Subject: [PATCH 1/6 net-2.6.25] IPv4 RAW: Compact the API for the kernel
Posted by Pavel Emelianov on Fri, 16 Nov 2007 14:05:59 GMT
View Forum Message <> Reply to Message

The raw sockets functions are explicitly used from
inside the kernel in two places:

1. in ip_local_deliver_finish to intercept skb-s
2. in icmp_error

For this purposes many functions and even data structures,
that are naturally internal for raw protocol, are exported.

Compact the API to two functions and hide all the other
(including hash table and rwlock) inside the net/ipv4/raw.c

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

```
diff --git a/include/net/raw.h b/include/net/raw.h
index e4af597..7fc3c77 100644
--- a/include/net/raw.h
+++ b/include/net/raw.h
@@ -22,23 +22,10 @@

 extern struct proto raw_prot;

-extern void  raw_err(struct sock *, struct sk_buff *, u32 info);
-extern int  raw_rcv(struct sock *, struct sk_buff *);
-
-/* Note: v4 ICMP wants to get at this stuff, if you change the
- *       hashing mechanism, make sure you update icmp.c as well.
- */
-#define RAWV4_HTABLE_SIZE MAX_INET_PROTOS
-extern struct hlist_head raw_v4_htable[RAWV4_HTABLE_SIZE];
-
-extern rwlock_t raw_v4_lock;
+void raw_icmp_error(struct sk_buff *, int, u32);
+int raw_local_deliver(struct sk_buff *, int);

-
-extern struct sock *__raw_v4_lookup(struct sock *sk, unsigned short num,
-       __be32 raddr, __be32 laddr,
-       int dif);
-
-extern int raw_v4_input(struct sk_buff *skb, struct iphdr *iph, int hash);
+extern int  raw_rcv(struct sock *, struct sk_buff *);
```

```
 #ifdef CONFIG_PROC_FS
 extern int  raw_proc_init(void);
diff --git a/net/ipv4/icmp.c b/net/ipv4/icmp.c
index 233de06..c0898c5 100644
--- a/net/ipv4/icmp.c
+++ b/net/ipv4/icmp.c
@@ -604,7 +604,6 @@ static void icmp_unreach(struct sk_buff *skb)
  struct icmphdr *icmph;
  int hash, protocol;
  struct net_protocol *ipprot;
- struct sock *raw_sk;
  u32 info = 0;

  /*
@@ -698,21 +697,9 @@ static void icmp_unreach(struct sk_buff *skb)
  /*
   * Deliver ICMP message to raw sockets. Pretty useless feature?
   */
+ raw_icmp_error(skb, protocol, info);

- /* Note: See raw.c and net/raw.h, RAWV4_HTABLE_SIZE==MAX_INET_PROTOS */
  hash = protocol & (MAX_INET_PROTOS - 1);
- read_lock(&raw_v4_lock);
- if ((raw_sk = sk_head(&raw_v4_htable[hash])) != NULL) {
-  while ((raw_sk = __raw_v4_lookup(raw_sk, protocol, iph->daddr,
-       iph->saddr,
-       skb->dev->ifindex)) != NULL) {
-   raw_err(raw_sk, skb, info);
-   raw_sk = sk_next(raw_sk);
-   iph = (struct iphdr *)skb->data;
-  }
- }
- read_unlock(&raw_v4_lock);
-
  rcu_read_lock();
  ipprot = rcu_dereference(inet_protos[hash]);
  if (ipprot && ipprot->err_handler)
diff --git a/net/ipv4/ip_input.c b/net/ipv4/ip_input.c
index 5b8a760..11281d6 100644
--- a/net/ipv4/ip_input.c
+++ b/net/ipv4/ip_input.c
@@ -204,22 +204,14 @@ static int ip_local_deliver_finish(struct sk_buff *skb)

  rcu_read_lock();
  {
-  /* Note: See raw.c and net/raw.h, RAWV4_HTABLE_SIZE==MAX_INET_PROTOS */
   int protocol = ip_hdr(skb)->protocol;
```

```
-  int hash;
-  struct sock *raw_sk;
+   int hash, raw;
    struct net_protocol *ipprot;

   resubmit:
-  hash = protocol & (MAX_INET_PROTOS - 1);
-  raw_sk = sk_head(&raw_v4_htable[hash]);
-
-  /* If there maybe a raw socket we must check - if not we
-   * don't care less
-   */
-  if (raw_sk && !raw_v4_input(skb, ip_hdr(skb), hash))
-   raw_sk = NULL;
+   raw = raw_local_deliver(skb, protocol);

+   hash = protocol & (MAX_INET_PROTOS - 1);
    if ((ipprot = rcu_dereference(inet_protos[hash])) != NULL) {
     int ret;

@@ -237,7 +229,7 @@ static int ip_local_deliver_finish(struct sk_buff *skb)
     }
     IP_INC_STATS_BH(IPSTATS_MIB_INDELIVERS);
    } else {
-    if (!raw_sk) {
+    if (!raw) {
      if (xfrm4_policy_check(NULL, XFRM_POLICY_IN, skb)) {
       IP_INC_STATS_BH(IPSTATS_MIB_INUNKNOWNPROTOS);
       icmp_send(skb, ICMP_DEST_UNREACH,
diff --git a/net/ipv4/raw.c b/net/ipv4/raw.c
index 3c12aa1..9fda2f9 100644
--- a/net/ipv4/raw.c
+++ b/net/ipv4/raw.c
@@ -80,8 +80,10 @@
 #include <linux/netfilter.h>
 #include <linux/netfilter_ipv4.h>

-struct hlist_head raw_v4_htable[RAWV4_HTABLE_SIZE];
-DEFINE_RWLOCK(raw_v4_lock);
+#define RAWV4_HTABLE_SIZE MAX_INET_PROTOS
+
+static struct hlist_head raw_v4_htable[RAWV4_HTABLE_SIZE];
+static DEFINE_RWLOCK(raw_v4_lock);

 static void raw_v4_hash(struct sock *sk)
 {
@@ -102,7 +104,7 @@ static void raw_v4_unhash(struct sock *sk)
   write_unlock_bh(&raw_v4_lock);
```

```
 }

-struct sock *__raw_v4_lookup(struct sock *sk, unsigned short num,
+static struct sock *__raw_v4_lookup(struct sock *sk, unsigned short num,
         __be32 raddr, __be32 laddr,
         int dif)
 {
@@ -150,7 +152,7 @@ static __inline__ int icmp_filter(struct sock *sk, struct sk_buff *skb)
  * RFC 1122: SHOULD pass TOS value up to the transport layer.
  * -> It does. And not only TOS, but all IP header.
  */
-int raw_v4_input(struct sk_buff *skb, struct iphdr *iph, int hash)
+static int raw_v4_input(struct sk_buff *skb, struct iphdr *iph, int hash)
 {
  struct sock *sk;
  struct hlist_head *head;
@@ -182,7 +184,25 @@ out:
  return delivered;
 }

-void raw_err (struct sock *sk, struct sk_buff *skb, u32 info)
+int raw_local_deliver(struct sk_buff *skb, int protocol)
+{
+ int hash;
+ struct sock *raw_sk;
+
+ hash = protocol & (RAWV4_HTABLE_SIZE - 1);
+ raw_sk = sk_head(&raw_v4_htable[hash]);
+
+ /* If there maybe a raw socket we must check - if not we
+  * don't care less
+  */
+ if (raw_sk && !raw_v4_input(skb, ip_hdr(skb), hash))
+  raw_sk = NULL;
+
+ return raw_sk != NULL;
+
+}
+
+static void raw_err(struct sock *sk, struct sk_buff *skb, u32 info)
 {
  struct inet_sock *inet = inet_sk(sk);
  const int type = icmp_hdr(skb)->type;
@@ -236,6 +256,29 @@ void raw_err (struct sock *sk, struct sk_buff *skb, u32 info)
  }
 }

+void raw_icmp_error(struct sk_buff *skb, int protocol, u32 info)
```

```
+{
+ int hash;
+ struct sock *raw_sk;
+ struct iphdr *iph;
+
+ hash = protocol & (RAWV4_HTABLE_SIZE - 1);
+
+ read_lock(&raw_v4_lock);
+ raw_sk = sk_head(&raw_v4_htable[hash]);
+ if (raw_sk != NULL) {
+  iph = (struct iphdr *)skb->data;
+  while ((raw_sk = __raw_v4_lookup(raw_sk, protocol, iph->daddr,
+      iph->saddr,
+      skb->dev->ifindex)) != NULL) {
+   raw_err(raw_sk, skb, info);
+   raw_sk = sk_next(raw_sk);
+   iph = (struct iphdr *)skb->data;
+  }
+ }
+ read_unlock(&raw_v4_lock);
+}
+
 static int raw_rcv_skb(struct sock * sk, struct sk_buff * skb)
 {
  /* Charge it to the socket. */
```