

Hi all,

I am trying to find a solution to handle the broadcast traffic on the I3 namespace.

The broadcast issue comes from the I2 isolation:

in udp.c

```
static inline struct sock *udp_v4_mcast_next(struct sock *sk,
__be16 loc_port,
__be32 loc_addr,
__be16 rmt_port,
__be32 rmt_addr,
int dif)
{
    struct hlist_node *node;
    struct sock *s = sk;
    struct net_namespace *ns = current_net_ns;
    unsigned short hnum = ntohs(loc_port);

    sk_for_each_from(s, node) {
        struct inet_sock *inet = inet_sk(s);

        if (inet->num != hnum ||
            (inet->daddr && inet->daddr != rmt_addr) ||
            (inet->dport != rmt_port && inet->dport) ||
            (inet->rcv_saddr && inet->rcv_saddr != loc_addr) ||
            ipv6_only_sock(s) ||
            !net_ns_match(sk->sk_net_ns, ns) ||
            (s->sk_bound_dev_if && s->sk_bound_dev_if != dif))
            continue;
        if (!ip_mc_sf_allow(s, loc_addr, rmt_addr, dif))
            continue;
        goto found;
    }
    s = NULL;
found:
    return s;
}
```

This is absolutely correct for I2 namespaces because they share the socket hash table. But that is not correct for I3 namespaces because we want to deliver the packet to each I3 namespaces which have binded to

the broadcast address, so we should avoid checking `net_ns_match` if we are in a layer 3 namespace. Doing that we will break the I2 isolation because another I2 namespace could have binded to the same broadcast address.

The solution I see here is:

```
if namespace is I3 then;  
    net_ns match any net_ns registered as listening on this address  
else  
    net_ns_match  
fi
```

The registered network namespace is a list shared between brothers I3 namespaces. This will add more overhead for sure. Does anyone have comments on that or perhaps a better solution ?

---

Containers mailing list  
Containers@lists.osdl.org  
<https://lists.osdl.org/mailman/listinfo/containers>

---