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
```