Subject: [PATCH 0/7] Next step in consolidating the IP fragment management
Posted by Pavel Emelianov on Tue, 16 Oct 2007 13:46:23 GMT
View Forum Message <> Reply to Message

This set concerns the xxx_find() routine only. This one
is used to obtain the fragment queue depending on some
criteria, and this can be consolidated.

The consolidation goes step-by-step, consolidating one
sub-routine at a time, so some functions, exports and
callbacks that are introduced in patches are temporary
and are removed in some subsequent patch.

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

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
```

Subject: [PATCH 2/7] Consolidate xxx_frag_intern

This routine checks for the existence of a given entry
in the hash table and inserts the new one if needed.

The ->equal callback is used to compare two frag_queue-s
together, but this one is temporary and will be removed
later. The netfilter code and the ipv6 one use the same
routine to compare frags.

The inet_frag_intern() always returns non-NULL pointer,
so convert the inet_frag_queue into protocol specific
one (with the container_of) without any checks.

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

```
diff --git a/include/net/inet_frag.h b/include/net/inet_frag.h
index 911c2cd..133e187 100644
--- a/include/net/inet_frag.h
+++ b/include/net/inet_frag.h
@@ -41,6 +41,8 @@ struct inet_frags {
  unsigned int  (*hashfn)(struct inet_frag_queue *);
  void   (*destructor)(struct inet_frag_queue *);
  void   (*skb_free)(struct sk_buff *);
+ int   (*equal)(struct inet_frag_queue *q1,
+       struct inet_frag_queue *q2);
 };

 void inet_frags_init(struct inet_frags *);
@@ -50,6 +52,8 @@ 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_intern(struct inet_frag_queue *q,
+  struct inet_frags *f, unsigned int hash);

 static inline void inet_frag_put(struct inet_frag_queue *q, struct inet_frags *f)
 {
diff --git a/include/net/ipv6.h b/include/net/ipv6.h
index cc796cb..ff12697 100644
--- a/include/net/ipv6.h
+++ b/include/net/ipv6.h
@@ -377,6 +377,9 @@ static inline int ipv6_prefix_equal(const struct in6_addr *a1,
        prefixlen);
 }
```

```
+struct inet_frag_queue;
+int ip6_frag_equal(struct inet_frag_queue *q1, struct inet_frag_queue *q2);
+
 static inline int ipv6_addr_any(const struct in6_addr *a)
 {
  return ((a->s6_addr32[0] | a->s6_addr32[1] |
diff --git a/net/ipv4/inet_fragment.c b/net/ipv4/inet_fragment.c
index 484cf51..15054eb 100644
--- a/net/ipv4/inet_fragment.c
+++ b/net/ipv4/inet_fragment.c
@@ -172,3 +172,40 @@ int inet_frag_evictor(struct inet_frags *f)
  return evicted;
 }
 EXPORT_SYMBOL(inet_frag_evictor);
+
+struct inet_frag_queue *inet_frag_intern(struct inet_frag_queue *qp_in,
+  struct inet_frags *f, unsigned int hash)
+{
+ struct inet_frag_queue *qp;
+#ifdef CONFIG_SMP
+ struct hlist_node *n;
+#endif
+
+ write_lock(&f->lock);
+#ifdef CONFIG_SMP
+ /* With SMP race we have to recheck hash table, because
+  * such entry could be created on other cpu, while we
+  * promoted read lock to write lock.
+  */
+ hlist_for_each_entry(qp, n, &f->hash[hash], list) {
+  if (f->equal(qp, qp_in)) {
+   atomic_inc(&qp->refcnt);
+   write_unlock(&f->lock);
+   qp_in->last_in |= COMPLETE;
+   inet_frag_put(qp_in, f);
+   return qp;
+  }
+ }
+#endif
+ qp = qp_in;
+ if (!mod_timer(&qp->timer, jiffies + f->ctl->timeout))
+  atomic_inc(&qp->refcnt);
+
+ atomic_inc(&qp->refcnt);
+ hlist_add_head(&qp->list, &f->hash[hash]);
+ list_add_tail(&qp->lru_list, &f->lru_list);
+ f->nqueues++;
+ write_unlock(&f->lock);
```

```
+ return qp;
+}
+EXPORT_SYMBOL(inet_frag_intern);
diff --git a/net/ipv4/ip_fragment.c b/net/ipv4/ip_fragment.c
index d12a18b..4b1bbbe 100644
--- a/net/ipv4/ip_fragment.c
+++ b/net/ipv4/ip_fragment.c
@@ -123,6 +123,20 @@ static unsigned int ip4_hashfn(struct inet_frag_queue *q)
  return ipqhashfn(ipq->id, ipq->saddr, ipq->daddr, ipq->protocol);
 }

+static int ip4_frag_equal(struct inet_frag_queue *q1,
+  struct inet_frag_queue *q2)
+{
+ struct ipq *qp1, *qp2;
+
+ qp1 = container_of(q1, struct ipq, q);
+ qp2 = container_of(q2, struct ipq, q);
+ return (qp1->id == qp2->id &&
+    qp1->saddr == qp2->saddr &&
+    qp1->daddr == qp2->daddr &&
+    qp1->protocol == qp2->protocol &&
+    qp1->user == qp2->user);
+}
+
 /* Memory Tracking Functions. */
 static __inline__ void frag_kfree_skb(struct sk_buff *skb, int *work)
 {
@@ -214,43 +228,10 @@ out:

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

- write_lock(&ip4_frags.lock);
-#ifdef CONFIG_SMP
- /* With SMP race we have to recheck hash table, because
-  * such entry could be created on other cpu, while we
-  * promoted read lock to write lock.
-  */
- hlist_for_each_entry(qp, n, &ip4_frags.hash[hash], q.list) {
-  if (qp->id == qp_in->id  &&
-      qp->saddr == qp_in->saddr &&
-      qp->daddr == qp_in->daddr &&
```

```
-      qp->protocol == qp_in->protocol &&
-      qp->user == qp_in->user) {
-    atomic_inc(&qp->q.refcnt);
-    write_unlock(&ip4_frags.lock);
-    qp_in->q.last_in |= COMPLETE;
-    ipq_put(qp_in);
-    return qp;
-  }
- }
-#endif
- qp = qp_in;
-
- if (!mod_timer(&qp->q.timer, jiffies + ip4_frags_ctl.timeout))
-   atomic_inc(&qp->q.refcnt);
-
- atomic_inc(&qp->q.refcnt);
- hlist_add_head(&qp->q.list, &ip4_frags.hash[hash]);
- INIT_LIST_HEAD(&qp->q.lru_list);
- list_add_tail(&qp->q.lru_list, &ip4_frags.lru_list);
- ip4_frags.nqueues++;
- write_unlock(&ip4_frags.lock);
- return qp;
+ q = inet_frag_intern(&qp_in->q, &ip4_frags, hash);
+ return container_of(q, struct ipq, q);
 }

 /* Add an entry to the 'ipq' queue for a newly received IP datagram. */
@@ -671,6 +652,7 @@ void __init ipfrag_init(void)
   ip4_frags.destructor = ip4_frag_free;
   ip4_frags.skb_free = NULL;
   ip4_frags.qsize = sizeof(struct ipq);
+ ip4_frags.equal = ip4_frag_equal;
   inet_frags_init(&ip4_frags);
 }

diff --git a/net/ipv6/netfilter/nf_conntrack_reasm.c b/net/ipv6/netfilter/nf_conntrack_reasm.c
index 726fafd..d7dc444 100644
--- a/net/ipv6/netfilter/nf_conntrack_reasm.c
+++ b/net/ipv6/netfilter/nf_conntrack_reasm.c
@@ -187,37 +187,10 @@ out:
 static struct nf_ct_frag6_queue *nf_ct_frag6_intern(unsigned int hash,
         struct nf_ct_frag6_queue *fq_in)
 {
- struct nf_ct_frag6_queue *fq;
-#ifdef CONFIG_SMP
- struct hlist_node *n;
-#endif
-
```

```
- write_lock(&nf_frags.lock);
-#ifdef CONFIG_SMP
- hlist_for_each_entry(fq, n, &nf_frags.hash[hash], q.list) {
-  if (fq->id == fq_in->id &&
-      ipv6_addr_equal(&fq_in->saddr, &fq->saddr) &&
-      ipv6_addr_equal(&fq_in->daddr, &fq->daddr)) {
-   atomic_inc(&fq->q.refcnt);
-   write_unlock(&nf_frags.lock);
-   fq_in->q.last_in |= COMPLETE;
-   fq_put(fq_in);
-   return fq;
-  }
- }
-#endif
- fq = fq_in;
+ struct inet_frag_queue *q;

- if (!mod_timer(&fq->q.timer, jiffies + nf_frags_ctl.timeout))
-  atomic_inc(&fq->q.refcnt);
-
- atomic_inc(&fq->q.refcnt);
- hlist_add_head(&fq->q.list, &nf_frags.hash[hash]);
- INIT_LIST_HEAD(&fq->q.lru_list);
- list_add_tail(&fq->q.lru_list, &nf_frags.lru_list);
- nf_frags.nqueues++;
- write_unlock(&nf_frags.lock);
- return fq;
+ q = inet_frag_intern(&fq_in->q, &nf_frags, hash);
+ return container_of(q, struct nf_ct_frag6_queue, q);
 }


@@ -752,6 +725,7 @@ int nf_ct_frag6_init(void)
  nf_frags.destructor = nf_frag_free;
  nf_frags.skb_free = nf_skb_free;
  nf_frags.qsize = sizeof(struct nf_ct_frag6_queue);
+ nf_frags.equal = ip6_frag_equal;
  inet_frags_init(&nf_frags);

  return 0;
diff --git a/net/ipv6/reassembly.c b/net/ipv6/reassembly.c
index 0a1bf43..73ea204 100644
--- a/net/ipv6/reassembly.c
+++ b/net/ipv6/reassembly.c
@@ -143,6 +143,18 @@ static unsigned int ip6_hashfn(struct inet_frag_queue *q)
  return ip6qhashfn(fq->id, &fq->saddr, &fq->daddr);
 }
```

```
+int ip6_frag_equal(struct inet_frag_queue *q1, struct inet_frag_queue *q2)
+{
+ struct frag_queue *fq1, *fq2;
+
+ fq1 = container_of(q1, struct frag_queue, q);
+ fq2 = container_of(q2, struct frag_queue, q);
+ return (fq1->id == fq2->id &&
+   ipv6_addr_equal(&fq2->saddr, &fq1->saddr) &&
+   ipv6_addr_equal(&fq2->daddr, &fq1->daddr));
+}
+EXPORT_SYMBOL(ip6_frag_equal);
+
 /* Memory Tracking Functions. */
 static inline void frag_kfree_skb(struct sk_buff *skb, int *work)
 {
@@ -236,37 +248,10 @@ out:
 static struct frag_queue *ip6_frag_intern(struct frag_queue *fq_in,
   unsigned int hash)
 {
- struct frag_queue *fq;
-#ifdef CONFIG_SMP
- struct hlist_node *n;
-#endif
-
- write_lock(&ip6_frags.lock);
-#ifdef CONFIG_SMP
- hlist_for_each_entry(fq, n, &ip6_frags.hash[hash], q.list) {
-  if (fq->id == fq_in->id &&
-      ipv6_addr_equal(&fq_in->saddr, &fq->saddr) &&
-      ipv6_addr_equal(&fq_in->daddr, &fq->daddr)) {
-   atomic_inc(&fq->q.refcnt);
-   write_unlock(&ip6_frags.lock);
-   fq_in->q.last_in |= COMPLETE;
-   fq_put(fq_in);
-   return fq;
-  }
- }
-#endif
- fq = fq_in;
-
- if (!mod_timer(&fq->q.timer, jiffies + ip6_frags_ctl.timeout))
-  atomic_inc(&fq->q.refcnt);
+ struct inet_frag_queue *q;

- atomic_inc(&fq->q.refcnt);
- hlist_add_head(&fq->q.list, &ip6_frags.hash[hash]);
- INIT_LIST_HEAD(&fq->q.lru_list);
- list_add_tail(&fq->q.lru_list, &ip6_frags.lru_list);
```

```
- ip6_frags.nqueues++;
- write_unlock(&ip6_frags.lock);
- return fq;
+ q = inet_frag_intern(&fq_in->q, &ip6_frags, hash);
+ return container_of(q, struct frag_queue, q);
 }


@@ -699,5 +684,6 @@ void __init ipv6_frag_init(void)
  ip6_frags.destructor = ip6_frag_free;
  ip6_frags.skb_free = NULL;
  ip6_frags.qsize = sizeof(struct frag_queue);
+ ip6_frags.equal = ip6_frag_equal;
  inet_frags_init(&ip6_frags);
 }
--
1.5.3.4
```

Subject: [PATCH 3/7] Consolidate xxx_frag_alloc()
Posted by Pavel Emelianov on Tue, 16 Oct 2007 13:57:44 GMT
View Forum Message <> Reply to Message

Just perform the kzalloc() allocation and setup common
fields in the inet_frag_queue(). Then return the result
to the caller to initialize the rest.

The inet_frag_alloc() may return NULL, so check the
return value before doing the container_of(). This looks
ugly, but the xxx_frag_alloc() will be removed soon.

The xxx_expire() timer callbacks are patches,
because the argument is now the inet_frag_queue, not
the protocol specific queue.

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

```
diff --git a/include/net/inet_frag.h b/include/net/inet_frag.h
index 133e187..412b858 100644
--- a/include/net/inet_frag.h
+++ b/include/net/inet_frag.h
@@ -43,6 +43,7 @@ struct inet_frags {
  void   (*skb_free)(struct sk_buff *);
  int   (*equal)(struct inet_frag_queue *q1,
       struct inet_frag_queue *q2);
+ void   (*frag_expire)(unsigned long data);
```

```
 };

 void inet_frags_init(struct inet_frags *);
@@ -54,6 +55,7 @@ void inet_frag_destroy(struct inet_frag_queue *q,
 int inet_frag_evictor(struct inet_frags *f);
 struct inet_frag_queue *inet_frag_intern(struct inet_frag_queue *q,
   struct inet_frags *f, unsigned int hash);
+struct inet_frag_queue *inet_frag_alloc(struct inet_frags *f);

 static inline void inet_frag_put(struct inet_frag_queue *q, struct inet_frags *f)
 {
diff --git a/net/ipv4/inet_fragment.c b/net/ipv4/inet_fragment.c
index 15054eb..22539fb 100644
--- a/net/ipv4/inet_fragment.c
+++ b/net/ipv4/inet_fragment.c
@@ -209,3 +209,20 @@ struct inet_frag_queue *inet_frag_intern(struct inet_frag_queue *qp_in,
  return qp;
 }
 EXPORT_SYMBOL(inet_frag_intern);
+
+struct inet_frag_queue *inet_frag_alloc(struct inet_frags *f)
+{
+ struct inet_frag_queue *q;
+
+ q = kzalloc(f->qsize, GFP_ATOMIC);
+ if (q == NULL)
+  return NULL;
+
+ atomic_add(f->qsize, &f->mem);
+ setup_timer(&q->timer, f->frag_expire, (unsigned long)q);
+ spin_lock_init(&q->lock);
+ atomic_set(&q->refcnt, 1);
+
+ return q;
+}
+EXPORT_SYMBOL(inet_frag_alloc);
diff --git a/net/ipv4/ip_fragment.c b/net/ipv4/ip_fragment.c
index 4b1bbbe..2cf8cdc 100644
--- a/net/ipv4/ip_fragment.c
+++ b/net/ipv4/ip_fragment.c
@@ -158,12 +158,10 @@ static __inline__ void ip4_frag_free(struct inet_frag_queue *q)

 static __inline__ struct ipq *frag_alloc_queue(void)
 {
- struct ipq *qp = kzalloc(sizeof(struct ipq), GFP_ATOMIC);
+ struct inet_frag_queue *q;

- if (!qp)
```

```
-  return NULL;
- atomic_add(sizeof(struct ipq), &ip4_frags.mem);
- return qp;
+ q = inet_frag_alloc(&ip4_frags);
+ return q ? container_of(q, struct ipq, q) : NULL;
 }


@@ -199,7 +197,9 @@ static void ip_evictor(void)
  */
 static void ip_expire(unsigned long arg)
 {
- struct ipq *qp = (struct ipq *) arg;
+ struct ipq *qp;
+
+ qp = container_of((struct inet_frag_queue *) arg, struct ipq, q);

  spin_lock(&qp->q.lock);

@@ -249,13 +249,6 @@ static struct ipq *ip_frag_create(struct iphdr *iph, u32 user, unsigned int
h)
  qp->user = user;
  qp->peer = sysctl_ipfrag_max_dist ? inet_getpeer(iph->saddr, 1) : NULL;

- /* Initialize a timer for this entry. */
- init_timer(&qp->q.timer);
- qp->q.timer.data = (unsigned long) qp; /* pointer to queue */
- qp->q.timer.function = ip_expire;  /* expire function */
- spin_lock_init(&qp->q.lock);
- atomic_set(&qp->q.refcnt, 1);
-
  return ip_frag_intern(qp, h);

 out_nomem:
@@ -653,6 +646,7 @@ void __init ipfrag_init(void)
  ip4_frags.skb_free = NULL;
  ip4_frags.qsize = sizeof(struct ipq);
  ip4_frags.equal = ip4_frag_equal;
+ ip4_frags.frag_expire = ip_expire;
  inet_frags_init(&ip4_frags);
 }

diff --git a/net/ipv6/netfilter/nf_conntrack_reasm.c b/net/ipv6/netfilter/nf_conntrack_reasm.c
index d7dc444..29a42d2 100644
--- a/net/ipv6/netfilter/nf_conntrack_reasm.c
+++ b/net/ipv6/netfilter/nf_conntrack_reasm.c
@@ -137,13 +137,10 @@ static void nf_frag_free(struct inet_frag_queue *q)
```

```
 static inline struct nf_ct_frag6_queue *frag_alloc_queue(void)
 {
- struct nf_ct_frag6_queue *fq;
+ struct inet_frag_queue *q;

- fq = kzalloc(sizeof(struct nf_ct_frag6_queue), GFP_ATOMIC);
- if (fq == NULL)
-  return NULL;
- atomic_add(sizeof(struct nf_ct_frag6_queue), &nf_frags.mem);
- return fq;
+ q = inet_frag_alloc(&nf_frags);
+ return q ? container_of(q, struct nf_ct_frag6_queue, q) : NULL;
 }

 /* Destruction primitives. */
@@ -168,7 +165,10 @@ static void nf_ct_frag6_evictor(void)

 static void nf_ct_frag6_expire(unsigned long data)
 {
- struct nf_ct_frag6_queue *fq = (struct nf_ct_frag6_queue *) data;
+ struct nf_ct_frag6_queue *fq;
+
+ fq = container_of((struct inet_frag_queue *)data,
+    struct nf_ct_frag6_queue, q);

  spin_lock(&fq->q.lock);

@@ -208,10 +208,6 @@ nf_ct_frag6_create(unsigned int hash, __be32 id, struct in6_addr
*src,       str
  ipv6_addr_copy(&fq->saddr, src);
  ipv6_addr_copy(&fq->daddr, dst);

- setup_timer(&fq->q.timer, nf_ct_frag6_expire, (unsigned long)fq);
- spin_lock_init(&fq->q.lock);
- atomic_set(&fq->q.refcnt, 1);
-
  return nf_ct_frag6_intern(hash, fq);

 oom:
@@ -726,6 +722,7 @@ int nf_ct_frag6_init(void)
  nf_frags.skb_free = nf_skb_free;
  nf_frags.qsize = sizeof(struct nf_ct_frag6_queue);
  nf_frags.equal = ip6_frag_equal;
+ nf_frags.frag_expire = nf_ct_frag6_expire;
  inet_frags_init(&nf_frags);

  return 0;
diff --git a/net/ipv6/reassembly.c b/net/ipv6/reassembly.c
```

```
index 73ea204..21913c7 100644
--- a/net/ipv6/reassembly.c
+++ b/net/ipv6/reassembly.c
@@ -171,12 +171,10 @@ static void ip6_frag_free(struct inet_frag_queue *fq)

 static inline struct frag_queue *frag_alloc_queue(void)
 {
- struct frag_queue *fq = kzalloc(sizeof(struct frag_queue), GFP_ATOMIC);
+ struct inet_frag_queue *q;

- if(!fq)
-  return NULL;
- atomic_add(sizeof(struct frag_queue), &ip6_frags.mem);
- return fq;
+ q = inet_frag_alloc(&ip6_frags);
+ return q ? container_of(q, struct frag_queue, q) : NULL;
 }

 /* Destruction primitives. */
@@ -205,9 +203,11 @@ static void ip6_evictor(struct inet6_dev *idev)

 static void ip6_frag_expire(unsigned long data)
 {
- struct frag_queue *fq = (struct frag_queue *) data;
+ struct frag_queue *fq;
  struct net_device *dev = NULL;

+ fq = container_of((struct inet_frag_queue *)data, struct frag_queue, q);
+
  spin_lock(&fq->q.lock);

  if (fq->q.last_in & COMPLETE)
@@ -268,12 +268,6 @@ ip6_frag_create(__be32 id, struct in6_addr *src, struct in6_addr *dst,
  ipv6_addr_copy(&fq->saddr, src);
  ipv6_addr_copy(&fq->daddr, dst);

- init_timer(&fq->q.timer);
- fq->q.timer.function = ip6_frag_expire;
- fq->q.timer.data = (long) fq;
- spin_lock_init(&fq->q.lock);
- atomic_set(&fq->q.refcnt, 1);
-
  return ip6_frag_intern(fq, hash);

 oom:
@@ -685,5 +679,6 @@ void __init ipv6_frag_init(void)
  ip6_frags.skb_free = NULL;
  ip6_frags.qsize = sizeof(struct frag_queue);
```

```
  ip6_frags.equal = ip6_frag_equal;
+ ip6_frags.frag_expire = ip6_frag_expire;
  inet_frags_init(&ip6_frags);
 }
--
1.5.3.4
```

Subject: [PATCH 4/7] Consolidate xxx_frag_create()
Posted by Pavel Emelianov on Tue, 16 Oct 2007 14:00:10 GMT
View Forum Message <> Reply to Message

This one uses the xxx_frag_intern() and xxx_frag_alloc()
routines, which are already consolidated, so remove them
from protocol code (as promised).

The ->constructor callback is used to init the rest of
the frag queue and it is the same for netfilter and ipv6.

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

```
diff --git a/include/net/inet_frag.h b/include/net/inet_frag.h
index 412b858..e33072b 100644
--- a/include/net/inet_frag.h
+++ b/include/net/inet_frag.h
@@ -39,6 +39,8 @@ struct inet_frags {
  struct inet_frags_ctl *ctl;

  unsigned int  (*hashfn)(struct inet_frag_queue *);
+ void   (*constructor)(struct inet_frag_queue *q,
+       void *arg);
  void   (*destructor)(struct inet_frag_queue *);
  void   (*skb_free)(struct sk_buff *);
  int   (*equal)(struct inet_frag_queue *q1,
@@ -53,9 +55,8 @@ 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_intern(struct inet_frag_queue *q,
-  struct inet_frags *f, unsigned int hash);
-struct inet_frag_queue *inet_frag_alloc(struct inet_frags *f);
+struct inet_frag_queue *inet_frag_create(struct inet_frags *f,
+  void *create_arg, unsigned int hash);

 static inline void inet_frag_put(struct inet_frag_queue *q, struct inet_frags *f)
 {
```

```
diff --git a/include/net/ipv6.h b/include/net/ipv6.h
index ff12697..9dc99bf 100644
--- a/include/net/ipv6.h
+++ b/include/net/ipv6.h
@@ -380,6 +380,14 @@ static inline int ipv6_prefix_equal(const struct in6_addr *a1,
 struct inet_frag_queue;
 int ip6_frag_equal(struct inet_frag_queue *q1, struct inet_frag_queue *q2);

+struct ip6_create_arg {
+ __be32 id;
+ struct in6_addr *src;
+ struct in6_addr *dst;
+};
+
+void ip6_frag_init(struct inet_frag_queue *q, void *a);
+
 static inline int ipv6_addr_any(const struct in6_addr *a)
 {
  return ((a->s6_addr32[0] | a->s6_addr32[1] |
diff --git a/net/ipv4/inet_fragment.c b/net/ipv4/inet_fragment.c
index 22539fb..0124885 100644
--- a/net/ipv4/inet_fragment.c
+++ b/net/ipv4/inet_fragment.c
@@ -173,7 +173,7 @@ int inet_frag_evictor(struct inet_frags *f)
 }
 EXPORT_SYMBOL(inet_frag_evictor);

-struct inet_frag_queue *inet_frag_intern(struct inet_frag_queue *qp_in,
+static struct inet_frag_queue *inet_frag_intern(struct inet_frag_queue *qp_in,
   struct inet_frags *f, unsigned int hash)
 {
  struct inet_frag_queue *qp;
@@ -208,9 +208,8 @@ struct inet_frag_queue *inet_frag_intern(struct inet_frag_queue *qp_in,
  write_unlock(&f->lock);
  return qp;
 }
-EXPORT_SYMBOL(inet_frag_intern);

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

@@ -218,6 +217,7 @@ struct inet_frag_queue *inet_frag_alloc(struct inet_frags *f)
  if (q == NULL)
   return NULL;

+ f->constructor(q, arg);
```

```
  atomic_add(f->qsize, &f->mem);
  setup_timer(&q->timer, f->frag_expire, (unsigned long)q);
  spin_lock_init(&q->lock);
@@ -225,4 +225,16 @@ struct inet_frag_queue *inet_frag_alloc(struct inet_frags *f)

  return q;
 }
-EXPORT_SYMBOL(inet_frag_alloc);
+
+struct inet_frag_queue *inet_frag_create(struct inet_frags *f, void *arg,
+  unsigned int hash)
+{
+ struct inet_frag_queue *q;
+
+ q = inet_frag_alloc(f, arg);
+ if (q == NULL)
+  return NULL;
+
+ return inet_frag_intern(q, f, hash);
+}
+EXPORT_SYMBOL(inet_frag_create);
diff --git a/net/ipv4/ip_fragment.c b/net/ipv4/ip_fragment.c
index 2cf8cdc..db29c3c 100644
--- a/net/ipv4/ip_fragment.c
+++ b/net/ipv4/ip_fragment.c
@@ -108,6 +108,11 @@ int ip_frag_mem(void)
 static int ip_frag_reasm(struct ipq *qp, struct sk_buff *prev,
     struct net_device *dev);

+struct ip4_create_arg {
+ struct iphdr *iph;
+ u32 user;
+};
+
 static unsigned int ipqhashfn(__be16 id, __be32 saddr, __be32 daddr, u8 prot)
 {
  return jhash_3words((__force u32)id << 16 | prot,
@@ -146,6 +151,20 @@ static __inline__ void frag_kfree_skb(struct sk_buff *skb, int *work)
  kfree_skb(skb);
 }

+static void ip4_frag_init(struct inet_frag_queue *q, void *a)
+{
+ struct ipq *qp = container_of(q, struct ipq, q);
+ struct ip4_create_arg *arg = a;
+
+ qp->protocol = arg->iph->protocol;
+ qp->id = arg->iph->id;
```

```
+ qp->saddr = arg->iph->saddr;
+ qp->daddr = arg->iph->daddr;
+ qp->user = arg->user;
+ qp->peer = sysctl_ipfrag_max_dist ?
+  inet_getpeer(arg->iph->saddr, 1) : NULL;
+}
+
 static __inline__ void ip4_frag_free(struct inet_frag_queue *q)
 {
  struct ipq *qp;
@@ -156,14 +175,6 @@ static __inline__ void ip4_frag_free(struct inet_frag_queue *q)
  kfree(qp);
 }

-static __inline__ struct ipq *frag_alloc_queue(void)
-{
- struct inet_frag_queue *q;
-
- q = inet_frag_alloc(&ip4_frags);
- return q ? container_of(q, struct ipq, q) : NULL;
-}
-

 /* Destruction primitives. */

@@ -226,30 +237,20 @@ out:

 /* Creation primitives. */

-static struct ipq *ip_frag_intern(struct ipq *qp_in, unsigned int hash)
-{
- struct inet_frag_queue *q;
-
- q = inet_frag_intern(&qp_in->q, &ip4_frags, hash);
- return container_of(q, struct ipq, q);
-}
-
 /* Add an entry to the 'ipq' queue for a newly received IP datagram. */
 static struct ipq *ip_frag_create(struct iphdr *iph, u32 user, unsigned int h)
 {
- struct ipq *qp;
+ struct inet_frag_queue *q;
+ struct ip4_create_arg arg;

- if ((qp = frag_alloc_queue()) == NULL)
-  goto out_nomem;
+ arg.iph = iph;
+ arg.user = user;
```

```
- qp->protocol = iph->protocol;
- qp->id = iph->id;
- qp->saddr = iph->saddr;
- qp->daddr = iph->daddr;
- qp->user = user;
- qp->peer = sysctl_ipfrag_max_dist ? inet_getpeer(iph->saddr, 1) : NULL;
+ q = inet_frag_create(&ip4_frags, &arg, h);
+ if (q == NULL)
+  goto out_nomem;

- return ip_frag_intern(qp, h);
+ return container_of(q, struct ipq, q);

 out_nomem:
  LIMIT_NETDEBUG(KERN_ERR "ip_frag_create: no memory left !\n");
@@ -642,6 +643,7 @@ void __init ipfrag_init(void)
 {
  ip4_frags.ctl = &ip4_frags_ctl;
  ip4_frags.hashfn = ip4_hashfn;
+ ip4_frags.constructor = ip4_frag_init;
  ip4_frags.destructor = ip4_frag_free;
  ip4_frags.skb_free = NULL;
  ip4_frags.qsize = sizeof(struct ipq);
diff --git a/net/ipv6/netfilter/nf_conntrack_reasm.c b/net/ipv6/netfilter/nf_conntrack_reasm.c
index 29a42d2..72451e2 100644
--- a/net/ipv6/netfilter/nf_conntrack_reasm.c
+++ b/net/ipv6/netfilter/nf_conntrack_reasm.c
@@ -135,14 +135,6 @@ static void nf_frag_free(struct inet_frag_queue *q)
  kfree(container_of(q, struct nf_ct_frag6_queue, q));
 }

-static inline struct nf_ct_frag6_queue *frag_alloc_queue(void)
-{
- struct inet_frag_queue *q;
-
- q = inet_frag_alloc(&nf_frags);
- return q ? container_of(q, struct nf_ct_frag6_queue, q) : NULL;
-}
-
 /* Destruction primitives. */

 static __inline__ void fq_put(struct nf_ct_frag6_queue *fq)
@@ -184,33 +176,25 @@ out:

 /* Creation primitives. */

-static struct nf_ct_frag6_queue *nf_ct_frag6_intern(unsigned int hash,
```

```
-       struct nf_ct_frag6_queue *fq_in)
+static struct nf_ct_frag6_queue *
+nf_ct_frag6_create(unsigned int hash, __be32 id, struct in6_addr *src,
+  struct in6_addr *dst)
 {
  struct inet_frag_queue *q;
+ struct ip6_create_arg arg;

- q = inet_frag_intern(&fq_in->q, &nf_frags, hash);
- return container_of(q, struct nf_ct_frag6_queue, q);
-}
-
-
-static struct nf_ct_frag6_queue *
-nf_ct_frag6_create(unsigned int hash, __be32 id, struct in6_addr *src,       struct in6_addr *dst)
-{
- struct nf_ct_frag6_queue *fq;
+ arg.id = id;
+ arg.src = src;
+ arg.dst = dst;

- if ((fq = frag_alloc_queue()) == NULL) {
-  pr_debug("Can't alloc new queue\n");
+ q = inet_frag_create(&nf_frags, &arg, hash);
+ if (q == NULL)
   goto oom;
- }

- fq->id = id;
- ipv6_addr_copy(&fq->saddr, src);
- ipv6_addr_copy(&fq->daddr, dst);
-
- return nf_ct_frag6_intern(hash, fq);
+ return container_of(q, struct nf_ct_frag6_queue, q);

 oom:
+ pr_debug("Can't alloc new queue\n");
  return NULL;
 }

@@ -718,6 +702,7 @@ 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 = nf_frag_free;
  nf_frags.skb_free = nf_skb_free;
  nf_frags.qsize = sizeof(struct nf_ct_frag6_queue);
```

```
diff --git a/net/ipv6/reassembly.c b/net/ipv6/reassembly.c
index 21913c7..ce87340 100644
--- a/net/ipv6/reassembly.c
+++ b/net/ipv6/reassembly.c
@@ -164,17 +164,20 @@ static inline void frag_kfree_skb(struct sk_buff *skb, int *work)
  kfree_skb(skb);
 }

-static void ip6_frag_free(struct inet_frag_queue *fq)
+void ip6_frag_init(struct inet_frag_queue *q, void *a)
 {
- kfree(container_of(fq, struct frag_queue, q));
+ struct frag_queue *fq = container_of(q, struct frag_queue, q);
+ struct ip6_create_arg *arg = a;
+
+ fq->id = arg->id;
+ ipv6_addr_copy(&fq->saddr, arg->src);
+ ipv6_addr_copy(&fq->daddr, arg->dst);
 }
+EXPORT_SYMBOL(ip6_frag_init);

-static inline struct frag_queue *frag_alloc_queue(void)
+static void ip6_frag_free(struct inet_frag_queue *fq)
 {
- struct inet_frag_queue *q;
-
- q = inet_frag_alloc(&ip6_frags);
- return q ? container_of(q, struct frag_queue, q) : NULL;
+ kfree(container_of(fq, struct frag_queue, q));
 }

 /* Destruction primitives. */
@@ -244,31 +247,22 @@ out:

 /* Creation primitives. */

-
-static struct frag_queue *ip6_frag_intern(struct frag_queue *fq_in,
-  unsigned int hash)
-{
- struct inet_frag_queue *q;
-
- q = inet_frag_intern(&fq_in->q, &ip6_frags, hash);
- return container_of(q, struct frag_queue, q);
-}
-
-
 static struct frag_queue *
```

```
 ip6_frag_create(__be32 id, struct in6_addr *src, struct in6_addr *dst,
   struct inet6_dev *idev, unsigned int hash)
 {
- struct frag_queue *fq;
+ struct inet_frag_queue *q;
+ struct ip6_create_arg arg;

- if ((fq = frag_alloc_queue()) == NULL)
-  goto oom;
+ arg.id = id;
+ arg.src = src;
+ arg.dst = dst;

- fq->id = id;
- ipv6_addr_copy(&fq->saddr, src);
- ipv6_addr_copy(&fq->daddr, dst);
+ q = inet_frag_create(&ip6_frags, &arg, hash);
+ if (q == NULL)
+  goto oom;

- return ip6_frag_intern(fq, hash);
+ return container_of(q, struct frag_queue, q);

 oom:
  IP6_INC_STATS_BH(idev, IPSTATS_MIB_REASMFAILS);
@@ -675,6 +669,7 @@ void __init ipv6_frag_init(void)

  ip6_frags.ctl = &ip6_frags_ctl;
  ip6_frags.hashfn = ip6_hashfn;
+ ip6_frags.constructor = ip6_frag_init;
  ip6_frags.destructor = ip6_frag_free;
  ip6_frags.skb_free = NULL;
  ip6_frags.qsize = sizeof(struct frag_queue);
--
1.5.3.4
```

Subject: [PATCH 5/7] Consolidate xxx_find() in fragment management
Posted by Pavel Emelianov on Tue, 16 Oct 2007 14:03:37 GMT
View Forum Message <> Reply to Message

Here we need another callback ->match to check whether the
entry found in hash matches the key passed. The key used
is the same as the creation argument for inet_frag_create.

Yet again, this ->match is the same for netfilter and ipv6.
Running a frew steps forward - this callback will later
replace the ->equal one.

Since the inet_frag_find() uses the already consolidated
inet_frag_create() remove the xxx_frag_create from protocol
codes.

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

```
diff --git a/include/net/inet_frag.h b/include/net/inet_frag.h
index e33072b..6429926 100644
--- a/include/net/inet_frag.h
+++ b/include/net/inet_frag.h
@@ -45,6 +45,8 @@ struct inet_frags {
  void    (*skb_free)(struct sk_buff *);
  int    (*equal)(struct inet_frag_queue *q1,
        struct inet_frag_queue *q2);
+ int    (*match)(struct inet_frag_queue *q,
+       void *arg);
  void    (*frag_expire)(unsigned long data);
 };

@@ -55,8 +57,8 @@ 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_create(struct inet_frags *f,
-  void *create_arg, unsigned int hash);
+struct inet_frag_queue *inet_frag_find(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/ipv6.h b/include/net/ipv6.h
index 9dc99bf..005853a 100644
--- a/include/net/ipv6.h
+++ b/include/net/ipv6.h
@@ -387,6 +387,7 @@ struct ip6_create_arg {
 };

 void ip6_frag_init(struct inet_frag_queue *q, void *a);
+int ip6_frag_match(struct inet_frag_queue *q, void *a);

 static inline int ipv6_addr_any(const struct in6_addr *a)
 {
diff --git a/net/ipv4/inet_fragment.c b/net/ipv4/inet_fragment.c
index 0124885..08901b4 100644
--- a/net/ipv4/inet_fragment.c
```

```
+++ b/net/ipv4/inet_fragment.c
@@ -226,8 +226,8 @@ static struct inet_frag_queue *inet_frag_alloc(struct inet_frags *f, void
*arg)
  return q;
 }

-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 inet_frags *f,
+  void *arg, unsigned int hash)
 {
  struct inet_frag_queue *q;

@@ -237,4 +237,23 @@ struct inet_frag_queue *inet_frag_create(struct inet_frags *f, void *arg,

  return inet_frag_intern(q, f, hash);
 }
-EXPORT_SYMBOL(inet_frag_create);
+
+struct inet_frag_queue *inet_frag_find(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)) {
+   atomic_inc(&q->refcnt);
+   read_unlock(&f->lock);
+   return q;
+  }
+ }
+ read_unlock(&f->lock);
+
+ return inet_frag_create(f, key, hash);
+}
+EXPORT_SYMBOL(inet_frag_find);
diff --git a/net/ipv4/ip_fragment.c b/net/ipv4/ip_fragment.c
index db29c3c..46f8de6 100644
--- a/net/ipv4/ip_fragment.c
+++ b/net/ipv4/ip_fragment.c
@@ -142,6 +142,19 @@ static int ip4_frag_equal(struct inet_frag_queue *q1,
    qp1->user == qp2->user);
 }

+static int ip4_frag_match(struct inet_frag_queue *q, void *a)
+{
```

```
+ struct ipq *qp;
+ struct ip4_create_arg *arg = a;
+
+ qp = container_of(q, struct ipq, q);
+ return (qp->id == arg->iph->id &&
+   qp->saddr == arg->iph->saddr &&
+   qp->daddr == arg->iph->daddr &&
+   qp->protocol == arg->iph->protocol &&
+   qp->user == arg->user);
+}
+
 /* Memory Tracking Functions. */
 static __inline__ void frag_kfree_skb(struct sk_buff *skb, int *work)
 {
@@ -235,18 +248,20 @@ out:
  ipq_put(qp);
 }

-/* Creation primitives. */
-
-/* Add an entry to the 'ipq' queue for a newly received IP datagram. */
-static struct ipq *ip_frag_create(struct iphdr *iph, u32 user, unsigned int h)
+/* 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)
 {
  struct inet_frag_queue *q;
  struct ip4_create_arg arg;
+ unsigned int hash;

  arg.iph = iph;
  arg.user = user;
+ hash = ipqhashfn(iph->id, iph->saddr, iph->daddr, iph->protocol);

- q = inet_frag_create(&ip4_frags, &arg, h);
+ q = inet_frag_find(&ip4_frags, &arg, hash);
  if (q == NULL)
   goto out_nomem;

@@ -257,37 +272,6 @@ out_nomem:
  return NULL;
 }

-/* 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)
```

```
-{
- __be16 id = iph->id;
- __be32 saddr = iph->saddr;
- __be32 daddr = iph->daddr;
- __u8 protocol = iph->protocol;
- unsigned int hash;
- struct ipq *qp;
- struct hlist_node *n;
-
- read_lock(&ip4_frags.lock);
- hash = ipqhashfn(id, saddr, daddr, protocol);
- hlist_for_each_entry(qp, n, &ip4_frags.hash[hash], q.list) {
-  if (qp->id == id  &&
-      qp->saddr == saddr &&
-      qp->daddr == daddr &&
-      qp->protocol == protocol &&
-      qp->user == user) {
-   atomic_inc(&qp->q.refcnt);
-   read_unlock(&ip4_frags.lock);
-   return qp;
-  }
- }
- read_unlock(&ip4_frags.lock);
-
- return ip_frag_create(iph, user, hash);
-}
-
 /* Is the fragment too far ahead to be part of ipq? */
 static inline int ip_frag_too_far(struct ipq *qp)
 {
@@ -648,6 +632,7 @@ void __init ipfrag_init(void)
  ip4_frags.skb_free = NULL;
  ip4_frags.qsize = sizeof(struct ipq);
  ip4_frags.equal = ip4_frag_equal;
+ ip4_frags.match = ip4_frag_match;
  ip4_frags.frag_expire = ip_expire;
  inet_frags_init(&ip4_frags);
 }
diff --git a/net/ipv6/netfilter/nf_conntrack_reasm.c b/net/ipv6/netfilter/nf_conntrack_reasm.c
index 72451e2..1ab52ef 100644
--- a/net/ipv6/netfilter/nf_conntrack_reasm.c
+++ b/net/ipv6/netfilter/nf_conntrack_reasm.c
@@ -176,18 +176,19 @@ out:

 /* Creation primitives. */

-static struct nf_ct_frag6_queue *
-nf_ct_frag6_create(unsigned int hash, __be32 id, struct in6_addr *src,
```

```
-  struct in6_addr *dst)
+static __inline__ struct nf_ct_frag6_queue *
+fq_find(__be32 id, struct in6_addr *src, struct in6_addr *dst)
 {
  struct inet_frag_queue *q;
  struct ip6_create_arg arg;
+ unsigned int hash;

  arg.id = id;
  arg.src = src;
  arg.dst = dst;
+ hash = ip6qhashfn(id, src, dst);

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

@@ -198,28 +199,6 @@ oom:
  return NULL;
 }

-static __inline__ struct nf_ct_frag6_queue *
-fq_find(__be32 id, struct in6_addr *src, struct in6_addr *dst)
-{
- struct nf_ct_frag6_queue *fq;
- struct hlist_node *n;
- unsigned int hash = ip6qhashfn(id, src, dst);
-
- read_lock(&nf_frags.lock);
- hlist_for_each_entry(fq, n, &nf_frags.hash[hash], q.list) {
-  if (fq->id == id &&
-      ipv6_addr_equal(src, &fq->saddr) &&
-      ipv6_addr_equal(dst, &fq->daddr)) {
-   atomic_inc(&fq->q.refcnt);
-   read_unlock(&nf_frags.lock);
-   return fq;
-  }
- }
- read_unlock(&nf_frags.lock);
-
- return nf_ct_frag6_create(hash, id, src, dst);
-}
-

 static int nf_ct_frag6_queue(struct nf_ct_frag6_queue *fq, struct sk_buff *skb,
        struct frag_hdr *fhdr, int nhoff)
@@ -706,6 +685,7 @@ int nf_ct_frag6_init(void)
```

```
  nf_frags.destructor = nf_frag_free;
  nf_frags.skb_free = nf_skb_free;
  nf_frags.qsize = sizeof(struct nf_ct_frag6_queue);
+ nf_frags.match = ip6_frag_match;
  nf_frags.equal = ip6_frag_equal;
  nf_frags.frag_expire = nf_ct_frag6_expire;
  inet_frags_init(&nf_frags);
diff --git a/net/ipv6/reassembly.c b/net/ipv6/reassembly.c
index ce87340..11fffe7 100644
--- a/net/ipv6/reassembly.c
+++ b/net/ipv6/reassembly.c
@@ -155,6 +155,18 @@ int ip6_frag_equal(struct inet_frag_queue *q1, struct inet_frag_queue
*q2)
 }
 EXPORT_SYMBOL(ip6_frag_equal);

+int ip6_frag_match(struct inet_frag_queue *q, void *a)
+{
+ struct frag_queue *fq;
+ struct ip6_create_arg *arg = a;
+
+ fq = container_of(q, struct frag_queue, q);
+ return (fq->id == arg->id &&
+   ipv6_addr_equal(&fq->saddr, arg->src) &&
+   ipv6_addr_equal(&fq->daddr, arg->dst));
+}
+EXPORT_SYMBOL(ip6_frag_match);
+
 /* Memory Tracking Functions. */
 static inline void frag_kfree_skb(struct sk_buff *skb, int *work)
 {
@@ -245,20 +257,20 @@ out:
  fq_put(fq);
 }

-/* Creation primitives. */
-
-static struct frag_queue *
-ip6_frag_create(__be32 id, struct in6_addr *src, struct in6_addr *dst,
-  struct inet6_dev *idev, unsigned int hash)
+static __inline__ struct frag_queue *
+fq_find(__be32 id, struct in6_addr *src, struct in6_addr *dst,
+ struct inet6_dev *idev)
 {
  struct inet_frag_queue *q;
  struct ip6_create_arg arg;
+ unsigned int hash;
```

```
  arg.id = id;
  arg.src = src;
  arg.dst = dst;
+ hash = ip6qhashfn(id, src, dst);

- q = inet_frag_create(&ip6_frags, &arg, hash);
+ q = inet_frag_find(&ip6_frags, &arg, hash);
  if (q == NULL)
   goto oom;

@@ -269,31 +281,6 @@ oom:
  return NULL;
 }

-static __inline__ struct frag_queue *
-fq_find(__be32 id, struct in6_addr *src, struct in6_addr *dst,
- struct inet6_dev *idev)
-{
- struct frag_queue *fq;
- struct hlist_node *n;
- unsigned int hash;
-
- read_lock(&ip6_frags.lock);
- hash = ip6qhashfn(id, src, dst);
- hlist_for_each_entry(fq, n, &ip6_frags.hash[hash], q.list) {
-  if (fq->id == id &&
-      ipv6_addr_equal(src, &fq->saddr) &&
-      ipv6_addr_equal(dst, &fq->daddr)) {
-   atomic_inc(&fq->q.refcnt);
-   read_unlock(&ip6_frags.lock);
-   return fq;
-  }
- }
- read_unlock(&ip6_frags.lock);
-
- return ip6_frag_create(id, src, dst, idev, hash);
-}
-
-
 static int ip6_frag_queue(struct frag_queue *fq, struct sk_buff *skb,
       struct frag_hdr *fhdr, int nhoff)
 {
@@ -673,6 +660,7 @@ void __init ipv6_frag_init(void)
  ip6_frags.destructor = ip6_frag_free;
  ip6_frags.skb_free = NULL;
  ip6_frags.qsize = sizeof(struct frag_queue);
+ ip6_frags.match = ip6_frag_match;
  ip6_frags.equal = ip6_frag_equal;
```

```
  ip6_frags.frag_expire = ip6_frag_expire;
  inet_frags_init(&ip6_frags);
--
1.5.3.4
```

Subject: [PATCH 6/7] Remove no longer needed ->equal callback
Posted by Pavel Emelianov on Tue, 16 Oct 2007 14:05:31 GMT
View Forum Message <> Reply to Message

Since this callback is used to check for conflicts in
hashtable when inserting a newly created frag queue, we can
do the same by checking for matching the queue with the
argument, used to create one.

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

```
diff --git a/include/net/inet_frag.h b/include/net/inet_frag.h
index 6429926..954def4 100644
--- a/include/net/inet_frag.h
+++ b/include/net/inet_frag.h
@@ -43,8 +43,6 @@ struct inet_frags {
       void *arg);
  void   (*destructor)(struct inet_frag_queue *);
  void   (*skb_free)(struct sk_buff *);
- int   (*equal)(struct inet_frag_queue *q1,
-       struct inet_frag_queue *q2);
  int   (*match)(struct inet_frag_queue *q,
       void *arg);
  void   (*frag_expire)(unsigned long data);
diff --git a/include/net/ipv6.h b/include/net/ipv6.h
index 005853a..ae328b6 100644
--- a/include/net/ipv6.h
+++ b/include/net/ipv6.h
@@ -378,7 +378,6 @@ static inline int ipv6_prefix_equal(const struct in6_addr *a1,
 }

 struct inet_frag_queue;
-int ip6_frag_equal(struct inet_frag_queue *q1, struct inet_frag_queue *q2);

 struct ip6_create_arg {
  __be32 id;
diff --git a/net/ipv4/inet_fragment.c b/net/ipv4/inet_fragment.c
index 08901b4..470b056 100644
--- a/net/ipv4/inet_fragment.c
+++ b/net/ipv4/inet_fragment.c
```

```
@@ -174,7 +174,7 @@ 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)
+  struct inet_frags *f, unsigned int hash, void *arg)
 {
  struct inet_frag_queue *qp;
 #ifdef CONFIG_SMP
@@ -188,7 +188,7 @@ 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->equal(qp, qp_in)) {
+  if (f->match(qp, arg)) {
    atomic_inc(&qp->refcnt);
    write_unlock(&f->lock);
    qp_in->last_in |= COMPLETE;
@@ -235,7 +235,7 @@ static struct inet_frag_queue *inet_frag_create(struct inet_frags *f,
  if (q == NULL)
   return NULL;

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

 struct inet_frag_queue *inet_frag_find(struct inet_frags *f, void *key,
diff --git a/net/ipv4/ip_fragment.c b/net/ipv4/ip_fragment.c
index 46f8de6..62dcb79 100644
--- a/net/ipv4/ip_fragment.c
+++ b/net/ipv4/ip_fragment.c
@@ -128,20 +128,6 @@ static unsigned int ip4_hashfn(struct inet_frag_queue *q)
  return ipqhashfn(ipq->id, ipq->saddr, ipq->daddr, ipq->protocol);
 }

-static int ip4_frag_equal(struct inet_frag_queue *q1,
-  struct inet_frag_queue *q2)
-{
- struct ipq *qp1, *qp2;
-
- qp1 = container_of(q1, struct ipq, q);
- qp2 = container_of(q2, struct ipq, q);
- return (qp1->id == qp2->id &&
-   qp1->saddr == qp2->saddr &&
-   qp1->daddr == qp2->daddr &&
-   qp1->protocol == qp2->protocol &&
-   qp1->user == qp2->user);
```

```
-}
-
 static int ip4_frag_match(struct inet_frag_queue *q, void *a)
 {
  struct ipq *qp;
@@ -631,7 +617,6 @@ void __init ipfrag_init(void)
  ip4_frags.destructor = ip4_frag_free;
  ip4_frags.skb_free = NULL;
  ip4_frags.qsize = sizeof(struct ipq);
- ip4_frags.equal = ip4_frag_equal;
  ip4_frags.match = ip4_frag_match;
  ip4_frags.frag_expire = ip_expire;
  inet_frags_init(&ip4_frags);
diff --git a/net/ipv6/netfilter/nf_conntrack_reasm.c b/net/ipv6/netfilter/nf_conntrack_reasm.c
index 1ab52ef..411daf5 100644
--- a/net/ipv6/netfilter/nf_conntrack_reasm.c
+++ b/net/ipv6/netfilter/nf_conntrack_reasm.c
@@ -686,7 +686,6 @@ int nf_ct_frag6_init(void)
  nf_frags.skb_free = nf_skb_free;
  nf_frags.qsize = sizeof(struct nf_ct_frag6_queue);
  nf_frags.match = ip6_frag_match;
- nf_frags.equal = ip6_frag_equal;
  nf_frags.frag_expire = nf_ct_frag6_expire;
  inet_frags_init(&nf_frags);

diff --git a/net/ipv6/reassembly.c b/net/ipv6/reassembly.c
index 11fffe7..01766bc 100644
--- a/net/ipv6/reassembly.c
+++ b/net/ipv6/reassembly.c
@@ -143,18 +143,6 @@ static unsigned int ip6_hashfn(struct inet_frag_queue *q)
  return ip6qhashfn(fq->id, &fq->saddr, &fq->daddr);
 }

-int ip6_frag_equal(struct inet_frag_queue *q1, struct inet_frag_queue *q2)
-{
- struct frag_queue *fq1, *fq2;
-
- fq1 = container_of(q1, struct frag_queue, q);
- fq2 = container_of(q2, struct frag_queue, q);
- return (fq1->id == fq2->id &&
-   ipv6_addr_equal(&fq2->saddr, &fq1->saddr) &&
-   ipv6_addr_equal(&fq2->daddr, &fq1->daddr));
-}
-EXPORT_SYMBOL(ip6_frag_equal);
-
 int ip6_frag_match(struct inet_frag_queue *q, void *a)
 {
  struct frag_queue *fq;
```

```
@@ -661,7 +649,6 @@ void __init ipv6_frag_init(void)
  ip6_frags.skb_free = NULL;
  ip6_frags.qsize = sizeof(struct frag_queue);
  ip6_frags.match = ip6_frag_match;
- ip6_frags.equal = ip6_frag_equal;
  ip6_frags.frag_expire = ip6_frag_expire;
  inet_frags_init(&ip6_frags);
 }
--
1.5.3.4
```

Subject: [PATCH 7/7] Consolidate frag queues freeing
Posted by Pavel Emelianov on Tue, 16 Oct 2007 14:07:26 GMT
View Forum Message <> Reply to Message

Since we now allocate the queues in inet_fragment.c, we
can safely free it in the same place. The ->destructor
callback thus becomes optional for inet_frags.

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

```
diff --git a/net/ipv4/inet_fragment.c b/net/ipv4/inet_fragment.c
index 470b056..a75f8cb 100644
--- a/net/ipv4/inet_fragment.c
+++ b/net/ipv4/inet_fragment.c
@@ -136,7 +136,9 @@ void inet_frag_destroy(struct inet_frag_queue *q, struct inet_frags *f,
   *work -= f->qsize;
  atomic_sub(f->qsize, &f->mem);

- f->destructor(q);
+ if (f->destructor)
+  f->destructor(q);
+ kfree(q);

 }
 EXPORT_SYMBOL(inet_frag_destroy);
diff --git a/net/ipv4/ip_fragment.c b/net/ipv4/ip_fragment.c
index 62dcb79..34e790d 100644
--- a/net/ipv4/ip_fragment.c
+++ b/net/ipv4/ip_fragment.c
@@ -171,7 +171,6 @@ static __inline__ void ip4_frag_free(struct inet_frag_queue *q)
  qp = container_of(q, struct ipq, q);
  if (qp->peer)
   inet_putpeer(qp->peer);
- kfree(qp);
```

```
 }


diff --git a/net/ipv6/netfilter/nf_conntrack_reasm.c b/net/ipv6/netfilter/nf_conntrack_reasm.c
index 411daf5..a70a482 100644
--- a/net/ipv6/netfilter/nf_conntrack_reasm.c
+++ b/net/ipv6/netfilter/nf_conntrack_reasm.c
@@ -130,11 +130,6 @@ static inline void frag_kfree_skb(struct sk_buff *skb, unsigned int *work)
  kfree_skb(skb);
 }

-static void nf_frag_free(struct inet_frag_queue *q)
-{
- kfree(container_of(q, struct nf_ct_frag6_queue, q));
-}
-
 /* Destruction primitives. */

 static __inline__ void fq_put(struct nf_ct_frag6_queue *fq)
@@ -682,7 +677,7 @@ 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 = nf_frag_free;
+ nf_frags.destructor = NULL;
  nf_frags.skb_free = nf_skb_free;
  nf_frags.qsize = sizeof(struct nf_ct_frag6_queue);
  nf_frags.match = ip6_frag_match;
diff --git a/net/ipv6/reassembly.c b/net/ipv6/reassembly.c
index 01766bc..76c88a9 100644
--- a/net/ipv6/reassembly.c
+++ b/net/ipv6/reassembly.c
@@ -175,11 +175,6 @@ void ip6_frag_init(struct inet_frag_queue *q, void *a)
 }
 EXPORT_SYMBOL(ip6_frag_init);

-static void ip6_frag_free(struct inet_frag_queue *fq)
-{
- kfree(container_of(fq, struct frag_queue, q));
-}
-
 /* Destruction primitives. */

 static __inline__ void fq_put(struct frag_queue *fq)
@@ -645,7 +640,7 @@ void __init ipv6_frag_init(void)
  ip6_frags.ctl = &ip6_frags_ctl;
  ip6_frags.hashfn = ip6_hashfn;
  ip6_frags.constructor = ip6_frag_init;
```

```
- ip6_frags.destructor = ip6_frag_free;
+ ip6_frags.destructor = NULL;
  ip6_frags.skb_free = NULL;
  ip6_frags.qsize = sizeof(struct frag_queue);
  ip6_frags.match = ip6_frag_match;
--
1.5.3.4
```

Subject: Re: [PATCH 1/7] Omit double hash calculations in xxx_frag_intern
Posted by davem on Thu, 18 Oct 2007 02:44:10 GMT
View Forum Message <> Reply to Message

From: Pavel Emelyanov <xemul@openvz.org>
Date: Tue, 16 Oct 2007 17:48:46 +0400

> 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>

Applied, but if we ever implement dynamically sized frag
queue hash tables, this recomputation of the hash function
in the hash insert function would need to be re-added.

Subject: Re: [PATCH 2/7] Consolidate xxx_frag_intern
Posted by davem on Thu, 18 Oct 2007 02:44:42 GMT
View Forum Message <> Reply to Message

From: Pavel Emelyanov <xemul@openvz.org>
Date: Tue, 16 Oct 2007 17:53:03 +0400

> This routine checks for the existence of a given entry
> in the hash table and inserts the new one if needed.
>
> The ->equal callback is used to compare two frag_queue-s
> together, but this one is temporary and will be removed
> later. The netfilter code and the ipv6 one use the same
> routine to compare frags.
>
> The inet_frag_intern() always returns non-NULL pointer,
> so convert the inet_frag_queue into protocol specific
> one (with the container_of) without any checks.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

Applied.

Subject: Re: [PATCH 3/7] Consolidate xxx_frag_alloc()
Posted by davem on Thu, 18 Oct 2007 02:45:52 GMT
View Forum Message <> Reply to Message

From: Pavel Emelyanov <xemul@openvz.org>
Date: Tue, 16 Oct 2007 17:57:44 +0400

> Just perform the kzalloc() allocation and setup common
> fields in the inet_frag_queue(). Then return the result
> to the caller to initialize the rest.
>
> The inet_frag_alloc() may return NULL, so check the
> return value before doing the container_of(). This looks
> ugly, but the xxx_frag_alloc() will be removed soon.
>
> The xxx_expire() timer callbacks are patches,
> because the argument is now the inet_frag_queue, not
> the protocol specific queue.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

Applied, although I had to correct the following white-space
problems in your patch:

Adds trailing whitespace.
diff:33:
Adds trailing whitespace.
diff:72:
Adds trailing whitespace.
diff:126:
warning: 3 lines add whitespace errors.

Subject: Re: [PATCH 4/7] Consolidate xxx_frag_create()
Posted by davem on Thu, 18 Oct 2007 02:46:56 GMT
View Forum Message <> Reply to Message

From: Pavel Emelyanov <xemul@openvz.org>
Date: Tue, 16 Oct 2007 18:00:10 +0400

> This one uses the xxx_frag_intern() and xxx_frag_alloc()
> routines, which are already consolidated, so remove them
> from protocol code (as promised).

>
> The ->constructor callback is used to init the rest of
> the frag queue and it is the same for netfilter and ipv6.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

Applied.

Subject: Re: [PATCH 5/7] Consolidate xxx_find() in fragment management
Posted by davem on Thu, 18 Oct 2007 02:47:29 GMT
View Forum Message <> Reply to Message

From: Pavel Emelyanov <xemul@openvz.org>
Date: Tue, 16 Oct 2007 18:03:37 +0400

> Here we need another callback ->match to check whether the
> entry found in hash matches the key passed. The key used
> is the same as the creation argument for inet_frag_create.
>
> Yet again, this ->match is the same for netfilter and ipv6.
> Running a frew steps forward - this callback will later
> replace the ->equal one.
>
> Since the inet_frag_find() uses the already consolidated
> inet_frag_create() remove the xxx_frag_create from protocol
> codes.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

Applied.

Subject: Re: [PATCH 6/7] Remove no longer needed ->equal callback
Posted by davem on Thu, 18 Oct 2007 02:48:03 GMT
View Forum Message <> Reply to Message

From: Pavel Emelyanov <xemul@openvz.org>
Date: Tue, 16 Oct 2007 18:05:31 +0400

> Since this callback is used to check for conflicts in
> hashtable when inserting a newly created frag queue, we can
> do the same by checking for matching the queue with the
> argument, used to create one.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

Applied.

Subject: Re: [PATCH 7/7] Consolidate frag queues freeing
Posted by davem on Thu, 18 Oct 2007 02:48:36 GMT
View Forum Message <> Reply to Message

From: Pavel Emelyanov <xemul@openvz.org>
Date: Tue, 16 Oct 2007 18:07:26 +0400

> Since we now allocate the queues in inet_fragment.c, we
> can safely free it in the same place. The ->destructor
> callback thus becomes optional for inet_frags.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

Also applied, thanks a lot Pavel!