Subject: [PATCH 1/7] Omit double hash calculations in xxx_frag_intern
Posted by Pavel Emelianov on Tue, 16 Oct 2007 13:48:46 GMT
View Forum Message <> Reply to Message

Since the hash value is already calculated in xxx_find, we can
simply use it later. This is already done in netfilter code,
so make the same in ipv4 and ipv6.

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

>From 5bcb464538cf504a9f6cf3be33dbd1a5ff18b693 Mon Sep 17 00:00:00 2001
From: Pavel <pavel@xemulnb.sw.ru>
Date: Tue, 16 Oct 2007 14:40:30 +0400
Subject: [PATCH 1/7] Pass the hash value as an extra argument

---
 net/ipv4/ip_fragment.c | 11 ++++-------
 net/ipv6/reassembly.c | 11 +++++------
 2 files changed, 9 insertions(+), 13 deletions(-)

diff --git a/net/ipv4/ip_fragment.c b/net/ipv4/ip_fragment.c
index 443b3f8..d12a18b 100644
--- a/net/ipv4/ip_fragment.c
+++ b/net/ipv4/ip_fragment.c
@@ -212,17 +212,14 @@ out:

 /* Creation primitives. */

-static struct ipq *ip_frag_intern(struct ipq *qp_in)
+static struct ipq *ip_frag_intern(struct ipq *qp_in, unsigned int hash)
 {
  struct ipq *qp;
 #ifdef CONFIG_SMP
  struct hlist_node *n;
 #endif
- unsigned int hash;

  write_lock(&ip4_frags.lock);
- hash = ipqhashfn(qp_in->id, qp_in->saddr, qp_in->daddr,
-    qp_in->protocol);
 #ifdef CONFIG_SMP
  /* With SMP race we have to recheck hash table, because
   * such entry could be created on other cpu, while we
@@ -257,7 +254,7 @@ static struct ipq *ip_frag_intern(struct ipq *qp_in)
 }

```
 /* Add an entry to the 'ipq' queue for a newly received IP datagram. */
-static struct ipq *ip_frag_create(struct iphdr *iph, u32 user)
+static struct ipq *ip_frag_create(struct iphdr *iph, u32 user, unsigned int h)
 {
  struct ipq *qp;

@@ -278,7 +275,7 @@ static struct ipq *ip_frag_create(struct iphdr *iph, u32 user)
  spin_lock_init(&qp->q.lock);
  atomic_set(&qp->q.refcnt, 1);

- return ip_frag_intern(qp);
+ return ip_frag_intern(qp, h);

 out_nomem:
  LIMIT_NETDEBUG(KERN_ERR "ip_frag_create: no memory left !\n");
@@ -313,7 +310,7 @@ static inline struct ipq *ip_find(struct iphdr *iph, u32 user)
  }
  read_unlock(&ip4_frags.lock);

- return ip_frag_create(iph, user);
+ return ip_frag_create(iph, user, hash);
 }

 /* Is the fragment too far ahead to be part of ipq? */
diff --git a/net/ipv6/reassembly.c b/net/ipv6/reassembly.c
index 6ad19cf..0a1bf43 100644
--- a/net/ipv6/reassembly.c
+++ b/net/ipv6/reassembly.c
@@ -233,16 +233,15 @@ out:
 /* Creation primitives. */


-static struct frag_queue *ip6_frag_intern(struct frag_queue *fq_in)
+static struct frag_queue *ip6_frag_intern(struct frag_queue *fq_in,
+  unsigned int hash)
 {
  struct frag_queue *fq;
- unsigned int hash;
 #ifdef CONFIG_SMP
  struct hlist_node *n;
 #endif

  write_lock(&ip6_frags.lock);
- hash = ip6qhashfn(fq_in->id, &fq_in->saddr, &fq_in->daddr);
 #ifdef CONFIG_SMP
  hlist_for_each_entry(fq, n, &ip6_frags.hash[hash], q.list) {
   if (fq->id == fq_in->id &&
@@ -273,7 +272,7 @@ static struct frag_queue *ip6_frag_intern(struct frag_queue *fq_in)
```

```
 static struct frag_queue *
 ip6_frag_create(__be32 id, struct in6_addr *src, struct in6_addr *dst,
-  struct inet6_dev *idev)
+  struct inet6_dev *idev, unsigned int hash)
 {
  struct frag_queue *fq;

@@ -290,7 +289,7 @@ ip6_frag_create(__be32 id, struct in6_addr *src, struct in6_addr *dst,
  spin_lock_init(&fq->q.lock);
  atomic_set(&fq->q.refcnt, 1);

- return ip6_frag_intern(fq);
+ return ip6_frag_intern(fq, hash);

 oom:
  IP6_INC_STATS_BH(idev, IPSTATS_MIB_REASMFAILS);
@@ -318,7 +317,7 @@ fq_find(__be32 id, struct in6_addr *src, struct in6_addr *dst,
  }
  read_unlock(&ip6_frags.lock);

- return ip6_frag_create(id, src, dst, idev);
+ return ip6_frag_create(id, src, dst, idev, hash);
 }


--
1.5.3.4
```