
Subject: [PATCH 6/6][NETNS]: Udp sockets per-net lookup.
Posted by [Pavel Emelianov](#) on Thu, 31 Jan 2008 12:41:58 GMT
[View Forum Message](#) <> [Reply to Message](#)

Add the net parameter to udp_get_port family of calls and
udp_lookup one and use it to filter sockets.

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

```
net/ipv4/udp.c | 25 ++++++-----
net/ipv6/udp.c | 10 ++++++----
2 files changed, 20 insertions(+), 15 deletions(-)
```

```
diff --git a/net/ipv4/udp.c b/net/ipv4/udp.c
index 2fb8d73..7ea1b67 100644
```

```
--- a/net/ipv4/udp.c
```

```
+++ b/net/ipv4/udp.c
```

```
@@ -130,14 +130,14 @@ EXPORT_SYMBOL(sysctl_udp_wmem_min);
atomic_t udp_memory_allocated;
EXPORT_SYMBOL(udp_memory_allocated);
```

```
-static inline int __udp_lib_lport_inuse(__u16 num,
+static inline int __udp_lib_lport_inuse(struct net *net, __u16 num,
      const struct hlist_head udptable[])
{
    struct sock *sk;
    struct hlist_node *node;
```

```
    sk_for_each(sk, node, &udptable[num & (UDP_HTABLE_SIZE - 1)])
- if (sk->sk_hash == num)
+ if (sk->sk_net == net && sk->sk_hash == num)
    return 1;
    return 0;
}
```

```
@@ -159,6 +159,7 @@ int __udp_lib_get_port(struct sock *sk, unsigned short snum,
    struct hlist_head *head;
    struct sock *sk2;
    int error = 1;
+ struct net *net = sk->sk_net;
```

```
    write_lock_bh(&udp_hash_lock);
```

```
@@ -198,7 +199,7 @@ int __udp_lib_get_port(struct sock *sk, unsigned short snum,
    /* 2nd pass: find hole in shortest hash chain */
    rover = best;
    for (i = 0; i < (1 << 16) / UDP_HTABLE_SIZE; i++) {
- if (! __udp_lib_lport_inuse(rover, udptable))
```

```

+ if (!__udp_lib_lport_inuse(net, rover, udptable))
    goto gotit;
    rover += UDP_HTABLE_SIZE;
    if (rover > high)
@@ -218,6 +219,7 @@ gotit:
    sk_for_each(sk2, node, head)
        if (sk2->sk_hash == snum                &&
            sk2 != sk                            &&
+         sk2->sk_net == net                    &&
            (!sk2->sk_reuse || !sk->sk_reuse)      &&
            (!sk2->sk_bound_dev_if || !sk->sk_bound_dev_if
             || sk2->sk_bound_dev_if == sk->sk_bound_dev_if) &&
@@ -261,9 +263,9 @@ static inline int udp_v4_get_port(struct sock *sk, unsigned short snum)
/* UDP is nearly always wildcards out the wazoo, it makes no sense to try
 * harder than this. -DaveM
 */
-static struct sock *__udp4_lib_lookup(__be32 saddr, __be16 sport,
-    __be32 daddr, __be16 dport,
-    int dif, struct hlist_head udptable[])
+static struct sock *__udp4_lib_lookup(struct net *net, __be32 saddr,
+    __be16 sport, __be32 daddr, __be16 dport,
+    int dif, struct hlist_head udptable[])
{
    struct sock *sk, *result = NULL;
    struct hlist_node *node;
@@ -274,7 +276,8 @@ static struct sock *__udp4_lib_lookup(__be32 saddr, __be16 sport,
    sk_for_each(sk, node, &udptable[hnum & (UDP_HTABLE_SIZE - 1)]) {
        struct inet_sock *inet = inet_sk(sk);

-    if (sk->sk_hash == hnum && !ipv6_only_sock(sk)) {
+    if (sk->sk_net == net && sk->sk_hash == hnum &&
+        !ipv6_only_sock(sk)) {
        int score = (sk->sk_family == PF_INET ? 1 : 0);
        if (inet->rcv_saddr) {
            if (inet->rcv_saddr != daddr)
@@ -361,8 +364,8 @@ void __udp4_lib_err(struct sk_buff *skb, u32 info, struct hlist_head
udptable[])
    int harderr;
    int err;

-    sk = __udp4_lib_lookup(iph->daddr, uh->dest, iph->saddr, uh->source,
-        skb->dev->ifindex, udptable    );
+    sk = __udp4_lib_lookup(skb->dev->nd_net, iph->daddr, uh->dest,
+        iph->saddr, uh->source, skb->dev->ifindex, udptable);
    if (sk == NULL) {
        ICMP_INC_STATS_BH(ICMP_MIB_INERRORS);
        return; /* No socket for error */
@@ -1185,8 +1188,8 @@ int __udp4_lib_rcv(struct sk_buff *skb, struct hlist_head udptable[],

```

```

if (rt->rt_flags & (RTCF_BROADCAST|RTCF_MULTICAST))
    return __udp4_lib_mcast_deliver(skb, uh, saddr, daddr, udptable);

- sk = __udp4_lib_lookup(saddr, uh->source, daddr, uh->dest,
-     inet_iif(skb), udptable);
+ sk = __udp4_lib_lookup(skb->dev->nd_net, saddr, uh->source, daddr,
+     uh->dest, inet_iif(skb), udptable);

    if (sk != NULL) {
        int ret = 0;
diff --git a/net/ipv6/udp.c b/net/ipv6/udp.c
index bd4b9df..53739de 100644
--- a/net/ipv6/udp.c
+++ b/net/ipv6/udp.c
@@ -56,7 +56,8 @@ static inline int udp_v6_get_port(struct sock *sk, unsigned short snum)
    return udp_get_port(sk, snum, ipv6_rcv_saddr_equal);
}

-static struct sock *__udp6_lib_lookup(struct in6_addr *saddr, __be16 sport,
+static struct sock *__udp6_lib_lookup(struct net *net,
+    struct in6_addr *saddr, __be16 sport,
+    struct in6_addr *daddr, __be16 dport,
+    int dif, struct hlist_head udptable[])
{
@@ -69,7 +70,8 @@ static struct sock *__udp6_lib_lookup(struct in6_addr *saddr, __be16 sport,
    sk_for_each(sk, node, &udptable[hnum & (UDP_HTABLE_SIZE - 1)]) {
        struct inet_sock *inet = inet_sk(sk);

-    if (sk->sk_hash == hnum && sk->sk_family == PF_INET6) {
+    if (sk->sk_net == net && sk->sk_hash == hnum &&
+        sk->sk_family == PF_INET6) {
        struct ipv6_pinfo *np = inet6_sk(sk);
        int score = 0;
        if (inet->dport) {
@@ -233,7 +235,7 @@ void __udp6_lib_err(struct sk_buff *skb, struct inet6_skb_parm *opt,
    struct sock *sk;
    int err;

-    sk = __udp6_lib_lookup(daddr, uh->dest,
+    sk = __udp6_lib_lookup(skb->dev->nd_net, daddr, uh->dest,
        saddr, uh->source, inet6_iif(skb), udptable);
    if (sk == NULL)
        return;
@@ -478,7 +480,7 @@ int __udp6_lib_rcv(struct sk_buff *skb, struct hlist_head udptable[],
    * check socket cache ... must talk to Alan about his plans
    * for sock caches... i'll skip this for now.
    */
-    sk = __udp6_lib_lookup(saddr, uh->source,

```

```
+ sk = __udp6_lib_lookup(skb->dev->nd_net, saddr, uh->source,  
    daddr, uh->dest, inet6_iif(skb), udptable);
```

```
    if (sk == NULL) {
```

```
--
```

```
1.5.3.4
```
