Subject: [PATCH 2/6 net-2.6.25] IPv6 RAW: Compact the API for the kernel
Posted by Pavel Emelianov on Fri, 16 Nov 2007 14:08:37 GMT
View Forum Message <> Reply to Message

Same as in the previous patch for ipv4, compact the
API and hide hash table and rwlock inside the raw.c
file.

Plus fix some "bad" places from checkpatch.pl point
of view (assignments inside if()).

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

```
diff --git a/include/net/rawv6.h b/include/net/rawv6.h
index a581989..8a22599 100644
--- a/include/net/rawv6.h
+++ b/include/net/rawv6.h
@@ -5,26 +5,13 @@

 #include <net/protocol.h>

-#define RAWV6_HTABLE_SIZE MAX_INET_PROTOS
-extern struct hlist_head raw_v6_htable[RAWV6_HTABLE_SIZE];
-extern rwlock_t raw_v6_lock;
-
-extern int ipv6_raw_deliver(struct sk_buff *skb, int nexthdr);
-
-extern struct sock *__raw_v6_lookup(struct sock *sk, unsigned short num,
-       struct in6_addr *loc_addr, struct in6_addr *rmt_addr,
-       int dif);
+void raw6_icmp_error(struct sk_buff *, int nexthdr,
+  int type, int code, int inner_offset, __be32);
+int raw6_local_deliver(struct sk_buff *, int);

 extern int   rawv6_rcv(struct sock *sk,
        struct sk_buff *skb);

-
-extern void   rawv6_err(struct sock *sk,
-       struct sk_buff *skb,
-       struct inet6_skb_parm *opt,
-       int type, int code,
-       int offset, __be32 info);
-
 #if defined(CONFIG_IPV6_MIP6) || defined(CONFIG_IPV6_MIP6_MODULE)
 int rawv6_mh_filter_register(int (*filter)(struct sock *sock,
```

```
        struct sk_buff *skb));
diff --git a/net/ipv6/icmp.c b/net/ipv6/icmp.c
index 9bb031f..4582bbc 100644
--- a/net/ipv6/icmp.c
+++ b/net/ipv6/icmp.c
@@ -557,9 +557,7 @@ out:

 static void icmpv6_notify(struct sk_buff *skb, int type, int code, __be32 info)
 {
- struct in6_addr *saddr, *daddr;
  struct inet6_protocol *ipprot;
- struct sock *sk;
  int inner_offset;
  int hash;
  u8 nexthdr;
@@ -581,9 +579,6 @@ static void icmpv6_notify(struct sk_buff *skb, int type, int code, __be32
info)
  if (!pskb_may_pull(skb, inner_offset+8))
   return;

- saddr = &ipv6_hdr(skb)->saddr;
- daddr = &ipv6_hdr(skb)->daddr;
-
  /* BUGGG_FUTURE: we should try to parse exthdrs in this packet.
     Without this we will not able f.e. to make source routed
     pmtu discovery.
@@ -599,15 +594,7 @@ static void icmpv6_notify(struct sk_buff *skb, int type, int code, __be32
info)
   ipprot->err_handler(skb, NULL, type, code, inner_offset, info);
  rcu_read_unlock();

- read_lock(&raw_v6_lock);
- if ((sk = sk_head(&raw_v6_htable[hash])) != NULL) {
-  while ((sk = __raw_v6_lookup(sk, nexthdr, saddr, daddr,
-         IP6CB(skb)->iif))) {
-   rawv6_err(sk, skb, NULL, type, code, inner_offset, info);
-   sk = sk_next(sk);
-  }
- }
- read_unlock(&raw_v6_lock);
+ raw6_icmp_error(skb, nexthdr, type, code, inner_offset, info);
 }

 /*
diff --git a/net/ipv6/ip6_input.c b/net/ipv6/ip6_input.c
index 79610b4..178aebc 100644
--- a/net/ipv6/ip6_input.c
+++ b/net/ipv6/ip6_input.c
```

```
@@ -153,9 +153,8 @@ out:
 static int ip6_input_finish(struct sk_buff *skb)
 {
  struct inet6_protocol *ipprot;
- struct sock *raw_sk;
  unsigned int nhoff;
- int nexthdr;
+ int nexthdr, raw;
  u8 hash;
  struct inet6_dev *idev;

@@ -171,9 +170,7 @@ resubmit:
  nhoff = IP6CB(skb)->nhoff;
  nexthdr = skb_network_header(skb)[nhoff];

- raw_sk = sk_head(&raw_v6_htable[nexthdr & (MAX_INET_PROTOS - 1)]);
- if (raw_sk && !ipv6_raw_deliver(skb, nexthdr))
-  raw_sk = NULL;
+ raw = raw6_local_deliver(skb, nexthdr);

  hash = nexthdr & (MAX_INET_PROTOS - 1);
  if ((ipprot = rcu_dereference(inet6_protos[hash])) != NULL) {
@@ -206,7 +203,7 @@ resubmit:
   else if (ret == 0)
    IP6_INC_STATS_BH(idev, IPSTATS_MIB_INDELIVERS);
  } else {
-  if (!raw_sk) {
+  if (!raw) {
    if (xfrm6_policy_check(NULL, XFRM_POLICY_IN, skb)) {
     IP6_INC_STATS_BH(idev, IPSTATS_MIB_INUNKNOWNPROTOS);
     icmpv6_send(skb, ICMPV6_PARAMPROB,
diff --git a/net/ipv6/raw.c b/net/ipv6/raw.c
index ad622cc..53f01b4 100644
--- a/net/ipv6/raw.c
+++ b/net/ipv6/raw.c
@@ -60,8 +60,10 @@
 #include <linux/proc_fs.h>
 #include <linux/seq_file.h>

-struct hlist_head raw_v6_htable[RAWV6_HTABLE_SIZE];
-DEFINE_RWLOCK(raw_v6_lock);
+#define RAWV6_HTABLE_SIZE MAX_INET_PROTOS
+
+static struct hlist_head raw_v6_htable[RAWV6_HTABLE_SIZE];
+static DEFINE_RWLOCK(raw_v6_lock);

 static void raw_v6_hash(struct sock *sk)
 {
```

```
@@ -83,10 +85,8 @@ static void raw_v6_unhash(struct sock *sk)
 }


-/* Grumble... icmp and ip_input want to get at this... */
-struct sock *__raw_v6_lookup(struct sock *sk, unsigned short num,
-       struct in6_addr *loc_addr, struct in6_addr *rmt_addr,
-       int dif)
+static struct sock *__raw_v6_lookup(struct sock *sk, unsigned short num,
+  struct in6_addr *loc_addr, struct in6_addr *rmt_addr, int dif)
 {
  struct hlist_node *node;
  int is_multicast = ipv6_addr_is_multicast(loc_addr);
@@ -167,7 +167,7 @@ EXPORT_SYMBOL(rawv6_mh_filter_unregister);
  *
  * Caller owns SKB so we must make clones.
  */
-int ipv6_raw_deliver(struct sk_buff *skb, int nexthdr)
+static int ipv6_raw_deliver(struct sk_buff *skb, int nexthdr)
 {
  struct in6_addr *saddr;
  struct in6_addr *daddr;
@@ -242,6 +242,17 @@ out:
  return delivered;
 }

+int raw6_local_deliver(struct sk_buff *skb, int nexthdr)
+{
+ struct sock *raw_sk;
+
+ raw_sk = sk_head(&raw_v6_htable[nexthdr & (MAX_INET_PROTOS - 1)]);
+ if (raw_sk && !ipv6_raw_deliver(skb, nexthdr))
+  raw_sk = NULL;
+
+ return raw_sk != NULL;
+}
+
 /* This cleans up af_inet6 a bit. -DaveM */
 static int rawv6_bind(struct sock *sk, struct sockaddr *uaddr, int addr_len)
 {
@@ -316,7 +327,7 @@ out:
  return err;
 }

-void rawv6_err(struct sock *sk, struct sk_buff *skb,
+static void rawv6_err(struct sock *sk, struct sk_buff *skb,
        struct inet6_skb_parm *opt,
        int type, int code, int offset, __be32 info)
```

```
 {
@@ -350,6 +361,31 @@ void rawv6_err(struct sock *sk, struct sk_buff *skb,
  }
 }

+void raw6_icmp_error(struct sk_buff *skb, int nexthdr,
+  int type, int code, int inner_offset, __be32 info)
+{
+ struct sock *sk;
+ int hash;
+ struct in6_addr *saddr, *daddr;
+
+ hash = nexthdr & (RAWV6_HTABLE_SIZE - 1);
+
+ read_lock(&raw_v6_lock);
+ sk = sk_head(&raw_v6_htable[hash]);
+ if (sk != NULL) {
+  saddr = &ipv6_hdr(skb)->saddr;
+  daddr = &ipv6_hdr(skb)->daddr;
+
+  while ((sk = __raw_v6_lookup(sk, nexthdr, saddr, daddr,
+       IP6CB(skb)->iif))) {
+   rawv6_err(sk, skb, NULL, type, code,
+     inner_offset, info);
+   sk = sk_next(sk);
+  }
+ }
+ read_unlock(&raw_v6_lock);
+}
+
 static inline int rawv6_rcv_skb(struct sock * sk, struct sk_buff * skb)
 {
  if ((raw6_sk(sk)->checksum || sk->sk_filter) &&
```