet_subsys(&ip6_flowlabel_net_ops);
  ip6_flowlabel_proc_fini(&init_net);
 }

--
```