Subject: [PATCH net-2.6.25 2/10][NETNS][FRAGS]: Make the inet_frag_queue lookup work in namespaces.
Posted by Pavel Emelianov on Tue, 22 Jan 2008 13:57:06 GMT
View Forum Message <> Reply to Message

Since fragment management code is consolidated, we
cannot have the pointer from inet_frag_queue to
struct net, since we must know what king of fragment
this is.

So, I introduce the netns_frags structure. This one
is currently empty, but will be eventually filled with
per-namespace attributes. Each inet_frag_queue is
tagged with this one.

The conntrack_reasm is not "netns-izated", so it has
one statis netns_frags instance to keep working in
init namespace.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

---
 include/net/inet_frag.h | 8 ++++++--
 include/net/netns/ipv4.h | 4 ++++
 include/net/netns/ipv6.h | 1 +
 net/ipv4/inet_fragment.c | 27 +++++++++++++++++++++++------------
 net/ipv4/ip_fragment.c | 8 +++++---
 net/ipv6/netfilter/nf_conntrack_reasm.c | 3 ++-
 net/ipv6/reassembly.c | 8 +++++---
 7 files changed, 38 insertions(+), 21 deletions(-)

diff --git a/include/net/inet_frag.h b/include/net/inet_frag.h
index 954def4..8ab6df6 100644
--- a/include/net/inet_frag.h
+++ b/include/net/inet_frag.h
@@ -1,8 +1,12 @@
 #ifndef __NET_FRAG_H__
 #define __NET_FRAG_H__

+struct netns_frags {
+};
+
 struct inet_frag_queue {
  struct hlist_node list;
+ struct netns_frags *net;
  struct list_head lru_list;   /* lru list member */
  spinlock_t lock;
  atomic_t refcnt;

```
@@ -55,8 +59,8 @@ void inet_frag_kill(struct inet_frag_queue *q, struct inet_frags *f);
 void inet_frag_destroy(struct inet_frag_queue *q,
     struct inet_frags *f, int *work);
 int inet_frag_evictor(struct inet_frags *f);
-struct inet_frag_queue *inet_frag_find(struct inet_frags *f, void *key,
-  unsigned int hash);
+struct inet_frag_queue *inet_frag_find(struct netns_frags *nf,
+  struct inet_frags *f, void *key, unsigned int hash);

 static inline void inet_frag_put(struct inet_frag_queue *q, struct inet_frags *f)
 {
diff --git a/include/net/netns/ipv4.h b/include/net/netns/ipv4.h
index 3872aa7..80680e0 100644
--- a/include/net/netns/ipv4.h
+++ b/include/net/netns/ipv4.h
@@ -5,6 +5,8 @@
 #ifndef __NETNS_IPV4_H__
 #define __NETNS_IPV4_H__

+#include <net/inet_frag.h>
+
 struct ctl_table_header;
 struct ipv4_devconf;
 struct fib_rules_ops;
@@ -22,5 +24,7 @@ struct netns_ipv4 {
 #endif
   struct hlist_head *fib_table_hash;
   struct sock  *fibnl;
+
+ struct netns_frags frags;
 };
 #endif
diff --git a/include/net/netns/ipv6.h b/include/net/netns/ipv6.h
index 06b4dc0..057c8e4 100644
--- a/include/net/netns/ipv6.h
+++ b/include/net/netns/ipv6.h
@@ -30,5 +30,6 @@ struct netns_ipv6 {
   struct netns_sysctl_ipv6 sysctl;
   struct ipv6_devconf *devconf_all;
   struct ipv6_devconf *devconf_dflt;
+ struct netns_frags frags;
 };
 #endif
diff --git a/net/ipv4/inet_fragment.c b/net/ipv4/inet_fragment.c
index 7379107..158c5f6 100644
--- a/net/ipv4/inet_fragment.c
+++ b/net/ipv4/inet_fragment.c
@@ -174,8 +174,9 @@ int inet_frag_evictor(struct inet_frags *f)
```

```
 }
 EXPORT_SYMBOL(inet_frag_evictor);

-static struct inet_frag_queue *inet_frag_intern(struct inet_frag_queue *qp_in,
-  struct inet_frags *f, unsigned int hash, void *arg)
+static struct inet_frag_queue *inet_frag_intern(struct netns_frags *nf,
+  struct inet_frag_queue *qp_in, struct inet_frags *f,
+  unsigned int hash, void *arg)
 {
  struct inet_frag_queue *qp;
 #ifdef CONFIG_SMP
@@ -189,7 +190,7 @@ static struct inet_frag_queue *inet_frag_intern(struct inet_frag_queue
*qp_in,
   * promoted read lock to write lock.
   */
  hlist_for_each_entry(qp, n, &f->hash[hash], list) {
-  if (f->match(qp, arg)) {
+  if (qp->net == nf && f->match(qp, arg)) {
    atomic_inc(&qp->refcnt);
    write_unlock(&f->lock);
    qp_in->last_in |= COMPLETE;
@@ -210,7 +211,8 @@ static struct inet_frag_queue *inet_frag_intern(struct inet_frag_queue
*qp_in,
  return qp;
 }

-static struct inet_frag_queue *inet_frag_alloc(struct inet_frags *f, void *arg)
+static struct inet_frag_queue *inet_frag_alloc(struct netns_frags *nf,
+  struct inet_frags *f, void *arg)
 {
  struct inet_frag_queue *q;

@@ -223,31 +225,32 @@ static struct inet_frag_queue *inet_frag_alloc(struct inet_frags *f, void
*arg)
  setup_timer(&q->timer, f->frag_expire, (unsigned long)q);
  spin_lock_init(&q->lock);
  atomic_set(&q->refcnt, 1);
+ q->net = nf;

  return q;
 }

-static struct inet_frag_queue *inet_frag_create(struct inet_frags *f,
-  void *arg, unsigned int hash)
+static struct inet_frag_queue *inet_frag_create(struct netns_frags *nf,
+  struct inet_frags *f, void *arg, unsigned int hash)
 {
  struct inet_frag_queue *q;
```

```
- q = inet_frag_alloc(f, arg);
+ q = inet_frag_alloc(nf, f, arg);
  if (q == NULL)
   return NULL;

- return inet_frag_intern(q, f, hash, arg);
+ return inet_frag_intern(nf, q, f, hash, arg);
 }

-struct inet_frag_queue *inet_frag_find(struct inet_frags *f, void *key,
-  unsigned int hash)
+struct inet_frag_queue *inet_frag_find(struct netns_frags *nf,
+  struct inet_frags *f, void *key, unsigned int hash)
 {
  struct inet_frag_queue *q;
  struct hlist_node *n;

  read_lock(&f->lock);
  hlist_for_each_entry(q, n, &f->hash[hash], list) {
-  if (f->match(q, key)) {
+  if (q->net == nf && f->match(q, key)) {
    atomic_inc(&q->refcnt);
    read_unlock(&f->lock);
    return q;
@@ -255,6 +258,6 @@ struct inet_frag_queue *inet_frag_find(struct inet_frags *f, void *key,
  }
  read_unlock(&f->lock);

- return inet_frag_create(f, key, hash);
+ return inet_frag_create(nf, f, key, hash);
 }
 EXPORT_SYMBOL(inet_frag_find);
diff --git a/net/ipv4/ip_fragment.c b/net/ipv4/ip_fragment.c
index a53463e..56211ef 100644
--- a/net/ipv4/ip_fragment.c
+++ b/net/ipv4/ip_fragment.c
@@ -236,7 +236,7 @@ out:
 /* Find the correct entry in the "incomplete datagrams" queue for
  * this IP datagram, and create new one, if nothing is found.
  */
-static inline struct ipq *ip_find(struct iphdr *iph, u32 user)
+static inline struct ipq *ip_find(struct net *net, struct iphdr *iph, u32 user)
 {
  struct inet_frag_queue *q;
  struct ip4_create_arg arg;
@@ -246,7 +246,7 @@ static inline struct ipq *ip_find(struct iphdr *iph, u32 user)
  arg.user = user;
```

```
  hash = ipqhashfn(iph->id, iph->saddr, iph->daddr, iph->protocol);

- q = inet_frag_find(&ip4_frags, &arg, hash);
+ q = inet_frag_find(&net->ipv4.frags, &ip4_frags, &arg, hash);
  if (q == NULL)
   goto out_nomem;

@@ -582,15 +582,17 @@ out_fail:
 int ip_defrag(struct sk_buff *skb, u32 user)
 {
  struct ipq *qp;
+ struct net *net;

  IP_INC_STATS_BH(IPSTATS_MIB_REASMREQDS);

+ net = skb->dev->nd_net;
  /* Start by cleaning up the memory. */
  if (atomic_read(&ip4_frags.mem) > ip4_frags_ctl.high_thresh)
   ip_evictor();

  /* Lookup (or create) queue header */
- if ((qp = ip_find(ip_hdr(skb), user)) != NULL) {
+ if ((qp = ip_find(net, ip_hdr(skb), user)) != NULL) {
   int ret;

   spin_lock(&qp->q.lock);
diff --git a/net/ipv6/netfilter/nf_conntrack_reasm.c b/net/ipv6/netfilter/nf_conntrack_reasm.c
index d631631..18accd4 100644
--- a/net/ipv6/netfilter/nf_conntrack_reasm.c
+++ b/net/ipv6/netfilter/nf_conntrack_reasm.c
@@ -78,6 +78,7 @@ static struct inet_frags_ctl nf_frags_ctl __read_mostly = {
 };

 static struct inet_frags nf_frags;
+static struct netns_frags nf_init_frags;

 #ifdef CONFIG_SYSCTL
 struct ctl_table nf_ct_ipv6_sysctl_table[] = {
@@ -212,7 +213,7 @@ fq_find(__be32 id, struct in6_addr *src, struct in6_addr *dst)
  arg.dst = dst;
  hash = ip6qhashfn(id, src, dst);

- q = inet_frag_find(&nf_frags, &arg, hash);
+ q = inet_frag_find(&nf_init_frags, &nf_frags, &arg, hash);
  if (q == NULL)
   goto oom;

diff --git a/net/ipv6/reassembly.c b/net/ipv6/reassembly.c
```

```
index 1815ff0..ab2d53b 100644
--- a/net/ipv6/reassembly.c
+++ b/net/ipv6/reassembly.c
@@ -234,7 +234,7 @@ out:
 }

 static __inline__ struct frag_queue *
-fq_find(__be32 id, struct in6_addr *src, struct in6_addr *dst,
+fq_find(struct net *net, __be32 id, struct in6_addr *src, struct in6_addr *dst,
  struct inet6_dev *idev)
 {
  struct inet_frag_queue *q;
@@ -246,7 +246,7 @@ fq_find(__be32 id, struct in6_addr *src, struct in6_addr *dst,
  arg.dst = dst;
  hash = ip6qhashfn(id, src, dst);

- q = inet_frag_find(&ip6_frags, &arg, hash);
+ q = inet_frag_find(&net->ipv6.frags, &ip6_frags, &arg, hash);
  if (q == NULL)
   goto oom;

@@ -568,6 +568,7 @@ static int ipv6_frag_rcv(struct sk_buff *skb)
  struct frag_hdr *fhdr;
  struct frag_queue *fq;
  struct ipv6hdr *hdr = ipv6_hdr(skb);
+ struct net *net;

  IP6_INC_STATS_BH(ip6_dst_idev(skb->dst), IPSTATS_MIB_REASMREQDS);

@@ -598,10 +599,11 @@ static int ipv6_frag_rcv(struct sk_buff *skb)
   return 1;
  }

+ net = skb->dev->nd_net;
  if (atomic_read(&ip6_frags.mem) > init_net.ipv6.sysctl.frags.high_thresh)
   ip6_evictor(ip6_dst_idev(skb->dst));

- if ((fq = fq_find(fhdr->identification, &hdr->saddr, &hdr->daddr,
+ if ((fq = fq_find(net, fhdr->identification, &hdr->saddr, &hdr->daddr,
       ip6_dst_idev(skb->dst))) != NULL) {
   int ret;

--
1.5.3.4
```