Subject: [PATCH net-2.6.25 0/10] Make fragments live in net namespaces
Posted by Pavel Emelianov on Tue, 22 Jan 2008 13:52:36 GMT
View Forum Message <> Reply to Message

The overall design I propose is to keep the hash table
global and tag inet_frag_queue with the net. Since the
fragments hash is going to be re-sizable, this is OK to
keep fragments from different namespace in one hash.

To speedup the evicting process LRU list is made per
namespace.

As far as the CTL-tuned variables are concerned, the
timeout and thresholds are made per namespace, since
they have the per namespace sense, but the secret rebuild
interval is read-only in sub-namespaces.

Since fragment management code is consolidated for ipv4
and ipv6 I make them all in one go. The conntrack_reasm
netns-ization is not done - we have to make at least the
core netfilter per namespace first, but this reasm code
is patched to keep working in the initial namespace.

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

Subject: [PATCH net-2.6.25 1/10][NETNS][FRAGS]: Move ctl tables around.
Posted by Pavel Emelianov on Tue, 22 Jan 2008 13:55:23 GMT
View Forum Message <> Reply to Message

This is a preparation for sysctl netns-ization.
Move the ctl tables to the files, where the tuning
variables reside. Plus make the helpers to register
the tables.

This will simplify the later patches and will keep
similar things closer to each other.

ipv4, ipv6 and conntrack_reasm are patched differently,
bu the result is all the tables are in appropriate files.

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

---
 include/net/ip.h | 5 --
 include/net/ipv6.h | 1 -
 include/net/netfilter/ipv6/nf_conntrack_ipv6.h | 4 +-
 net/ipv4/ip_fragment.c | 74 ++++++++++++++++++++++++++-

```
 net/ipv4/sysctl_net_ipv4.c                  |   42 ----------
 net/ipv6/af_inet6.c                         |    5 --
 net/ipv6/netfilter/nf_conntrack_l3proto_ipv6.c |   29 ---------
 net/ipv6/netfilter/nf_conntrack_reasm.c     |   31 ++++++++++-
 net/ipv6/reassembly.c                       |   66 ++++++++++++++++++++++-
 net/ipv6/sysctl_net_ipv6.c                  |   40 +----------
 10 files changed, 169 insertions(+), 128 deletions(-)

diff --git a/include/net/ip.h b/include/net/ip.h
index 2ad4d2f..ff14fc8 100644
--- a/include/net/ip.h
+++ b/include/net/ip.h
@@ -179,11 +179,6 @@ extern int sysctl_ip_nonlocal_bind;

 extern struct ctl_path net_ipv4_ctl_path[];

-/* From ip_fragment.c */
-struct inet_frags_ctl;
-extern struct inet_frags_ctl ip4_frags_ctl;
-extern int sysctl_ipfrag_max_dist;
-
 /* From inetpeer.c */
 extern int inet_peer_threshold;
 extern int inet_peer_minttl;
diff --git a/include/net/ipv6.h b/include/net/ipv6.h
index 3712cae..87ca1bf 100644
--- a/include/net/ipv6.h
+++ b/include/net/ipv6.h
@@ -587,7 +587,6 @@ extern int ip6_mc_msfget(struct sock *sk, struct group_filter *gsf,

 #ifdef CONFIG_PROC_FS
 extern struct ctl_table *ipv6_icmp_sysctl_init(struct net *net);
-extern void ipv6_frag_sysctl_init(struct net *net);
 extern struct ctl_table *ipv6_route_sysctl_init(struct net *net);

 extern int  ac6_proc_init(void);
diff --git a/include/net/netfilter/ipv6/nf_conntrack_ipv6.h
b/include/net/netfilter/ipv6/nf_conntrack_ipv6.h
index f703533..abc55ad 100644
--- a/include/net/netfilter/ipv6/nf_conntrack_ipv6.h
+++ b/include/net/netfilter/ipv6/nf_conntrack_ipv6.h
@@ -16,6 +16,8 @@ extern void nf_ct_frag6_output(unsigned int hooknum, struct sk_buff *skb,
          int (*okfn)(struct sk_buff *));

 struct inet_frags_ctl;
-extern struct inet_frags_ctl nf_frags_ctl;
+
+#include <linux/sysctl.h>
```

```
+extern struct ctl_table nf_ct_ipv6_sysctl_table[];

 #endif /* _NF_CONNTRACK_IPV6_H*/
diff --git a/net/ipv4/ip_fragment.c b/net/ipv4/ip_fragment.c
index 2143bf3..a53463e 100644
--- a/net/ipv4/ip_fragment.c
+++ b/net/ipv4/ip_fragment.c
@@ -50,7 +50,7 @@
  * as well. Or notify me, at least. --ANK
  */

-int sysctl_ipfrag_max_dist __read_mostly = 64;
+static int sysctl_ipfrag_max_dist __read_mostly = 64;

 struct ipfrag_skb_cb
 {
@@ -74,7 +74,7 @@ struct ipq {
  struct inet_peer *peer;
 };

-struct inet_frags_ctl ip4_frags_ctl __read_mostly = {
+static struct inet_frags_ctl ip4_frags_ctl __read_mostly = {
  /*
   * Fragment cache limits. We will commit 256K at one time. Should we
   * cross that limit we will prune down to 192K. This should cope with
@@ -607,8 +607,78 @@ int ip_defrag(struct sk_buff *skb, u32 user)
  return -ENOMEM;
 }

+#ifdef CONFIG_SYSCTL
+static int zero;
+
+static struct ctl_table ip4_frags_ctl_table[] = {
+ {
+  .ctl_name = NET_IPV4_IPFRAG_HIGH_THRESH,
+  .procname = "ipfrag_high_thresh",
+  .data  = &ip4_frags_ctl.high_thresh,
+  .maxlen  = sizeof(int),
+  .mode  = 0644,
+  .proc_handler = &proc_dointvec
+ },
+ {
+  .ctl_name = NET_IPV4_IPFRAG_LOW_THRESH,
+  .procname = "ipfrag_low_thresh",
+  .data  = &ip4_frags_ctl.low_thresh,
+  .maxlen  = sizeof(int),
+  .mode  = 0644,
+  .proc_handler = &proc_dointvec
```

```
+ },
+ {
+  .ctl_name = NET_IPV4_IPFRAG_TIME,
+  .procname = "ipfrag_time",
+  .data  = &ip4_frags_ctl.timeout,
+  .maxlen  = sizeof(int),
+  .mode  = 0644,
+  .proc_handler = &proc_dointvec_jiffies,
+  .strategy = &sysctl_jiffies
+ },
+ {
+  .ctl_name = NET_IPV4_IPFRAG_SECRET_INTERVAL,
+  .procname = "ipfrag_secret_interval",
+  .data  = &ip4_frags_ctl.secret_interval,
+  .maxlen  = sizeof(int),
+  .mode  = 0644,
+  .proc_handler = &proc_dointvec_jiffies,
+  .strategy = &sysctl_jiffies
+ },
+ {
+  .procname = "ipfrag_max_dist",
+  .data  = &sysctl_ipfrag_max_dist,
+  .maxlen  = sizeof(int),
+  .mode  = 0644,
+  .proc_handler = &proc_dointvec_minmax,
+  .extra1  = &zero
+ },
+ { }
+};
+
+static int ip4_frags_ctl_register(struct net *net)
+{
+ struct ctl_table_header *hdr;
+
+ hdr = register_net_sysctl_table(net, net_ipv4_ctl_path,
+   ip4_frags_ctl_table);
+ return hdr == NULL ? -ENOMEM : 0;
+}
+#else
+static inline int ip4_frags_ctl_register(struct net *net)
+{
+ return 0;
+}
+#endif
+
+static int ipv4_frags_init_net(struct net *net)
+{
+ return ip4_frags_ctl_register(net);
```

```
+}
+
 void __init ipfrag_init(void)
 {
+ ipv4_frags_init_net(&init_net);
  ip4_frags.ctl = &ip4_frags_ctl;
  ip4_frags.hashfn = ip4_hashfn;
  ip4_frags.constructor = ip4_frag_init;
diff --git a/net/ipv4/sysctl_net_ipv4.c b/net/ipv4/sysctl_net_ipv4.c
index 45536a9..82cdf23 100644
--- a/net/ipv4/sysctl_net_ipv4.c
+++ b/net/ipv4/sysctl_net_ipv4.c
@@ -284,22 +284,6 @@ static struct ctl_table ipv4_table[] = {
   .proc_handler = &proc_dointvec
  },
  {
-  .ctl_name = NET_IPV4_IPFRAG_HIGH_THRESH,
-  .procname = "ipfrag_high_thresh",
-  .data  = &ip4_frags_ctl.high_thresh,
-  .maxlen  = sizeof(int),
-  .mode  = 0644,
-  .proc_handler = &proc_dointvec
- },
- {
-  .ctl_name = NET_IPV4_IPFRAG_LOW_THRESH,
-  .procname = "ipfrag_low_thresh",
-  .data  = &ip4_frags_ctl.low_thresh,
-  .maxlen  = sizeof(int),
-  .mode  = 0644,
-  .proc_handler = &proc_dointvec
- },
- {
   .ctl_name = NET_IPV4_DYNADDR,
   .procname = "ip_dynaddr",
   .data  = &sysctl_ip_dynaddr,
@@ -308,15 +292,6 @@ static struct ctl_table ipv4_table[] = {
   .proc_handler = &proc_dointvec
  },
  {
-  .ctl_name = NET_IPV4_IPFRAG_TIME,
-  .procname = "ipfrag_time",
-  .data  = &ip4_frags_ctl.timeout,
-  .maxlen  = sizeof(int),
-  .mode  = 0644,
-  .proc_handler = &proc_dointvec_jiffies,
-  .strategy = &sysctl_jiffies
- },
- {
```

```
   .ctl_name = NET_IPV4_TCP_KEEPALIVE_TIME,
   .procname = "tcp_keepalive_time",
   .data  = &sysctl_tcp_keepalive_time,
@@ -659,23 +634,6 @@ static struct ctl_table ipv4_table[] = {
   .proc_handler = &proc_dointvec
  },
  {
-  .ctl_name = NET_IPV4_IPFRAG_SECRET_INTERVAL,
-  .procname = "ipfrag_secret_interval",
-  .data  = &ip4_frags_ctl.secret_interval,
-  .maxlen  = sizeof(int),
-  .mode  = 0644,
-  .proc_handler = &proc_dointvec_jiffies,
-  .strategy = &sysctl_jiffies
- },
- {
-  .procname = "ipfrag_max_dist",
-  .data  = &sysctl_ipfrag_max_dist,
-  .maxlen  = sizeof(int),
-  .mode  = 0644,
-  .proc_handler = &proc_dointvec_minmax,
-  .extra1  = &zero
- },
- {
   .ctl_name = NET_TCP_NO_METRICS_SAVE,
   .procname = "tcp_no_metrics_save",
   .data  = &sysctl_tcp_nometrics_save,
diff --git a/net/ipv6/af_inet6.c b/net/ipv6/af_inet6.c
index 6738a7b..bddac0e 100644
--- a/net/ipv6/af_inet6.c
+++ b/net/ipv6/af_inet6.c
@@ -721,10 +721,6 @@ static void cleanup_ipv6_mibs(void)
 static int inet6_net_init(struct net *net)
 {
  net->ipv6.sysctl.bindv6only = 0;
- net->ipv6.sysctl.frags.high_thresh = 256 * 1024;
- net->ipv6.sysctl.frags.low_thresh = 192 * 1024;
- net->ipv6.sysctl.frags.timeout = IPV6_FRAG_TIMEOUT;
- net->ipv6.sysctl.frags.secret_interval = 10 * 60 * HZ;
  net->ipv6.sysctl.flush_delay = 0;
  net->ipv6.sysctl.ip6_rt_max_size = 4096;
  net->ipv6.sysctl.ip6_rt_gc_min_interval = HZ / 2;
@@ -734,7 +730,6 @@ static int inet6_net_init(struct net *net)
  net->ipv6.sysctl.ip6_rt_mtu_expires = 10*60*HZ;
  net->ipv6.sysctl.ip6_rt_min_advmss = IPV6_MIN_MTU - 20 - 40;
  net->ipv6.sysctl.icmpv6_time = 1*HZ;
- ipv6_frag_sysctl_init(net);
```

```
  return 0;
 }
diff --git a/net/ipv6/netfilter/nf_conntrack_l3proto_ipv6.c
b/net/ipv6/netfilter/nf_conntrack_l3proto_ipv6.c
index cf42f5c..2d7b024 100644
--- a/net/ipv6/netfilter/nf_conntrack_l3proto_ipv6.c
+++ b/net/ipv6/netfilter/nf_conntrack_l3proto_ipv6.c
@@ -297,35 +297,6 @@ static struct nf_hook_ops ipv6_conntrack_ops[] __read_mostly = {
  },
 };

-#ifdef CONFIG_SYSCTL
-static ctl_table nf_ct_ipv6_sysctl_table[] = {
- {
-  .procname = "nf_conntrack_frag6_timeout",
-  .data  = &nf_frags_ctl.timeout,
-  .maxlen  = sizeof(unsigned int),
-  .mode  = 0644,
-  .proc_handler = &proc_dointvec_jiffies,
- },
- {
-  .ctl_name = NET_NF_CONNTRACK_FRAG6_LOW_THRESH,
-  .procname = "nf_conntrack_frag6_low_thresh",
-  .data  = &nf_frags_ctl.low_thresh,
-  .maxlen  = sizeof(unsigned int),
-  .mode  = 0644,
-  .proc_handler = &proc_dointvec,
- },
- {
-  .ctl_name = NET_NF_CONNTRACK_FRAG6_HIGH_THRESH,
-  .procname = "nf_conntrack_frag6_high_thresh",
-  .data  = &nf_frags_ctl.high_thresh,
-  .maxlen  = sizeof(unsigned int),
-  .mode  = 0644,
-  .proc_handler = &proc_dointvec,
- },
- { .ctl_name = 0 }
-};
-#endif
-
 #if defined(CONFIG_NF_CT_NETLINK) || defined(CONFIG_NF_CT_NETLINK_MODULE)

 #include <linux/netfilter/nfnetlink.h>
diff --git a/net/ipv6/netfilter/nf_conntrack_reasm.c b/net/ipv6/netfilter/nf_conntrack_reasm.c
index e170c67..d631631 100644
--- a/net/ipv6/netfilter/nf_conntrack_reasm.c
+++ b/net/ipv6/netfilter/nf_conntrack_reasm.c
@@ -70,7 +70,7 @@ struct nf_ct_frag6_queue
```

```
  __u16   nhoffset;
 };

-struct inet_frags_ctl nf_frags_ctl __read_mostly = {
+static struct inet_frags_ctl nf_frags_ctl __read_mostly = {
  .high_thresh  = 256 * 1024,
  .low_thresh  = 192 * 1024,
  .timeout  = IPV6_FRAG_TIMEOUT,
@@ -79,6 +79,35 @@ struct inet_frags_ctl nf_frags_ctl __read_mostly = {

 static struct inet_frags nf_frags;

+#ifdef CONFIG_SYSCTL
+struct ctl_table nf_ct_ipv6_sysctl_table[] = {
+ {
+  .procname = "nf_conntrack_frag6_timeout",
+  .data  = &nf_frags_ctl.timeout,
+  .maxlen  = sizeof(unsigned int),
+  .mode  = 0644,
+  .proc_handler = &proc_dointvec_jiffies,
+ },
+ {
+  .ctl_name = NET_NF_CONNTRACK_FRAG6_LOW_THRESH,
+  .procname = "nf_conntrack_frag6_low_thresh",
+  .data  = &nf_frags_ctl.low_thresh,
+  .maxlen  = sizeof(unsigned int),
+  .mode  = 0644,
+  .proc_handler = &proc_dointvec,
+ },
+ {
+  .ctl_name = NET_NF_CONNTRACK_FRAG6_HIGH_THRESH,
+  .procname = "nf_conntrack_frag6_high_thresh",
+  .data  = &nf_frags_ctl.high_thresh,
+  .maxlen  = sizeof(unsigned int),
+  .mode  = 0644,
+  .proc_handler = &proc_dointvec,
+ },
+ { .ctl_name = 0 }
+};
+#endif
+
 static unsigned int ip6qhashfn(__be32 id, struct in6_addr *saddr,
         struct in6_addr *daddr)
 {
diff --git a/net/ipv6/reassembly.c b/net/ipv6/reassembly.c
index 4dfcddc..1815ff0 100644
--- a/net/ipv6/reassembly.c
+++ b/net/ipv6/reassembly.c
```

```
@@ -625,12 +625,70 @@ static struct inet6_protocol frag_protocol =
  .flags  = INET6_PROTO_NOPOLICY,
 };

-void ipv6_frag_sysctl_init(struct net *net)
+#ifdef CONFIG_SYSCTL
+static struct ctl_table ip6_frags_ctl_table[] = {
+ {
+  .ctl_name = NET_IPV6_IP6FRAG_HIGH_THRESH,
+  .procname = "ip6frag_high_thresh",
+  .data  = &init_net.ipv6.sysctl.frags.high_thresh,
+  .maxlen  = sizeof(int),
+  .mode  = 0644,
+  .proc_handler = &proc_dointvec
+ },
+ {
+  .ctl_name = NET_IPV6_IP6FRAG_LOW_THRESH,
+  .procname = "ip6frag_low_thresh",
+  .data  = &init_net.ipv6.sysctl.frags.low_thresh,
+  .maxlen  = sizeof(int),
+  .mode  = 0644,
+  .proc_handler = &proc_dointvec
+ },
+ {
+  .ctl_name = NET_IPV6_IP6FRAG_TIME,
+  .procname = "ip6frag_time",
+  .data  = &init_net.ipv6.sysctl.frags.timeout,
+  .maxlen  = sizeof(int),
+  .mode  = 0644,
+  .proc_handler = &proc_dointvec_jiffies,
+  .strategy = &sysctl_jiffies,
+ },
+ {
+  .ctl_name = NET_IPV6_IP6FRAG_SECRET_INTERVAL,
+  .procname = "ip6frag_secret_interval",
+  .data  = &init_net.ipv6.sysctl.frags.secret_interval,
+  .maxlen  = sizeof(int),
+  .mode  = 0644,
+  .proc_handler = &proc_dointvec_jiffies,
+  .strategy = &sysctl_jiffies
+ },
+ { }
+};
+
+static int ip6_frags_sysctl_register(struct net *net)
+{
+ struct ctl_table_header *hdr;
+
```

```
+ hdr = register_net_sysctl_table(net, net_ipv6_ctl_path,
+   ip6_frags_ctl_table);
+ return hdr == NULL ? -ENOMEM : 0;
+}
+#else
+static inline int ip6_frags_sysctl_register(struct net *net)
 {
- if (net != &init_net)
-  return;
+ return 0;
+}
+#endif

+static int ipv6_frags_init_net(struct net *net)
+{
  ip6_frags.ctl = &net->ipv6.sysctl.frags;
+
+ net->ipv6.sysctl.frags.high_thresh = 256 * 1024;
+ net->ipv6.sysctl.frags.low_thresh = 192 * 1024;
+ net->ipv6.sysctl.frags.timeout = IPV6_FRAG_TIMEOUT;
+ net->ipv6.sysctl.frags.secret_interval = 10 * 60 * HZ;
+
+ return ip6_frags_sysctl_register(net);
 }

 int __init ipv6_frag_init(void)
@@ -641,6 +699,8 @@ int __init ipv6_frag_init(void)
  if (ret)
   goto out;

+ ipv6_frags_init_net(&init_net);
+
  ip6_frags.hashfn = ip6_hashfn;
  ip6_frags.constructor = ip6_frag_init;
  ip6_frags.destructor = NULL;
diff --git a/net/ipv6/sysctl_net_ipv6.c b/net/ipv6/sysctl_net_ipv6.c
index 7197eb7..408691b 100644
--- a/net/ipv6/sysctl_net_ipv6.c
+++ b/net/ipv6/sysctl_net_ipv6.c
@@ -38,40 +38,6 @@ static ctl_table ipv6_table_template[] = {
   .proc_handler = &proc_dointvec
  },
  {
-  .ctl_name = NET_IPV6_IP6FRAG_HIGH_THRESH,
-  .procname = "ip6frag_high_thresh",
-  .data  = &init_net.ipv6.sysctl.frags.high_thresh,
-  .maxlen  = sizeof(int),
-  .mode  = 0644,
```

```
-  .proc_handler = &proc_dointvec
- },
- {
-  .ctl_name = NET_IPV6_IP6FRAG_LOW_THRESH,
-  .procname = "ip6frag_low_thresh",
-  .data  = &init_net.ipv6.sysctl.frags.low_thresh,
-  .maxlen  = sizeof(int),
-  .mode  = 0644,
-  .proc_handler = &proc_dointvec
- },
- {
-  .ctl_name = NET_IPV6_IP6FRAG_TIME,
-  .procname = "ip6frag_time",
-  .data  = &init_net.ipv6.sysctl.frags.timeout,
-  .maxlen  = sizeof(int),
-  .mode  = 0644,
-  .proc_handler = &proc_dointvec_jiffies,
-  .strategy = &sysctl_jiffies,
- },
- {
-  .ctl_name = NET_IPV6_IP6FRAG_SECRET_INTERVAL,
-  .procname = "ip6frag_secret_interval",
-  .data  = &init_net.ipv6.sysctl.frags.secret_interval,
-  .maxlen  = sizeof(int),
-  .mode  = 0644,
-  .proc_handler = &proc_dointvec_jiffies,
-  .strategy = &sysctl_jiffies
- },
- {
   .ctl_name = NET_IPV6_MLD_MAX_MSF,
   .procname = "mld_max_msf",
   .data  = &sysctl_mld_max_msf,
@@ -126,16 +92,12 @@ static int ipv6_sysctl_net_init(struct net *net)
  ipv6_table[1].child = ipv6_icmp_table;

  ipv6_table[2].data = &net->ipv6.sysctl.bindv6only;
- ipv6_table[3].data = &net->ipv6.sysctl.frags.high_thresh;
- ipv6_table[4].data = &net->ipv6.sysctl.frags.low_thresh;
- ipv6_table[5].data = &net->ipv6.sysctl.frags.timeout;
- ipv6_table[6].data = &net->ipv6.sysctl.frags.secret_interval;

  /* We don't want this value to be per namespace, it should be global
     to all namespaces, so make it read-only when we are not in the
     init network namespace */
  if (net != &init_net)
-  ipv6_table[7].mode = 0444;
+  ipv6_table[3].mode = 0444;
```

```
  net->ipv6.sysctl.table = register_net_sysctl_table(net, net_ipv6_ctl_path,
        ipv6_table);
--
1.5.3.4
```

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,
 }

-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

Subject: [PATCH net-2.6.25 3/10][NETNS][FRAGS]: Make the nqueues counter per-namespace.
Posted by Pavel Emelianov on Tue, 22 Jan 2008 13:58:13 GMT
View Forum Message <> Reply to Message

This is simple - just move the variable from struct inet_frags
to struct netns_frags and adjust the usage appropriately.

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

```
 include/net/inet_frag.h                |    4 +++-
 include/net/ip.h                       |    2 +-
 include/net/ipv6.h                     |    2 +-
 net/ipv4/inet_fragment.c               |   11 +++++++++---
 net/ipv4/ip_fragment.c                 |    6 ++++--
 net/ipv4/proc.c                        |    2 +-
 net/ipv6/netfilter/nf_conntrack_reasm.c |    1 +
 net/ipv6/proc.c                        |    2 +-
 net/ipv6/reassembly.c                  |    6 ++++--
 9 files changed, 24 insertions(+), 12 deletions(-)

diff --git a/include/net/inet_frag.h b/include/net/inet_frag.h
index 8ab6df6..d36f3a6 100644
--- a/include/net/inet_frag.h
+++ b/include/net/inet_frag.h
@@ -2,6 +2,7 @@
 #define __NET_FRAG_H__

 struct netns_frags {
+ int   nqueues;
 };

 struct inet_frag_queue {
@@ -36,7 +37,6 @@ struct inet_frags {
  struct hlist_head hash[INETFRAGS_HASHSZ];
  rwlock_t  lock;
  u32   rnd;
- int   nqueues;
  int   qsize;
  atomic_t  mem;
  struct timer_list secret_timer;
@@ -55,6 +55,8 @@ struct inet_frags {
 void inet_frags_init(struct inet_frags *);
```

```
 void inet_frags_fini(struct inet_frags *);

+void inet_frags_init_net(struct netns_frags *nf);
+
 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);
diff --git a/include/net/ip.h b/include/net/ip.h
index ff14fc8..9ea1bc5 100644
--- a/include/net/ip.h
+++ b/include/net/ip.h
@@ -330,7 +330,7 @@ enum ip_defrag_users

 int ip_defrag(struct sk_buff *skb, u32 user);
 int ip_frag_mem(void);
-int ip_frag_nqueues(void);
+int ip_frag_nqueues(struct net *net);

 /*
  * Functions provided by ip_forward.c
diff --git a/include/net/ipv6.h b/include/net/ipv6.h
index 87ca1bf..da1c089 100644
--- a/include/net/ipv6.h
+++ b/include/net/ipv6.h
@@ -245,7 +245,7 @@ struct ipv6_txoptions *ipv6_fixup_options(struct ipv6_txoptions
*opt_space,

 extern int ipv6_opt_accepted(struct sock *sk, struct sk_buff *skb);

-int ip6_frag_nqueues(void);
+int ip6_frag_nqueues(struct net *net);
 int ip6_frag_mem(void);

 #define IPV6_FRAG_TIMEOUT (60*HZ)  /* 60 seconds */
diff --git a/net/ipv4/inet_fragment.c b/net/ipv4/inet_fragment.c
index 158c5f6..4fec0b9 100644
--- a/net/ipv4/inet_fragment.c
+++ b/net/ipv4/inet_fragment.c
@@ -63,7 +63,6 @@ void inet_frags_init(struct inet_frags *f)
  f->rnd = (u32) ((num_physpages ^ (num_physpages>>7)) ^
        (jiffies ^ (jiffies >> 6)));

- f->nqueues = 0;
  atomic_set(&f->mem, 0);

  setup_timer(&f->secret_timer, inet_frag_secret_rebuild,
@@ -73,6 +72,12 @@ void inet_frags_init(struct inet_frags *f)
 }
```

```
 EXPORT_SYMBOL(inet_frags_init);

+void inet_frags_init_net(struct netns_frags *nf)
+{
+ nf->nqueues = 0;
+}
+EXPORT_SYMBOL(inet_frags_init_net);
+
 void inet_frags_fini(struct inet_frags *f)
 {
  del_timer(&f->secret_timer);
@@ -84,7 +89,7 @@ static inline void fq_unlink(struct inet_frag_queue *fq, struct inet_frags *f)
  write_lock(&f->lock);
  hlist_del(&fq->list);
  list_del(&fq->lru_list);
- f->nqueues--;
+ fq->net->nqueues--;
  write_unlock(&f->lock);
 }

@@ -206,7 +211,7 @@ static struct inet_frag_queue *inet_frag_intern(struct netns_frags *nf,
  atomic_inc(&qp->refcnt);
  hlist_add_head(&qp->list, &f->hash[hash]);
  list_add_tail(&qp->lru_list, &f->lru_list);
- f->nqueues++;
+ nf->nqueues++;
  write_unlock(&f->lock);
  return qp;
 }
diff --git a/net/ipv4/ip_fragment.c b/net/ipv4/ip_fragment.c
index 56211ef..cd8c830 100644
--- a/net/ipv4/ip_fragment.c
+++ b/net/ipv4/ip_fragment.c
@@ -95,9 +95,9 @@ static struct inet_frags_ctl ip4_frags_ctl __read_mostly = {

 static struct inet_frags ip4_frags;

-int ip_frag_nqueues(void)
+int ip_frag_nqueues(struct net *net)
 {
- return ip4_frags.nqueues;
+ return net->ipv4.frags.nqueues;
 }

 int ip_frag_mem(void)
@@ -675,6 +675,8 @@ static inline int ip4_frags_ctl_register(struct net *net)

 static int ipv4_frags_init_net(struct net *net)
```

```
 {
+ inet_frags_init_net(&net->ipv4.frags);
+
  return ip4_frags_ctl_register(net);
 }

diff --git a/net/ipv4/proc.c b/net/ipv4/proc.c
index cb3787f..bae3280 100644
--- a/net/ipv4/proc.c
+++ b/net/ipv4/proc.c
@@ -62,7 +62,7 @@ static int sockstat_seq_show(struct seq_file *seq, void *v)
  seq_printf(seq, "UDPLITE: inuse %d\n", sock_prot_inuse_get(&udplite_prot));
  seq_printf(seq, "RAW: inuse %d\n", sock_prot_inuse_get(&raw_prot));
  seq_printf(seq,  "FRAG: inuse %d memory %d\n",
-   ip_frag_nqueues(), ip_frag_mem());
+   ip_frag_nqueues(&init_net), ip_frag_mem());
  return 0;
 }

diff --git a/net/ipv6/netfilter/nf_conntrack_reasm.c b/net/ipv6/netfilter/nf_conntrack_reasm.c
index 18accd4..0b9d009 100644
--- a/net/ipv6/netfilter/nf_conntrack_reasm.c
+++ b/net/ipv6/netfilter/nf_conntrack_reasm.c
@@ -712,6 +712,7 @@ int nf_ct_frag6_init(void)
  nf_frags.qsize = sizeof(struct nf_ct_frag6_queue);
  nf_frags.match = ip6_frag_match;
  nf_frags.frag_expire = nf_ct_frag6_expire;
+ inet_frags_init_net(&nf_init_frags);
  inet_frags_init(&nf_frags);

  return 0;
diff --git a/net/ipv6/proc.c b/net/ipv6/proc.c
index 6b0314e..0b55785 100644
--- a/net/ipv6/proc.c
+++ b/net/ipv6/proc.c
@@ -44,7 +44,7 @@ static int sockstat6_seq_show(struct seq_file *seq, void *v)
  seq_printf(seq, "RAW6: inuse %d\n",
         sock_prot_inuse_get(&rawv6_prot));
  seq_printf(seq, "FRAG6: inuse %d memory %d\n",
-        ip6_frag_nqueues(), ip6_frag_mem());
+         ip6_frag_nqueues(&init_net), ip6_frag_mem());
  return 0;
 }

diff --git a/net/ipv6/reassembly.c b/net/ipv6/reassembly.c
index ab2d53b..77a8740 100644
--- a/net/ipv6/reassembly.c
+++ b/net/ipv6/reassembly.c
```

```
@@ -84,9 +84,9 @@ struct frag_queue

 static struct inet_frags ip6_frags;

-int ip6_frag_nqueues(void)
+int ip6_frag_nqueues(struct net *net)
 {
- return ip6_frags.nqueues;
+ return net->ipv6.frags.nqueues;
 }

 int ip6_frag_mem(void)
@@ -690,6 +690,8 @@ static int ipv6_frags_init_net(struct net *net)
  net->ipv6.sysctl.frags.timeout = IPV6_FRAG_TIMEOUT;
  net->ipv6.sysctl.frags.secret_interval = 10 * 60 * HZ;

+ inet_frags_init_net(&net->ipv6.frags);
+
  return ip6_frags_sysctl_register(net);
 }

--
1.5.3.4
```

Subject: [PATCH net-2.6.25 4/10][NETNS][FRAGS]: Make the mem counter per-namespace.

Posted by Pavel Emelianov on Tue, 22 Jan 2008 13:59:41 GMT

View Forum Message <> Reply to Message

This is also simple, but introduces more changes, since
then mem counter is altered in more places.

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

```
---
 include/net/inet_frag.h              |    4 ++--
 include/net/ip.h                     |    2 +-
 include/net/ipv6.h                   |    2 +-
 net/ipv4/inet_fragment.c             |   21 +++++++++++----------
 net/ipv4/ip_fragment.c               |   29 +++++++++++++++--------------
 net/ipv4/proc.c                      |    2 +-
 net/ipv6/netfilter/nf_conntrack_reasm.c |   14 +++++++-------
 net/ipv6/proc.c                      |    2 +-
 net/ipv6/reassembly.c                |   28 +++++++++++++--------------
 9 files changed, 54 insertions(+), 50 deletions(-)

diff --git a/include/net/inet_frag.h b/include/net/inet_frag.h
```

```
index d36f3a6..6edce7b 100644
--- a/include/net/inet_frag.h
+++ b/include/net/inet_frag.h
@@ -3,6 +3,7 @@

 struct netns_frags {
  int   nqueues;
+ atomic_t  mem;
 };

 struct inet_frag_queue {
@@ -38,7 +39,6 @@ struct inet_frags {
  rwlock_t  lock;
  u32   rnd;
  int   qsize;
- atomic_t  mem;
  struct timer_list secret_timer;
  struct inet_frags_ctl *ctl;

@@ -60,7 +60,7 @@ void inet_frags_init_net(struct netns_frags *nf);
 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);
+int inet_frag_evictor(struct netns_frags *nf, struct inet_frags *f);
 struct inet_frag_queue *inet_frag_find(struct netns_frags *nf,
   struct inet_frags *f, void *key, unsigned int hash);

diff --git a/include/net/ip.h b/include/net/ip.h
index 9ea1bc5..d41ff83 100644
--- a/include/net/ip.h
+++ b/include/net/ip.h
@@ -329,7 +329,7 @@ enum ip_defrag_users
 };

 int ip_defrag(struct sk_buff *skb, u32 user);
-int ip_frag_mem(void);
+int ip_frag_mem(struct net *net);
 int ip_frag_nqueues(struct net *net);

 /*
diff --git a/include/net/ipv6.h b/include/net/ipv6.h
index da1c089..fa80ea4 100644
--- a/include/net/ipv6.h
+++ b/include/net/ipv6.h
@@ -246,7 +246,7 @@ struct ipv6_txoptions *ipv6_fixup_options(struct ipv6_txoptions
*opt_space,
 extern int ipv6_opt_accepted(struct sock *sk, struct sk_buff *skb);
```

```
 int ip6_frag_nqueues(struct net *net);
-int ip6_frag_mem(void);
+int ip6_frag_mem(struct net *net);

 #define IPV6_FRAG_TIMEOUT (60*HZ)  /* 60 seconds */

diff --git a/net/ipv4/inet_fragment.c b/net/ipv4/inet_fragment.c
index 4fec0b9..ad79ae0 100644
--- a/net/ipv4/inet_fragment.c
+++ b/net/ipv4/inet_fragment.c
@@ -63,8 +63,6 @@ void inet_frags_init(struct inet_frags *f)
  f->rnd = (u32) ((num_physpages ^ (num_physpages>>7)) ^
        (jiffies ^ (jiffies >> 6)));

- atomic_set(&f->mem, 0);
-
  setup_timer(&f->secret_timer, inet_frag_secret_rebuild,
    (unsigned long)f);
  f->secret_timer.expires = jiffies + f->ctl->secret_interval;
@@ -75,6 +73,7 @@ EXPORT_SYMBOL(inet_frags_init);
 void inet_frags_init_net(struct netns_frags *nf)
 {
  nf->nqueues = 0;
+ atomic_set(&nf->mem, 0);
 }
 EXPORT_SYMBOL(inet_frags_init_net);

@@ -107,13 +106,13 @@ void inet_frag_kill(struct inet_frag_queue *fq, struct inet_frags *f)

 EXPORT_SYMBOL(inet_frag_kill);

-static inline void frag_kfree_skb(struct inet_frags *f, struct sk_buff *skb,
-      int *work)
+static inline void frag_kfree_skb(struct netns_frags *nf, struct inet_frags *f,
+  struct sk_buff *skb, int *work)
 {
  if (work)
   *work -= skb->truesize;

- atomic_sub(skb->truesize, &f->mem);
+ atomic_sub(skb->truesize, &nf->mem);
  if (f->skb_free)
   f->skb_free(skb);
  kfree_skb(skb);
@@ -123,22 +122,24 @@ void inet_frag_destroy(struct inet_frag_queue *q, struct inet_frags *f,
      int *work)
 {
```

```
  struct sk_buff *fp;
+ struct netns_frags *nf;

  BUG_TRAP(q->last_in & COMPLETE);
  BUG_TRAP(del_timer(&q->timer) == 0);

  /* Release all fragment data. */
  fp = q->fragments;
+ nf = q->net;
  while (fp) {
   struct sk_buff *xp = fp->next;

-  frag_kfree_skb(f, fp, work);
+  frag_kfree_skb(nf, f, fp, work);
   fp = xp;
  }

  if (work)
   *work -= f->qsize;
- atomic_sub(f->qsize, &f->mem);
+ atomic_sub(f->qsize, &nf->mem);

  if (f->destructor)
   f->destructor(q);
@@ -147,12 +148,12 @@ void inet_frag_destroy(struct inet_frag_queue *q, struct inet_frags *f,
 }
 EXPORT_SYMBOL(inet_frag_destroy);

-int inet_frag_evictor(struct inet_frags *f)
+int inet_frag_evictor(struct netns_frags *nf, struct inet_frags *f)
 {
  struct inet_frag_queue *q;
  int work, evicted = 0;

- work = atomic_read(&f->mem) - f->ctl->low_thresh;
+ work = atomic_read(&nf->mem) - f->ctl->low_thresh;
  while (work > 0) {
   read_lock(&f->lock);
   if (list_empty(&f->lru_list)) {
@@ -226,7 +227,7 @@ static struct inet_frag_queue *inet_frag_alloc(struct netns_frags *nf,
   return NULL;

  f->constructor(q, arg);
- atomic_add(f->qsize, &f->mem);
+ atomic_add(f->qsize, &nf->mem);
  setup_timer(&q->timer, f->frag_expire, (unsigned long)q);
  spin_lock_init(&q->lock);
  atomic_set(&q->refcnt, 1);
```

```
diff --git a/net/ipv4/ip_fragment.c b/net/ipv4/ip_fragment.c
index cd8c830..4f01334 100644
--- a/net/ipv4/ip_fragment.c
+++ b/net/ipv4/ip_fragment.c
@@ -100,9 +100,9 @@ int ip_frag_nqueues(struct net *net)
  return net->ipv4.frags.nqueues;
 }

-int ip_frag_mem(void)
+int ip_frag_mem(struct net *net)
 {
- return atomic_read(&ip4_frags.mem);
+ return atomic_read(&net->ipv4.frags.mem);
 }

 static int ip_frag_reasm(struct ipq *qp, struct sk_buff *prev,
@@ -142,11 +142,12 @@ static int ip4_frag_match(struct inet_frag_queue *q, void *a)
 }

 /* Memory Tracking Functions. */
-static __inline__ void frag_kfree_skb(struct sk_buff *skb, int *work)
+static __inline__ void frag_kfree_skb(struct netns_frags *nf,
+  struct sk_buff *skb, int *work)
 {
  if (work)
   *work -= skb->truesize;
- atomic_sub(skb->truesize, &ip4_frags.mem);
+ atomic_sub(skb->truesize, &nf->mem);
  kfree_skb(skb);
 }

@@ -192,11 +193,11 @@ static void ipq_kill(struct ipq *ipq)
 /* Memory limiting on fragments.  Evictor trashes the oldest
  * fragment queue until we are back under the threshold.
  */
-static void ip_evictor(void)
+static void ip_evictor(struct net *net)
 {
  int evicted;

- evicted = inet_frag_evictor(&ip4_frags);
+ evicted = inet_frag_evictor(&net->ipv4.frags, &ip4_frags);
  if (evicted)
   IP_ADD_STATS_BH(IPSTATS_MIB_REASMFAILS, evicted);
 }
@@ -294,7 +295,7 @@ static int ip_frag_reinit(struct ipq *qp)
  fp = qp->q.fragments;
  do {
```

```
   struct sk_buff *xp = fp->next;
-  frag_kfree_skb(fp, NULL);
+  frag_kfree_skb(qp->q.net, fp, NULL);
   fp = xp;
  } while (fp);

@@ -431,7 +432,7 @@ static int ip_frag_queue(struct ipq *qp, struct sk_buff *skb)
     qp->q.fragments = next;

    qp->q.meat -= free_it->len;
-   frag_kfree_skb(free_it, NULL);
+   frag_kfree_skb(qp->q.net, free_it, NULL);
   }
  }

@@ -451,7 +452,7 @@ static int ip_frag_queue(struct ipq *qp, struct sk_buff *skb)
  }
  qp->q.stamp = skb->tstamp;
  qp->q.meat += skb->len;
- atomic_add(skb->truesize, &ip4_frags.mem);
+ atomic_add(skb->truesize, &qp->q.net->mem);
  if (offset == 0)
   qp->q.last_in |= FIRST_IN;

@@ -534,12 +535,12 @@ static int ip_frag_reasm(struct ipq *qp, struct sk_buff *prev,
   head->len -= clone->len;
   clone->csum = 0;
   clone->ip_summed = head->ip_summed;
-  atomic_add(clone->truesize, &ip4_frags.mem);
+  atomic_add(clone->truesize, &qp->q.net->mem);
  }

  skb_shinfo(head)->frag_list = head->next;
  skb_push(head, head->data - skb_network_header(head));
- atomic_sub(head->truesize, &ip4_frags.mem);
+ atomic_sub(head->truesize, &qp->q.net->mem);

  for (fp=head->next; fp; fp = fp->next) {
   head->data_len += fp->len;
@@ -549,7 +550,7 @@ static int ip_frag_reasm(struct ipq *qp, struct sk_buff *prev,
   else if (head->ip_summed == CHECKSUM_COMPLETE)
    head->csum = csum_add(head->csum, fp->csum);
   head->truesize += fp->truesize;
-  atomic_sub(fp->truesize, &ip4_frags.mem);
+  atomic_sub(fp->truesize, &qp->q.net->mem);
  }

  head->next = NULL;
```

```
@@ -588,8 +589,8 @@ int ip_defrag(struct sk_buff *skb, u32 user)

  net = skb->dev->nd_net;
  /* Start by cleaning up the memory. */
- if (atomic_read(&ip4_frags.mem) > ip4_frags_ctl.high_thresh)
-  ip_evictor();
+ if (atomic_read(&net->ipv4.frags.mem) > ip4_frags_ctl.high_thresh)
+  ip_evictor(net);

  /* Lookup (or create) queue header */
  if ((qp = ip_find(net, ip_hdr(skb), user)) != NULL) {
diff --git a/net/ipv4/proc.c b/net/ipv4/proc.c
index bae3280..d63474c 100644
--- a/net/ipv4/proc.c
+++ b/net/ipv4/proc.c
@@ -62,7 +62,7 @@ static int sockstat_seq_show(struct seq_file *seq, void *v)
  seq_printf(seq, "UDPLITE: inuse %d\n", sock_prot_inuse_get(&udplite_prot));
  seq_printf(seq, "RAW: inuse %d\n", sock_prot_inuse_get(&raw_prot));
  seq_printf(seq,  "FRAG: inuse %d memory %d\n",
-   ip_frag_nqueues(&init_net), ip_frag_mem());
+   ip_frag_nqueues(&init_net), ip_frag_mem(&init_net));
  return 0;
 }

diff --git a/net/ipv6/netfilter/nf_conntrack_reasm.c b/net/ipv6/netfilter/nf_conntrack_reasm.c
index 0b9d009..cb826be 100644
--- a/net/ipv6/netfilter/nf_conntrack_reasm.c
+++ b/net/ipv6/netfilter/nf_conntrack_reasm.c
@@ -155,7 +155,7 @@ static inline void frag_kfree_skb(struct sk_buff *skb, unsigned int *work)
 {
  if (work)
   *work -= skb->truesize;
- atomic_sub(skb->truesize, &nf_frags.mem);
+ atomic_sub(skb->truesize, &nf_init_frags.mem);
  nf_skb_free(skb);
  kfree_skb(skb);
 }
@@ -177,7 +177,7 @@ static __inline__ void fq_kill(struct nf_ct_frag6_queue *fq)

 static void nf_ct_frag6_evictor(void)
 {
- inet_frag_evictor(&nf_frags);
+ inet_frag_evictor(&nf_init_frags, &nf_frags);
 }

 static void nf_ct_frag6_expire(unsigned long data)
@@ -382,7 +382,7 @@ static int nf_ct_frag6_queue(struct nf_ct_frag6_queue *fq, struct sk_buff
*skb,
```

```
  skb->dev = NULL;
  fq->q.stamp = skb->tstamp;
  fq->q.meat += skb->len;
- atomic_add(skb->truesize, &nf_frags.mem);
+ atomic_add(skb->truesize, &nf_init_frags.mem);

  /* The first fragment.
   * nhoffset is obtained from the first fragment, of course.
@@ -459,7 +459,7 @@ nf_ct_frag6_reasm(struct nf_ct_frag6_queue *fq, struct net_device *dev)
   clone->ip_summed = head->ip_summed;

   NFCT_FRAG6_CB(clone)->orig = NULL;
-  atomic_add(clone->truesize, &nf_frags.mem);
+  atomic_add(clone->truesize, &nf_init_frags.mem);
  }

  /* We have to remove fragment header from datagram and to relocate
@@ -473,7 +473,7 @@ nf_ct_frag6_reasm(struct nf_ct_frag6_queue *fq, struct net_device *dev)
  skb_shinfo(head)->frag_list = head->next;
  skb_reset_transport_header(head);
  skb_push(head, head->data - skb_network_header(head));
- atomic_sub(head->truesize, &nf_frags.mem);
+ atomic_sub(head->truesize, &nf_init_frags.mem);

  for (fp=head->next; fp; fp = fp->next) {
   head->data_len += fp->len;
@@ -483,7 +483,7 @@ nf_ct_frag6_reasm(struct nf_ct_frag6_queue *fq, struct net_device *dev)
   else if (head->ip_summed == CHECKSUM_COMPLETE)
    head->csum = csum_add(head->csum, fp->csum);
   head->truesize += fp->truesize;
-  atomic_sub(fp->truesize, &nf_frags.mem);
+  atomic_sub(fp->truesize, &nf_init_frags.mem);
  }

  head->next = NULL;
@@ -633,7 +633,7 @@ struct sk_buff *nf_ct_frag6_gather(struct sk_buff *skb)
   goto ret_orig;
  }

- if (atomic_read(&nf_frags.mem) > nf_frags_ctl.high_thresh)
+ if (atomic_read(&nf_init_frags.mem) > nf_frags_ctl.high_thresh)
   nf_ct_frag6_evictor();

  fq = fq_find(fhdr->identification, &hdr->saddr, &hdr->daddr);
diff --git a/net/ipv6/proc.c b/net/ipv6/proc.c
index 0b55785..c51cf34 100644
--- a/net/ipv6/proc.c
+++ b/net/ipv6/proc.c
```

```
@@ -44,7 +44,7 @@ static int sockstat6_seq_show(struct seq_file *seq, void *v)
  seq_printf(seq, "RAW6: inuse %d\n",
         sock_prot_inuse_get(&rawv6_prot));
  seq_printf(seq, "FRAG6: inuse %d memory %d\n",
-         ip6_frag_nqueues(&init_net), ip6_frag_mem());
+         ip6_frag_nqueues(&init_net), ip6_frag_mem(&init_net));
  return 0;
 }

diff --git a/net/ipv6/reassembly.c b/net/ipv6/reassembly.c
index 77a8740..241b2cc 100644
--- a/net/ipv6/reassembly.c
+++ b/net/ipv6/reassembly.c
@@ -89,9 +89,9 @@ int ip6_frag_nqueues(struct net *net)
  return net->ipv6.frags.nqueues;
 }

-int ip6_frag_mem(void)
+int ip6_frag_mem(struct net *net)
 {
- return atomic_read(&ip6_frags.mem);
+ return atomic_read(&net->ipv6.frags.mem);
 }

 static int ip6_frag_reasm(struct frag_queue *fq, struct sk_buff *prev,
@@ -149,11 +149,12 @@ int ip6_frag_match(struct inet_frag_queue *q, void *a)
 EXPORT_SYMBOL(ip6_frag_match);

 /* Memory Tracking Functions. */
-static inline void frag_kfree_skb(struct sk_buff *skb, int *work)
+static inline void frag_kfree_skb(struct netns_frags *nf,
+  struct sk_buff *skb, int *work)
 {
  if (work)
   *work -= skb->truesize;
- atomic_sub(skb->truesize, &ip6_frags.mem);
+ atomic_sub(skb->truesize, &nf->mem);
  kfree_skb(skb);
 }

@@ -183,11 +184,11 @@ static __inline__ void fq_kill(struct frag_queue *fq)
  inet_frag_kill(&fq->q, &ip6_frags);
 }

-static void ip6_evictor(struct inet6_dev *idev)
+static void ip6_evictor(struct net *net, struct inet6_dev *idev)
 {
  int evicted;
```

```
- evicted = inet_frag_evictor(&ip6_frags);
+ evicted = inet_frag_evictor(&net->ipv6.frags, &ip6_frags);
  if (evicted)
   IP6_ADD_STATS_BH(idev, IPSTATS_MIB_REASMFAILS, evicted);
 }
@@ -389,7 +390,7 @@ static int ip6_frag_queue(struct frag_queue *fq, struct sk_buff *skb,
     fq->q.fragments = next;

    fq->q.meat -= free_it->len;
-   frag_kfree_skb(free_it, NULL);
+   frag_kfree_skb(fq->q.net, free_it, NULL);
   }
  }

@@ -409,7 +410,7 @@ static int ip6_frag_queue(struct frag_queue *fq, struct sk_buff *skb,
  }
  fq->q.stamp = skb->tstamp;
  fq->q.meat += skb->len;
- atomic_add(skb->truesize, &ip6_frags.mem);
+ atomic_add(skb->truesize, &fq->q.net->mem);

  /* The first fragment.
   * nhoffset is obtained from the first fragment, of course.
@@ -503,7 +504,7 @@ static int ip6_frag_reasm(struct frag_queue *fq, struct sk_buff *prev,
   head->len -= clone->len;
   clone->csum = 0;
   clone->ip_summed = head->ip_summed;
-  atomic_add(clone->truesize, &ip6_frags.mem);
+  atomic_add(clone->truesize, &fq->q.net->mem);
  }

  /* We have to remove fragment header from datagram and to relocate
@@ -518,7 +519,7 @@ static int ip6_frag_reasm(struct frag_queue *fq, struct sk_buff *prev,
  skb_shinfo(head)->frag_list = head->next;
  skb_reset_transport_header(head);
  skb_push(head, head->data - skb_network_header(head));
- atomic_sub(head->truesize, &ip6_frags.mem);
+ atomic_sub(head->truesize, &fq->q.net->mem);

  for (fp=head->next; fp; fp = fp->next) {
   head->data_len += fp->len;
@@ -528,7 +529,7 @@ static int ip6_frag_reasm(struct frag_queue *fq, struct sk_buff *prev,
   else if (head->ip_summed == CHECKSUM_COMPLETE)
    head->csum = csum_add(head->csum, fp->csum);
   head->truesize += fp->truesize;
-  atomic_sub(fp->truesize, &ip6_frags.mem);
+  atomic_sub(fp->truesize, &fq->q.net->mem);
```

```
 }

 head->next = NULL;
@@ -600,8 +601,9 @@ static int ipv6_frag_rcv(struct sk_buff *skb)
 }

 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 (atomic_read(&net->ipv6.frags.mem) >
+    init_net.ipv6.sysctl.frags.high_thresh)
+  ip6_evictor(net, ip6_dst_idev(skb->dst));

 if ((fq = fq_find(net, fhdr->identification, &hdr->saddr, &hdr->daddr,
      ip6_dst_idev(skb->dst))) != NULL) {
--
1.5.3.4
```

Subject: Re: [PATCH net-2.6.25 1/10][NETNS][FRAGS]: Move ctl tables around.
Posted by davem on Tue, 22 Jan 2008 14:00:54 GMT
View Forum Message <> Reply to Message

From: Pavel Emelyanov <xemul@openvz.org>
Date: Tue, 22 Jan 2008 16:55:23 +0300

> This is a preparation for sysctl netns-ization.
> Move the ctl tables to the files, where the tuning
> variables reside. Plus make the helpers to register
> the tables.
>
> This will simplify the later patches and will keep
> similar things closer to each other.
>
> ipv4, ipv6 and conntrack_reasm are patched differently,
> bu the result is all the tables are in appropriate files.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

Applied.

Subject: [PATCH net-2.6.25 5/10][NETNS][FRAGS]: Duplicate sysctl tables for new namespaces.
Posted by Pavel Emelianov on Tue, 22 Jan 2008 14:01:02 GMT
View Forum Message <> Reply to Message

Each namespace has to have own tables to tune their
different parameters, so duplicate the tables and
register them.

All the tables in sub-namespaces are temporarily made
read-only.

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

---
 include/net/netns/ipv4.h | 1 +
 include/net/netns/ipv6.h | 1 +
 net/ipv4/ip_fragment.c | 42 ++---
 net/ipv6/reassembly.c | 41 +++++++++++++++++++++++++++++++++++++++---
 4 files changed, 79 insertions(+), 6 deletions(-)

```
diff --git a/include/net/netns/ipv4.h b/include/net/netns/ipv4.h
index 80680e0..15a0b05 100644
--- a/include/net/netns/ipv4.h
+++ b/include/net/netns/ipv4.h
@@ -16,6 +16,7 @@ struct sock;
 struct netns_ipv4 {
 #ifdef CONFIG_SYSCTL
  struct ctl_table_header *forw_hdr;
+ struct ctl_table_header *frags_hdr;
 #endif
  struct ipv4_devconf *devconf_all;
  struct ipv4_devconf *devconf_dflt;
diff --git a/include/net/netns/ipv6.h b/include/net/netns/ipv6.h
index 057c8e4..87ab56a 100644
--- a/include/net/netns/ipv6.h
+++ b/include/net/netns/ipv6.h
@@ -12,6 +12,7 @@ struct ctl_table_header;
 struct netns_sysctl_ipv6 {
 #ifdef CONFIG_SYSCTL
  struct ctl_table_header *table;
+ struct ctl_table_header *frags_hdr;
 #endif
  struct inet_frags_ctl frags;
  int bindv6only;
diff --git a/net/ipv4/ip_fragment.c b/net/ipv4/ip_fragment.c
index 4f01334..c51e1a1 100644
--- a/net/ipv4/ip_fragment.c
+++ b/net/ipv4/ip_fragment.c
@@ -661,17 +661,53 @@ static struct ctl_table ip4_frags_ctl_table[] = {

 static int ip4_frags_ctl_register(struct net *net)
 {
```

```
+ struct ctl_table *table;
  struct ctl_table_header *hdr;

- hdr = register_net_sysctl_table(net, net_ipv4_ctl_path,
-   ip4_frags_ctl_table);
- return hdr == NULL ? -ENOMEM : 0;
+ table = ip4_frags_ctl_table;
+ if (net != &init_net) {
+  table = kmemdup(table, sizeof(ip4_frags_ctl_table), GFP_KERNEL);
+  if (table == NULL)
+   goto err_alloc;
+
+  table[0].mode &= ~0222;
+  table[1].mode &= ~0222;
+  table[2].mode &= ~0222;
+  table[3].mode &= ~0222;
+  table[4].mode &= ~0222;
+ }
+
+ hdr = register_net_sysctl_table(net, net_ipv4_ctl_path, table);
+ if (hdr == NULL)
+  goto err_reg;
+
+ net->ipv4.frags_hdr = hdr;
+ return 0;
+
+err_reg:
+ if (net != &init_net)
+  kfree(table);
+err_alloc:
+ return -ENOMEM;
+}
+
+static void ip4_frags_ctl_unregister(struct net *net)
+{
+ struct ctl_table *table;
+
+ table = net->ipv4.frags_hdr->ctl_table_arg;
+ unregister_net_sysctl_table(net->ipv4.frags_hdr);
+ kfree(table);
 }
 #else
 static inline int ip4_frags_ctl_register(struct net *net)
 {
  return 0;
 }
+
+static inline void ip4_frags_ctl_unregister(struct net *net)
```

```
+{
+}
 #endif

 static int ipv4_frags_init_net(struct net *net)
diff --git a/net/ipv6/reassembly.c b/net/ipv6/reassembly.c
index 241b2cc..0300dcb 100644
--- a/net/ipv6/reassembly.c
+++ b/net/ipv6/reassembly.c
@@ -670,17 +670,52 @@ static struct ctl_table ip6_frags_ctl_table[] = {

 static int ip6_frags_sysctl_register(struct net *net)
 {
+ struct ctl_table *table;
  struct ctl_table_header *hdr;

- hdr = register_net_sysctl_table(net, net_ipv6_ctl_path,
-   ip6_frags_ctl_table);
- return hdr == NULL ? -ENOMEM : 0;
+ table = ip6_frags_ctl_table;
+ if (net != &init_net) {
+  table = kmemdup(table, sizeof(ip6_frags_ctl_table), GFP_KERNEL);
+  if (table == NULL)
+   goto err_alloc;
+
+  table[0].mode &= ~0222;
+  table[1].mode &= ~0222;
+  table[2].mode &= ~0222;
+  table[3].mode &= ~0222;
+ }
+
+ hdr = register_net_sysctl_table(net, net_ipv6_ctl_path, table);
+ if (hdr == NULL)
+  goto err_reg;
+
+ net->ipv6.sysctl.frags_hdr = hdr;
+ return 0;
+
+err_reg:
+ if (net != &init_net)
+  kfree(table);
+err_alloc:
+ return -ENOMEM;
+}
+
+static void ip6_frags_sysctl_unregister(struct net *net)
+{
+ struct ctl_table *table;
```

```
+
+ table = net->ipv6.sysctl.frags_hdr->ctl_table_arg;
+ unregister_net_sysctl_table(net->ipv6.sysctl.frags_hdr);
+ kfree(table);
 }
 #else
 static inline int ip6_frags_sysctl_register(struct net *net)
 {
  return 0;
 }
+
+static inline void ip6_frags_sysctl_unregister(struct net *net)
+{
+}
 #endif

 static int ipv6_frags_init_net(struct net *net)
--
1.5.3.4
```

Subject: [PATCH net-2.6.25 6/10][NETNS][FRAGS]: Make the net.ipv4.ipfrag_timeout work in namespaces.
Posted by Pavel Emelianov on Tue, 22 Jan 2008 14:02:17 GMT
View Forum Message <> Reply to Message

Move it to the netns_frags, adjust the usage and
make the appropriate ctl table writable.

Now fragment, that live in different namespaces can
live for different times.

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

```
---
 include/net/inet_frag.h                |    4 +++-
 net/ipv4/inet_fragment.c               |    2 +-
 net/ipv4/ip_fragment.c                 |   20 ++++++++++----------
 net/ipv6/netfilter/nf_conntrack_reasm.c |    4 ++--
 net/ipv6/reassembly.c                  |    6 +++---
 5 files changed, 19 insertions(+), 17 deletions(-)

diff --git a/include/net/inet_frag.h b/include/net/inet_frag.h
index 6edce7b..f56e296 100644
--- a/include/net/inet_frag.h
+++ b/include/net/inet_frag.h
@@ -4,6 +4,9 @@
 struct netns_frags {
```

```
  int   nqueues;
  atomic_t  mem;
+
+ /* sysctls */
+ int   timeout;
 };

 struct inet_frag_queue {
@@ -29,7 +32,6 @@ struct inet_frag_queue {
 struct inet_frags_ctl {
  int high_thresh;
  int low_thresh;
- int timeout;
  int secret_interval;
 };

diff --git a/net/ipv4/inet_fragment.c b/net/ipv4/inet_fragment.c
index ad79ae0..9da9679 100644
--- a/net/ipv4/inet_fragment.c
+++ b/net/ipv4/inet_fragment.c
@@ -206,7 +206,7 @@ static struct inet_frag_queue *inet_frag_intern(struct netns_frags *nf,
  }
 #endif
  qp = qp_in;
- if (!mod_timer(&qp->timer, jiffies + f->ctl->timeout))
+ if (!mod_timer(&qp->timer, jiffies + nf->timeout))
   atomic_inc(&qp->refcnt);

  atomic_inc(&qp->refcnt);
diff --git a/net/ipv4/ip_fragment.c b/net/ipv4/ip_fragment.c
index c51e1a1..70d241c 100644
--- a/net/ipv4/ip_fragment.c
+++ b/net/ipv4/ip_fragment.c
@@ -83,13 +83,6 @@ static struct inet_frags_ctl ip4_frags_ctl __read_mostly = {
   */
  .high_thresh  = 256 * 1024,
  .low_thresh  = 192 * 1024,
-
- /*
-  * Important NOTE! Fragment queue must be destroyed before MSL expires.
-  * RFC791 is wrong proposing to prolongate timer each fragment arrival
-  * by TTL.
-  */
- .timeout  = IP_FRAG_TIME,
  .secret_interval = 10 * 60 * HZ,
 };

@@ -287,7 +280,7 @@ static int ip_frag_reinit(struct ipq *qp)
```

```
 {
  struct sk_buff *fp;

- if (!mod_timer(&qp->q.timer, jiffies + ip4_frags_ctl.timeout)) {
+ if (!mod_timer(&qp->q.timer, jiffies + qp->q.net->timeout)) {
   atomic_inc(&qp->q.refcnt);
   return -ETIMEDOUT;
  }
@@ -633,7 +626,7 @@ static struct ctl_table ip4_frags_ctl_table[] = {
  {
   .ctl_name = NET_IPV4_IPFRAG_TIME,
   .procname = "ipfrag_time",
-  .data  = &ip4_frags_ctl.timeout,
+  .data  = &init_net.ipv4.frags.timeout,
   .maxlen  = sizeof(int),
   .mode  = 0644,
   .proc_handler = &proc_dointvec_jiffies,
@@ -672,7 +665,7 @@ static int ip4_frags_ctl_register(struct net *net)

   table[0].mode &= ~0222;
   table[1].mode &= ~0222;
-  table[2].mode &= ~0222;
+  table[2].data = &net->ipv4.frags.timeout;
   table[3].mode &= ~0222;
   table[4].mode &= ~0222;
  }
@@ -712,6 +705,13 @@ static inline void ip4_frags_ctl_unregister(struct net *net)

 static int ipv4_frags_init_net(struct net *net)
 {
+ /*
+  * Important NOTE! Fragment queue must be destroyed before MSL expires.
+  * RFC791 is wrong proposing to prolongate timer each fragment arrival
+  * by TTL.
+  */
+ net->ipv4.frags.timeout = IP_FRAG_TIME;
+
  inet_frags_init_net(&net->ipv4.frags);

  return ip4_frags_ctl_register(net);
diff --git a/net/ipv6/netfilter/nf_conntrack_reasm.c b/net/ipv6/netfilter/nf_conntrack_reasm.c
index cb826be..92a311f 100644
--- a/net/ipv6/netfilter/nf_conntrack_reasm.c
+++ b/net/ipv6/netfilter/nf_conntrack_reasm.c
@@ -73,7 +73,6 @@ struct nf_ct_frag6_queue
 static struct inet_frags_ctl nf_frags_ctl __read_mostly = {
  .high_thresh  = 256 * 1024,
  .low_thresh  = 192 * 1024,
```

```
- .timeout  = IPV6_FRAG_TIMEOUT,
  .secret_interval = 10 * 60 * HZ,
 };

@@ -84,7 +83,7 @@ static struct netns_frags nf_init_frags;
 struct ctl_table nf_ct_ipv6_sysctl_table[] = {
  {
   .procname = "nf_conntrack_frag6_timeout",
-  .data  = &nf_frags_ctl.timeout,
+  .data  = &nf_init_frags.timeout,
   .maxlen  = sizeof(unsigned int),
   .mode  = 0644,
   .proc_handler = &proc_dointvec_jiffies,
@@ -712,6 +711,7 @@ int nf_ct_frag6_init(void)
  nf_frags.qsize = sizeof(struct nf_ct_frag6_queue);
  nf_frags.match = ip6_frag_match;
  nf_frags.frag_expire = nf_ct_frag6_expire;
+ nf_init_frags.timeout = IPV6_FRAG_TIMEOUT;
  inet_frags_init_net(&nf_init_frags);
  inet_frags_init(&nf_frags);

diff --git a/net/ipv6/reassembly.c b/net/ipv6/reassembly.c
index 0300dcb..9176136 100644
--- a/net/ipv6/reassembly.c
+++ b/net/ipv6/reassembly.c
@@ -650,7 +650,7 @@ static struct ctl_table ip6_frags_ctl_table[] = {
  {
   .ctl_name = NET_IPV6_IP6FRAG_TIME,
   .procname = "ip6frag_time",
-  .data  = &init_net.ipv6.sysctl.frags.timeout,
+  .data  = &init_net.ipv6.frags.timeout,
   .maxlen  = sizeof(int),
   .mode  = 0644,
   .proc_handler = &proc_dointvec_jiffies,
@@ -681,7 +681,7 @@ static int ip6_frags_sysctl_register(struct net *net)

   table[0].mode &= ~0222;
   table[1].mode &= ~0222;
-  table[2].mode &= ~0222;
+  table[2].data = &net->ipv6.frags.timeout;
   table[3].mode &= ~0222;
  }

@@ -724,7 +724,7 @@ static int ipv6_frags_init_net(struct net *net)

  net->ipv6.sysctl.frags.high_thresh = 256 * 1024;
  net->ipv6.sysctl.frags.low_thresh = 192 * 1024;
- net->ipv6.sysctl.frags.timeout = IPV6_FRAG_TIMEOUT;
```

```
+ net->ipv6.frags.timeout = IPV6_FRAG_TIMEOUT;
  net->ipv6.sysctl.frags.secret_interval = 10 * 60 * HZ;

  inet_frags_init_net(&net->ipv6.frags);
--
1.5.3.4
```

Subject: [PATCH net-2.6.25 7/10][NETNS][FRAGS]: Make thresholds work in namespaces.
Posted by Pavel Emelianov on Tue, 22 Jan 2008 14:05:47 GMT
View Forum Message <> Reply to Message

This is the same as with the timeout variable.

Currently, after exceeding the high threshold _all_
the fragments are evicted, but it will be fixed in
later patch.

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

```
---
 include/net/inet_frag.h                |    4 ++--
 net/ipv4/inet_fragment.c               |    2 +-
 net/ipv4/ip_fragment.c                 |   26 ++++++++++++++++++----------
 net/ipv6/netfilter/nf_conntrack_reasm.c |   12 ++++++------
 net/ipv6/reassembly.c                  |   15 +++++++--------
 5 files changed, 29 insertions(+), 30 deletions(-)

diff --git a/include/net/inet_frag.h b/include/net/inet_frag.h
index f56e296..de41359 100644
--- a/include/net/inet_frag.h
+++ b/include/net/inet_frag.h
@@ -7,6 +7,8 @@ struct netns_frags {

   /* sysctls */
   int   timeout;
+ int   high_thresh;
+ int   low_thresh;
 };

 struct inet_frag_queue {
@@ -30,8 +32,6 @@ struct inet_frag_queue {
 #define INETFRAGS_HASHSZ  64

 struct inet_frags_ctl {
- int high_thresh;
- int low_thresh;
```

```
  int secret_interval;
 };

diff --git a/net/ipv4/inet_fragment.c b/net/ipv4/inet_fragment.c
index 9da9679..5ab399c 100644
--- a/net/ipv4/inet_fragment.c
+++ b/net/ipv4/inet_fragment.c
@@ -153,7 +153,7 @@ int inet_frag_evictor(struct netns_frags *nf, struct inet_frags *f)
  struct inet_frag_queue *q;
  int work, evicted = 0;

- work = atomic_read(&nf->mem) - f->ctl->low_thresh;
+ work = atomic_read(&nf->mem) - nf->low_thresh;
  while (work > 0) {
   read_lock(&f->lock);
   if (list_empty(&f->lru_list)) {
diff --git a/net/ipv4/ip_fragment.c b/net/ipv4/ip_fragment.c
index 70d241c..80c2c19 100644
--- a/net/ipv4/ip_fragment.c
+++ b/net/ipv4/ip_fragment.c
@@ -75,14 +75,6 @@ struct ipq {
 };

 static struct inet_frags_ctl ip4_frags_ctl __read_mostly = {
- /*
-  * Fragment cache limits. We will commit 256K at one time. Should we
-  * cross that limit we will prune down to 192K. This should cope with
-  * even the most extreme cases without allowing an attacker to
-  * measurably harm machine performance.
-  */
- .high_thresh  = 256 * 1024,
- .low_thresh  = 192 * 1024,
  .secret_interval = 10 * 60 * HZ,
 };

@@ -582,7 +574,7 @@ int ip_defrag(struct sk_buff *skb, u32 user)

  net = skb->dev->nd_net;
  /* Start by cleaning up the memory. */
- if (atomic_read(&net->ipv4.frags.mem) > ip4_frags_ctl.high_thresh)
+ if (atomic_read(&net->ipv4.frags.mem) > net->ipv4.frags.high_thresh)
   ip_evictor(net);

  /* Lookup (or create) queue header */
@@ -610,7 +602,7 @@ static struct ctl_table ip4_frags_ctl_table[] = {
  {
   .ctl_name = NET_IPV4_IPFRAG_HIGH_THRESH,
   .procname = "ipfrag_high_thresh",
```

```
-  .data  = &ip4_frags_ctl.high_thresh,
+  .data  = &init_net.ipv4.frags.high_thresh,
   .maxlen  = sizeof(int),
   .mode  = 0644,
   .proc_handler = &proc_dointvec
@@ -618,7 +610,7 @@ static struct ctl_table ip4_frags_ctl_table[] = {
  {
   .ctl_name = NET_IPV4_IPFRAG_LOW_THRESH,
   .procname = "ipfrag_low_thresh",
-  .data  = &ip4_frags_ctl.low_thresh,
+  .data  = &init_net.ipv4.frags.low_thresh,
   .maxlen  = sizeof(int),
   .mode  = 0644,
   .proc_handler = &proc_dointvec
@@ -663,8 +655,8 @@ static int ip4_frags_ctl_register(struct net *net)
   if (table == NULL)
    goto err_alloc;

-  table[0].mode &= ~0222;
-  table[1].mode &= ~0222;
+  table[0].data = &net->ipv4.frags.high_thresh;
+  table[1].data = &net->ipv4.frags.low_thresh;
   table[2].data = &net->ipv4.frags.timeout;
   table[3].mode &= ~0222;
   table[4].mode &= ~0222;
@@ -706,6 +698,14 @@ static inline void ip4_frags_ctl_unregister(struct net *net)
 static int ipv4_frags_init_net(struct net *net)
 {
  /*
+  * Fragment cache limits. We will commit 256K at one time. Should we
+  * cross that limit we will prune down to 192K. This should cope with
+  * even the most extreme cases without allowing an attacker to
+  * measurably harm machine performance.
+  */
+ net->ipv4.frags.high_thresh = 256 * 1024;
+ net->ipv4.frags.low_thresh = 192 * 1024;
+ /*
   * Important NOTE! Fragment queue must be destroyed before MSL expires.
   * RFC791 is wrong proposing to prolongate timer each fragment arrival
   * by TTL.
diff --git a/net/ipv6/netfilter/nf_conntrack_reasm.c b/net/ipv6/netfilter/nf_conntrack_reasm.c
index 92a311f..c75ac17 100644
--- a/net/ipv6/netfilter/nf_conntrack_reasm.c
+++ b/net/ipv6/netfilter/nf_conntrack_reasm.c
@@ -71,8 +71,6 @@ struct nf_ct_frag6_queue
 };

 static struct inet_frags_ctl nf_frags_ctl __read_mostly = {
```

```
- .high_thresh  = 256 * 1024,
- .low_thresh  = 192 * 1024,
  .secret_interval = 10 * 60 * HZ,
 };

@@ -91,7 +89,7 @@ struct ctl_table nf_ct_ipv6_sysctl_table[] = {
  {
   .ctl_name = NET_NF_CONNTRACK_FRAG6_LOW_THRESH,
   .procname = "nf_conntrack_frag6_low_thresh",
-  .data  = &nf_frags_ctl.low_thresh,
+  .data  = &nf_init_frags.low_thresh,
   .maxlen  = sizeof(unsigned int),
   .mode  = 0644,
   .proc_handler = &proc_dointvec,
@@ -99,7 +97,7 @@ struct ctl_table nf_ct_ipv6_sysctl_table[] = {
  {
   .ctl_name = NET_NF_CONNTRACK_FRAG6_HIGH_THRESH,
   .procname = "nf_conntrack_frag6_high_thresh",
-  .data  = &nf_frags_ctl.high_thresh,
+  .data  = &nf_init_frags.high_thresh,
   .maxlen  = sizeof(unsigned int),
   .mode  = 0644,
   .proc_handler = &proc_dointvec,
@@ -632,7 +630,7 @@ struct sk_buff *nf_ct_frag6_gather(struct sk_buff *skb)
   goto ret_orig;
  }

- if (atomic_read(&nf_init_frags.mem) > nf_frags_ctl.high_thresh)
+ if (atomic_read(&nf_init_frags.mem) > nf_init_frags.high_thresh)
   nf_ct_frag6_evictor();

  fq = fq_find(fhdr->identification, &hdr->saddr, &hdr->daddr);
@@ -712,6 +710,8 @@ int nf_ct_frag6_init(void)
  nf_frags.match = ip6_frag_match;
  nf_frags.frag_expire = nf_ct_frag6_expire;
  nf_init_frags.timeout = IPV6_FRAG_TIMEOUT;
+ nf_init_frags.high_thresh = 256 * 1024;
+ nf_init_frags.low_thresh = 192 * 1024;
  inet_frags_init_net(&nf_init_frags);
  inet_frags_init(&nf_frags);

@@ -722,6 +722,6 @@ void nf_ct_frag6_cleanup(void)
 {
  inet_frags_fini(&nf_frags);

- nf_frags_ctl.low_thresh = 0;
+ nf_init_frags.low_thresh = 0;
  nf_ct_frag6_evictor();
```

```
 }
diff --git a/net/ipv6/reassembly.c b/net/ipv6/reassembly.c
index 9176136..85f3fa3 100644
--- a/net/ipv6/reassembly.c
+++ b/net/ipv6/reassembly.c
@@ -601,8 +601,7 @@ static int ipv6_frag_rcv(struct sk_buff *skb)
  }

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

  if ((fq = fq_find(net, fhdr->identification, &hdr->saddr, &hdr->daddr,
@@ -634,7 +633,7 @@ static struct ctl_table ip6_frags_ctl_table[] = {
  {
    .ctl_name = NET_IPV6_IP6FRAG_HIGH_THRESH,
    .procname = "ip6frag_high_thresh",
-   .data  = &init_net.ipv6.sysctl.frags.high_thresh,
+   .data  = &init_net.ipv6.frags.high_thresh,
    .maxlen  = sizeof(int),
    .mode  = 0644,
    .proc_handler = &proc_dointvec
@@ -642,7 +641,7 @@ static struct ctl_table ip6_frags_ctl_table[] = {
  {
    .ctl_name = NET_IPV6_IP6FRAG_LOW_THRESH,
    .procname = "ip6frag_low_thresh",
-   .data  = &init_net.ipv6.sysctl.frags.low_thresh,
+   .data  = &init_net.ipv6.frags.low_thresh,
    .maxlen  = sizeof(int),
    .mode  = 0644,
    .proc_handler = &proc_dointvec
@@ -679,8 +678,8 @@ static int ip6_frags_sysctl_register(struct net *net)
    if (table == NULL)
      goto err_alloc;

-   table[0].mode &= ~0222;
-   table[1].mode &= ~0222;
+   table[0].data = &net->ipv6.frags.high_thresh;
+   table[1].data = &net->ipv6.frags.low_thresh;
    table[2].data = &net->ipv6.frags.timeout;
    table[3].mode &= ~0222;
  }
@@ -722,8 +721,8 @@ static int ipv6_frags_init_net(struct net *net)
 {
  ip6_frags.ctl = &net->ipv6.sysctl.frags;
```

```
- net->ipv6.sysctl.frags.high_thresh = 256 * 1024;
- net->ipv6.sysctl.frags.low_thresh = 192 * 1024;
+ net->ipv6.frags.high_thresh = 256 * 1024;
+ net->ipv6.frags.low_thresh = 192 * 1024;
  net->ipv6.frags.timeout = IPV6_FRAG_TIMEOUT;
  net->ipv6.sysctl.frags.secret_interval = 10 * 60 * HZ;

--
1.5.3.4
```

Subject: Re: [PATCH net-2.6.25 2/10][NETNS][FRAGS]: Make the inet_frag_queue lookup work in namespaces.
Posted by davem on Tue, 22 Jan 2008 14:05:58 GMT
View Forum Message <> Reply to Message

From: Pavel Emelyanov <xemul@openvz.org>
Date: Tue, 22 Jan 2008 16:57:06 +0300

> 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>

Applied.

Subject: Re: [PATCH net-2.6.25 3/10][NETNS][FRAGS]: Make the nqueues counter per-namespace.
Posted by davem on Tue, 22 Jan 2008 14:06:35 GMT
View Forum Message <> Reply to Message

From: Pavel Emelyanov <xemul@openvz.org>
Date: Tue, 22 Jan 2008 16:58:13 +0300

> This is simple - just move the variable from struct inet_frags

> to struct netns_frags and adjust the usage appropriately.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

Applied.

Subject: [PATCH net-2.6.25 8/10][NETNS][FRAGS]: Isolate the secret interval from namespaces.
Posted by Pavel Emelianov on Tue, 22 Jan 2008 14:07:08 GMT
View Forum Message <> Reply to Message

Since we have one hashtable to lookup the fragment, having
different secret_interval-s for hash rebuild doesn't make
sense, so move this one to inet_frags.

The inet_frags_ctl becomes empty after this, so remove it.
The appropriate ctl table is kept read-only in namespaces.

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

---
 include/net/inet_frag.h | 6 +-----
 include/net/netns/ipv6.h | 1 -
 net/ipv4/inet_fragment.c | 4 ++--
 net/ipv4/ip_fragment.c | 8 ++------
 net/ipv6/netfilter/nf_conntrack_reasm.c | 6 +-----
 net/ipv6/reassembly.c | 6 ++----
 6 files changed, 8 insertions(+), 23 deletions(-)

diff --git a/include/net/inet_frag.h b/include/net/inet_frag.h
index de41359..1917fbe 100644
--- a/include/net/inet_frag.h
+++ b/include/net/inet_frag.h
@@ -31,18 +31,14 @@ struct inet_frag_queue {

 #define INETFRAGS_HASHSZ 64

-struct inet_frags_ctl {
- int secret_interval;
-};
-
 struct inet_frags {
  struct list_head lru_list;
  struct hlist_head hash[INETFRAGS_HASHSZ];
  rwlock_t lock;
  u32 rnd;
  int qsize;

```
+ int   secret_interval;
  struct timer_list secret_timer;
- struct inet_frags_ctl *ctl;

  unsigned int  (*hashfn)(struct inet_frag_queue *);
  void   (*constructor)(struct inet_frag_queue *q,
diff --git a/include/net/netns/ipv6.h b/include/net/netns/ipv6.h
index 87ab56a..187c424 100644
--- a/include/net/netns/ipv6.h
+++ b/include/net/netns/ipv6.h
@@ -14,7 +14,6 @@ struct netns_sysctl_ipv6 {
  struct ctl_table_header *table;
  struct ctl_table_header *frags_hdr;
 #endif
- struct inet_frags_ctl frags;
  int bindv6only;
  int flush_delay;
  int ip6_rt_max_size;
diff --git a/net/ipv4/inet_fragment.c b/net/ipv4/inet_fragment.c
index 5ab399c..fcf5252 100644
--- a/net/ipv4/inet_fragment.c
+++ b/net/ipv4/inet_fragment.c
@@ -47,7 +47,7 @@ static void inet_frag_secret_rebuild(unsigned long dummy)
  }
  write_unlock(&f->lock);

- mod_timer(&f->secret_timer, now + f->ctl->secret_interval);
+ mod_timer(&f->secret_timer, now + f->secret_interval);
 }

 void inet_frags_init(struct inet_frags *f)
@@ -65,7 +65,7 @@ void inet_frags_init(struct inet_frags *f)

  setup_timer(&f->secret_timer, inet_frag_secret_rebuild,
    (unsigned long)f);
- f->secret_timer.expires = jiffies + f->ctl->secret_interval;
+ f->secret_timer.expires = jiffies + f->secret_interval;
  add_timer(&f->secret_timer);
 }
 EXPORT_SYMBOL(inet_frags_init);
diff --git a/net/ipv4/ip_fragment.c b/net/ipv4/ip_fragment.c
index 80c2c19..00646ed 100644
--- a/net/ipv4/ip_fragment.c
+++ b/net/ipv4/ip_fragment.c
@@ -74,10 +74,6 @@ struct ipq {
  struct inet_peer *peer;
 };
```

```
-static struct inet_frags_ctl ip4_frags_ctl __read_mostly = {
- .secret_interval = 10 * 60 * HZ,
-};
-
 static struct inet_frags ip4_frags;

 int ip_frag_nqueues(struct net *net)
@@ -627,7 +623,7 @@ static struct ctl_table ip4_frags_ctl_table[] = {
  {
   .ctl_name = NET_IPV4_IPFRAG_SECRET_INTERVAL,
   .procname = "ipfrag_secret_interval",
-  .data  = &ip4_frags_ctl.secret_interval,
+  .data  = &ip4_frags.secret_interval,
   .maxlen  = sizeof(int),
   .mode  = 0644,
   .proc_handler = &proc_dointvec_jiffies,
@@ -720,7 +716,6 @@ static int ipv4_frags_init_net(struct net *net)
 void __init ipfrag_init(void)
 {
  ipv4_frags_init_net(&init_net);
- ip4_frags.ctl = &ip4_frags_ctl;
  ip4_frags.hashfn = ip4_hashfn;
  ip4_frags.constructor = ip4_frag_init;
  ip4_frags.destructor = ip4_frag_free;
@@ -728,6 +723,7 @@ void __init ipfrag_init(void)
  ip4_frags.qsize = sizeof(struct ipq);
  ip4_frags.match = ip4_frag_match;
  ip4_frags.frag_expire = ip_expire;
+ ip4_frags.secret_interval = 10 * 60 * HZ;
  inet_frags_init(&ip4_frags);
 }

diff --git a/net/ipv6/netfilter/nf_conntrack_reasm.c b/net/ipv6/netfilter/nf_conntrack_reasm.c
index c75ac17..6eed991 100644
--- a/net/ipv6/netfilter/nf_conntrack_reasm.c
+++ b/net/ipv6/netfilter/nf_conntrack_reasm.c
@@ -70,10 +70,6 @@ struct nf_ct_frag6_queue
  __u16   nhoffset;
 };

-static struct inet_frags_ctl nf_frags_ctl __read_mostly = {
- .secret_interval = 10 * 60 * HZ,
-};
-
 static struct inet_frags nf_frags;
 static struct netns_frags nf_init_frags;

@@ -701,7 +697,6 @@ int nf_ct_frag6_kfree_frags(struct sk_buff *skb)
```

```
 int nf_ct_frag6_init(void)
 {
- nf_frags.ctl = &nf_frags_ctl;
  nf_frags.hashfn = nf_hashfn;
  nf_frags.constructor = ip6_frag_init;
  nf_frags.destructor = NULL;
@@ -709,6 +704,7 @@ int nf_ct_frag6_init(void)
  nf_frags.qsize = sizeof(struct nf_ct_frag6_queue);
  nf_frags.match = ip6_frag_match;
  nf_frags.frag_expire = nf_ct_frag6_expire;
+ nf_frags.secret_interval = 10 * 60 * HZ;
  nf_init_frags.timeout = IPV6_FRAG_TIMEOUT;
  nf_init_frags.high_thresh = 256 * 1024;
  nf_init_frags.low_thresh = 192 * 1024;
diff --git a/net/ipv6/reassembly.c b/net/ipv6/reassembly.c
index 85f3fa3..8520700 100644
--- a/net/ipv6/reassembly.c
+++ b/net/ipv6/reassembly.c
@@ -658,7 +658,7 @@ static struct ctl_table ip6_frags_ctl_table[] = {
  {
   .ctl_name = NET_IPV6_IP6FRAG_SECRET_INTERVAL,
   .procname = "ip6frag_secret_interval",
-  .data  = &init_net.ipv6.sysctl.frags.secret_interval,
+  .data  = &ip6_frags.secret_interval,
   .maxlen  = sizeof(int),
   .mode  = 0644,
   .proc_handler = &proc_dointvec_jiffies,
@@ -719,12 +719,9 @@ static inline void ip6_frags_sysctl_unregister(struct net *net)

 static int ipv6_frags_init_net(struct net *net)
 {
- ip6_frags.ctl = &net->ipv6.sysctl.frags;
-
  net->ipv6.frags.high_thresh = 256 * 1024;
  net->ipv6.frags.low_thresh = 192 * 1024;
  net->ipv6.frags.timeout = IPV6_FRAG_TIMEOUT;
- net->ipv6.sysctl.frags.secret_interval = 10 * 60 * HZ;

  inet_frags_init_net(&net->ipv6.frags);

@@ -748,6 +745,7 @@ int __init ipv6_frag_init(void)
  ip6_frags.qsize = sizeof(struct frag_queue);
  ip6_frags.match = ip6_frag_match;
  ip6_frags.frag_expire = ip6_frag_expire;
+ ip6_frags.secret_interval = 10 * 60 * HZ;
  inet_frags_init(&ip6_frags);
 out:
```

return ret;
--
1.5.3.4

Subject: Re: [PATCH net-2.6.25 4/10][NETNS][FRAGS]: Make the mem counter per-namespace.
Posted by davem on Tue, 22 Jan 2008 14:07:37 GMT
View Forum Message <> Reply to Message

From: Pavel Emelyanov <xemul@openvz.org>
Date: Tue, 22 Jan 2008 16:59:41 +0300

> This is also simple, but introduces more changes, since
> then mem counter is altered in more places.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

Applied.

Subject: [PATCH net-2.6.25 9/10][NETNS][FRAGS]: Make the LRU list per namespace.
Posted by Pavel Emelianov on Tue, 22 Jan 2008 14:08:54 GMT
View Forum Message <> Reply to Message

The inet_frags.lru_list is used for evicting only, so we have
to make it per-namespace, to evict only those fragments, who's
namespace exceeded its high threshold, but not the whole hash.
Besides, this helps to avoid long loops in evictor.

The spinlock is not per-namespace because it protects the
hash table as well, which is global.

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

---
include/net/inet_frag.h | 2 +-
net/ipv4/inet_fragment.c | 8 ++++----
net/ipv4/ip_fragment.c | 2 +-
net/ipv6/netfilter/nf_conntrack_reasm.c | 2 +-
net/ipv6/reassembly.c | 2 +-
5 files changed, 8 insertions(+), 8 deletions(-)

diff --git a/include/net/inet_frag.h b/include/net/inet_frag.h
index 1917fbe..3695ff4 100644
--- a/include/net/inet_frag.h

```
+++ b/include/net/inet_frag.h
@@ -4,6 +4,7 @@
 struct netns_frags {
  int   nqueues;
  atomic_t  mem;
+ struct list_head lru_list;

  /* sysctls */
  int   timeout;
@@ -32,7 +33,6 @@ struct inet_frag_queue {
 #define INETFRAGS_HASHSZ  64

 struct inet_frags {
- struct list_head lru_list;
  struct hlist_head hash[INETFRAGS_HASHSZ];
  rwlock_t  lock;
  u32   rnd;
diff --git a/net/ipv4/inet_fragment.c b/net/ipv4/inet_fragment.c
index fcf5252..f1b95e1 100644
--- a/net/ipv4/inet_fragment.c
+++ b/net/ipv4/inet_fragment.c
@@ -57,7 +57,6 @@ void inet_frags_init(struct inet_frags *f)
  for (i = 0; i < INETFRAGS_HASHSZ; i++)
   INIT_HLIST_HEAD(&f->hash[i]);

- INIT_LIST_HEAD(&f->lru_list);
  rwlock_init(&f->lock);

  f->rnd = (u32) ((num_physpages ^ (num_physpages>>7)) ^
@@ -74,6 +73,7 @@ void inet_frags_init_net(struct netns_frags *nf)
 {
  nf->nqueues = 0;
  atomic_set(&nf->mem, 0);
+ INIT_LIST_HEAD(&nf->lru_list);
 }
 EXPORT_SYMBOL(inet_frags_init_net);

@@ -156,12 +156,12 @@ int inet_frag_evictor(struct netns_frags *nf, struct inet_frags *f)
  work = atomic_read(&nf->mem) - nf->low_thresh;
  while (work > 0) {
   read_lock(&f->lock);
-  if (list_empty(&f->lru_list)) {
+  if (list_empty(&nf->lru_list)) {
    read_unlock(&f->lock);
    break;
   }

-  q = list_first_entry(&f->lru_list,
```

```
+  q = list_first_entry(&nf->lru_list,
     struct inet_frag_queue, lru_list);
   atomic_inc(&q->refcnt);
   read_unlock(&f->lock);
@@ -211,7 +211,7 @@ static struct inet_frag_queue *inet_frag_intern(struct netns_frags *nf,

  atomic_inc(&qp->refcnt);
  hlist_add_head(&qp->list, &f->hash[hash]);
- list_add_tail(&qp->lru_list, &f->lru_list);
+ list_add_tail(&qp->lru_list, &nf->lru_list);
  nf->nqueues++;
  write_unlock(&f->lock);
  return qp;
diff --git a/net/ipv4/ip_fragment.c b/net/ipv4/ip_fragment.c
index 00646ed..29b4b09 100644
--- a/net/ipv4/ip_fragment.c
+++ b/net/ipv4/ip_fragment.c
@@ -441,7 +441,7 @@ static int ip_frag_queue(struct ipq *qp, struct sk_buff *skb)
   return ip_frag_reasm(qp, prev, dev);

  write_lock(&ip4_frags.lock);
- list_move_tail(&qp->q.lru_list, &ip4_frags.lru_list);
+ list_move_tail(&qp->q.lru_list, &qp->q.net->lru_list);
  write_unlock(&ip4_frags.lock);
  return -EINPROGRESS;

diff --git a/net/ipv6/netfilter/nf_conntrack_reasm.c b/net/ipv6/netfilter/nf_conntrack_reasm.c
index 6eed991..022da6c 100644
--- a/net/ipv6/netfilter/nf_conntrack_reasm.c
+++ b/net/ipv6/netfilter/nf_conntrack_reasm.c
@@ -385,7 +385,7 @@ static int nf_ct_frag6_queue(struct nf_ct_frag6_queue *fq, struct sk_buff
*skb,
   fq->q.last_in |= FIRST_IN;
  }
  write_lock(&nf_frags.lock);
- list_move_tail(&fq->q.lru_list, &nf_frags.lru_list);
+ list_move_tail(&fq->q.lru_list, &nf_init_frags.lru_list);
  write_unlock(&nf_frags.lock);
  return 0;

diff --git a/net/ipv6/reassembly.c b/net/ipv6/reassembly.c
index 8520700..0c4bc46 100644
--- a/net/ipv6/reassembly.c
+++ b/net/ipv6/reassembly.c
@@ -424,7 +424,7 @@ static int ip6_frag_queue(struct frag_queue *fq, struct sk_buff *skb,
   return ip6_frag_reasm(fq, prev, dev);

  write_lock(&ip6_frags.lock);
```

```
- list_move_tail(&fq->q.lru_list, &ip6_frags.lru_list);
+ list_move_tail(&fq->q.lru_list, &fq->q.net->lru_list);
  write_unlock(&ip6_frags.lock);
  return -1;

--
1.5.3.4
```

Subject: Re: [PATCH net-2.6.25 5/10][NETNS][FRAGS]: Duplicate sysctl tables for new namespaces.
Posted by davem on Tue, 22 Jan 2008 14:09:09 GMT
View Forum Message <> Reply to Message

From: Pavel Emelyanov <xemul@openvz.org>
Date: Tue, 22 Jan 2008 17:01:02 +0300

> Each namespace has to have own tables to tune their
> different parameters, so duplicate the tables and
> register them.
>
> All the tables in sub-namespaces are temporarily made
> read-only.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

Applied.

Subject: Re: [PATCH net-2.6.25 6/10][NETNS][FRAGS]: Make the net.ipv4.ipfrag_timeout work in namespaces.
Posted by davem on Tue, 22 Jan 2008 14:09:47 GMT
View Forum Message <> Reply to Message

From: Pavel Emelyanov <xemul@openvz.org>
Date: Tue, 22 Jan 2008 17:02:17 +0300

> Move it to the netns_frags, adjust the usage and
> make the appropriate ctl table writable.
>
> Now fragment, that live in different namespaces can
> live for different times.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

Applied.

Subject: Re: [PATCH net-2.6.25 7/10][NETNS][FRAGS]: Make thresholds work in namespaces.
Posted by davem on Tue, 22 Jan 2008 14:10:25 GMT
View Forum Message <> Reply to Message

From: Pavel Emelyanov <xemul@openvz.org>
Date: Tue, 22 Jan 2008 17:05:47 +0300

> This is the same as with the timeout variable.
>
> Currently, after exceeding the high threshold _all_
> the fragments are evicted, but it will be fixed in
> later patch.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

Applied.

Subject: [PATCH net-2.6.25 10/10][NETNS][FRAGS]: Make the pernet subsystem for fragments.
Posted by Pavel Emelianov on Tue, 22 Jan 2008 14:10:51 GMT
View Forum Message <> Reply to Message

On namespace start we mainly prepare the ctl variables.

When the namespace is stopped we have to kill all the
fragments that point to this namespace.
The inet_frags_exit_net() handles it.

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

---
 include/net/inet_frag.h | 1 +
 net/ipv4/inet_fragment.c | 7 +++++++
 net/ipv4/ip_fragment.c | 13 ++++++++++++-
 net/ipv6/reassembly.c | 14 ++++++++++++++
 4 files changed, 33 insertions(+), 2 deletions(-)

diff --git a/include/net/inet_frag.h b/include/net/inet_frag.h
index 3695ff4..7374251 100644
--- a/include/net/inet_frag.h
+++ b/include/net/inet_frag.h
@@ -54,6 +54,7 @@ void inet_frags_init(struct inet_frags *);
 void inet_frags_fini(struct inet_frags *);

 void inet_frags_init_net(struct netns_frags *nf);
+void inet_frags_exit_net(struct netns_frags *nf, struct inet_frags *f);

```
 void inet_frag_kill(struct inet_frag_queue *q, struct inet_frags *f);
 void inet_frag_destroy(struct inet_frag_queue *q,
diff --git a/net/ipv4/inet_fragment.c b/net/ipv4/inet_fragment.c
index f1b95e1..724d69a 100644
--- a/net/ipv4/inet_fragment.c
+++ b/net/ipv4/inet_fragment.c
@@ -83,6 +83,13 @@ void inet_frags_fini(struct inet_frags *f)
 }
 EXPORT_SYMBOL(inet_frags_fini);

+void inet_frags_exit_net(struct netns_frags *nf, struct inet_frags *f)
+{
+ nf->low_thresh = 0;
+ inet_frag_evictor(nf, f);
+}
+EXPORT_SYMBOL(inet_frags_exit_net);
+
 static inline void fq_unlink(struct inet_frag_queue *fq, struct inet_frags *f)
 {
  write_lock(&f->lock);
diff --git a/net/ipv4/ip_fragment.c b/net/ipv4/ip_fragment.c
index 29b4b09..a2e92f9 100644
--- a/net/ipv4/ip_fragment.c
+++ b/net/ipv4/ip_fragment.c
@@ -713,9 +713,20 @@ static int ipv4_frags_init_net(struct net *net)
  return ip4_frags_ctl_register(net);
 }

+static void ipv4_frags_exit_net(struct net *net)
+{
+ ip4_frags_ctl_unregister(net);
+ inet_frags_exit_net(&net->ipv4.frags, &ip4_frags);
+}
+
+static struct pernet_operations ip4_frags_ops = {
+ .init = ipv4_frags_init_net,
+ .exit = ipv4_frags_exit_net,
+};
+
 void __init ipfrag_init(void)
 {
- ipv4_frags_init_net(&init_net);
+ register_pernet_subsys(&ip4_frags_ops);
  ip4_frags.hashfn = ip4_hashfn;
  ip4_frags.constructor = ip4_frag_init;
  ip4_frags.destructor = ip4_frag_free;
diff --git a/net/ipv6/reassembly.c b/net/ipv6/reassembly.c
```

```
index 0c4bc46..f936d04 100644
--- a/net/ipv6/reassembly.c
+++ b/net/ipv6/reassembly.c
@@ -728,6 +728,17 @@ static int ipv6_frags_init_net(struct net *net)
  return ip6_frags_sysctl_register(net);
 }

+static void ipv6_frags_exit_net(struct net *net)
+{
+ ip6_frags_sysctl_unregister(net);
+ inet_frags_exit_net(&net->ipv6.frags, &ip6_frags);
+}
+
+static struct pernet_operations ip6_frags_ops = {
+ .init = ipv6_frags_init_net,
+ .exit = ipv6_frags_exit_net,
+};
+
 int __init ipv6_frag_init(void)
 {
  int ret;
@@ -736,7 +747,7 @@ int __init ipv6_frag_init(void)
  if (ret)
   goto out;

- ipv6_frags_init_net(&init_net);
+ register_pernet_subsys(&ip6_frags_ops);

  ip6_frags.hashfn = ip6_hashfn;
  ip6_frags.constructor = ip6_frag_init;
@@ -754,5 +765,6 @@ out:
 void ipv6_frag_exit(void)
 {
  inet_frags_fini(&ip6_frags);
+ unregister_pernet_subsys(&ip6_frags_ops);
  inet6_del_protocol(&frag_protocol, IPPROTO_FRAGMENT);
 }
--
1.5.3.4
```

Subject: Re: [PATCH net-2.6.25 8/10][NETNS][FRAGS]: Isolate the secret interval from namespaces.
Posted by davem on Tue, 22 Jan 2008 14:11:16 GMT
View Forum Message <> Reply to Message

From: Pavel Emelyanov <xemul@openvz.org>
Date: Tue, 22 Jan 2008 17:07:10 +0300

> Since we have one hashtable to lookup the fragment, having
> different secret_interval-s for hash rebuild doesn't make
> sense, so move this one to inet_frags.
>
> The inet_frags_ctl becomes empty after this, so remove it.
> The appropriate ctl table is kept read-only in namespaces.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

Applied.

Subject: Re: [PATCH net-2.6.25 9/10][NETNS][FRAGS]: Make the LRU list per namespace.
Posted by davem on Tue, 22 Jan 2008 14:11:58 GMT
View Forum Message <> Reply to Message

From: Pavel Emelyanov <xemul@openvz.org>
Date: Tue, 22 Jan 2008 17:08:54 +0300

> The inet_frags.lru_list is used for evicting only, so we have
> to make it per-namespace, to evict only those fragments, who's
> namespace exceeded its high threshold, but not the whole hash.
> Besides, this helps to avoid long loops in evictor.
>
> The spinlock is not per-namespace because it protects the
> hash table as well, which is global.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

Applied.

Subject: Re: [PATCH net-2.6.25 10/10][NETNS][FRAGS]: Make the pernet subsystem for fragments.
Posted by davem on Tue, 22 Jan 2008 14:12:52 GMT
View Forum Message <> Reply to Message

From: Pavel Emelyanov <xemul@openvz.org>
Date: Tue, 22 Jan 2008 17:10:51 +0300

> On namespace start we mainly prepare the ctl variables.
>
> When the namespace is stopped we have to kill all the
> fragments that point to this namespace.
> The inet_frags_exit_net() handles it.

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

Also applied, thanks a lot!