
Subject: Re: [PATCH 2/6][INET]: Consolidate inet(6)_hash_connect.
Posted by [Araldo Carvalho de M](#) on Thu, 31 Jan 2008 15:42:45 GMT
[View Forum Message](#) <> [Reply to Message](#)

Em Thu, Jan 31, 2008 at 11:39:55AM -0200, Araldo Carvalho de Melo escreveu:

> Em Thu, Jan 31, 2008 at 04:18:51PM +0300, Pavel Emelyanov escreveu:

> > Araldo Carvalho de Melo wrote:

> > > Em Thu, Jan 31, 2008 at 03:32:09PM +0300, Pavel Emelyanov escreveu:

> > > > These two functions are the same except for what they call

> > > > to "check_established" and "hash" for a socket.

> > > >

> > > > This saves half-a-kilo for ipv4 and ipv6.

> > >

> > > Good stuff!

> > >

> > > Yesterday I was perusing tcp_hash and I think we could have the hashinfo

> > > pointer stored perhaps in sk->sk_prot.

> > >

> > > That way we would be able to kill tcp_hash(), inet_put_port() could

> > > receive just sk, etc.

> >

> > But each proto will still have its own hashfn, so proto's

> > callbacks will be called to hash/unhash sockets, so this will

> > give us just one extra dereference. No?

> >

> > > What do you think?

> >

> > Hmmm... Even raw_hash, etc may become simpler. On the other hand

> > maybe this is a good idea, but I'm not very common with this code

> > yet to foresee such things in advance... I think that we should

> > try to prepare a patch and look, but if you have smth ready, then

> > it's better to review your stuff first.

>

> gimme some minutes

A bit more than minutes tho, but here it is, I'm testing it now.

Take a look and if testing is ok I'll submit it with a proper description.

- Araldo

```
diff --git a/include/net/inet6_hashtables.h b/include/net/inet6_hashtables.h
```

```
index fdff630..62a5b69 100644
```

```
--- a/include/net/inet6_hashtables.h
```

```
+++ b/include/net/inet6_hashtables.h
```

```
@@ -49,7 +49,7 @@ static inline int inet6_sk_ehashfn(const struct sock *sk)
     return inet6_ehashfn(laddr, lport, faddr, fport);
```

```

}

-extern void __inet6_hash(struct inet_hashinfo *hashinfo, struct sock *sk);
+extern void __inet6_hash(struct sock *sk);

/*
 * Sockets in TCP_CLOSE state are _always_ taken out of the hash, so
diff --git a/include/net/inet_connection_sock.h b/include/net/inet_connection_sock.h
index 133cf30..f00f057 100644
--- a/include/net/inet_connection_sock.h
+++ b/include/net/inet_connection_sock.h
@@ -29,7 +29,6 @@
 #undef INET_CSK_CLEAR_TIMERS

 struct inet_bind_bucket;
-struct inet_hashinfo;
 struct tcp_congestion_ops;

/*
@@ -59,6 +58,8 @@ struct inet_connection_sock_af_ops {
    int level, int optname,
    char __user *optval, int __user *optlen);
    void (*addr2sockaddr)(struct sock *sk, struct sockaddr *);
+ int (*bind_conflict)(const struct sock *sk,
+ const struct inet_bind_bucket *tb);
};

/** inet_connection_sock - INET connection oriented sock
@@ -244,10 +245,7 @@ extern struct request_sock *inet_csk_search_req(const struct sock *sk,
    const __be32 laddr);
extern int inet_csk_bind_conflict(const struct sock *sk,
    const struct inet_bind_bucket *tb);
-extern int inet_csk_get_port(struct inet_hashinfo *hashinfo,
- struct sock *sk, unsigned short snum,
- int (*bind_conflict)(const struct sock *sk,
- const struct inet_bind_bucket *tb));
+extern int inet_csk_get_port(struct sock *sk, unsigned short snum);

extern struct dst_entry* inet_csk_route_req(struct sock *sk,
    const struct request_sock *req);
diff --git a/include/net/inet_hashtables.h b/include/net/inet_hashtables.h
index c23c4ed..48ac620 100644
--- a/include/net/inet_hashtables.h
+++ b/include/net/inet_hashtables.h
@@ -221,9 +221,9 @@ static inline int inet_sk_listen_hashfn(const struct sock *sk)
}

/* Caller must disable local BH processing. */

```

```

-static inline void __inet_inherit_port(struct inet_hashinfo *table,
-      struct sock *sk, struct sock *child)
+static inline void __inet_inherit_port(struct sock *sk, struct sock *child)
{
+ struct inet_hashinfo *table = sk->sk_prot->hashinfo;
  const int bhash = inet_bhashfn(inet_sk(child)->num, table->bhash_size);
  struct inet_bind_hashbucket *head = &table->bhash[bhash];
  struct inet_bind_bucket *tb;
@@ -235,15 +235,14 @@ static inline void __inet_inherit_port(struct inet_hashinfo *table,
  spin_unlock(&head->lock);
}

-static inline void inet_inherit_port(struct inet_hashinfo *table,
-      struct sock *sk, struct sock *child)
+static inline void inet_inherit_port(struct sock *sk, struct sock *child)
{
  local_bh_disable();
- __inet_inherit_port(table, sk, child);
+ __inet_inherit_port(sk, child);
  local_bh_enable();
}

-extern void inet_put_port(struct inet_hashinfo *table, struct sock *sk);
+extern void inet_put_port(struct sock *sk);

extern void inet_listen_wlock(struct inet_hashinfo *hashinfo);

@@ -266,41 +265,9 @@ static inline void inet_listen_unlock(struct inet_hashinfo *hashinfo)
  wake_up(&hashinfo->lhash_wait);
}

-extern void __inet_hash(struct inet_hashinfo *hashinfo, struct sock *sk);
-extern void __inet_hash_nolisten(struct inet_hashinfo *hinfo, struct sock *sk);
-
-static inline void inet_hash(struct inet_hashinfo *hashinfo, struct sock *sk)
-{
- if (sk->sk_state != TCP_CLOSE) {
-   local_bh_disable();
-   __inet_hash(hashinfo, sk);
-   local_bh_enable();
- }
-}
-
-static inline void inet_unhash(struct inet_hashinfo *hashinfo, struct sock *sk)
-{
- rwlock_t *lock;
-
- if (sk_unhashed(sk))

```

```

- goto out;
-
- if (sk->sk_state == TCP_LISTEN) {
- local_bh_disable();
- inet_listen_wlock(hashinfo);
- lock = &hashinfo->lhash_lock;
- } else {
- lock = inet_eshash_lockp(hashinfo, sk->sk_hash);
- write_lock_bh(lock);
- }
-
- if (__sk_del_node_init(sk))
- sock_prot_inuse_add(sk->sk_prot, -1);
- write_unlock_bh(lock);
-out:
- if (sk->sk_state == TCP_LISTEN)
- wake_up(&hashinfo->lhash_wait);
-}
+extern void __inet_hash_nolisten(struct sock *sk);
+extern void inet_hash(struct sock *sk);
+extern void inet_unhash(struct sock *sk);

extern struct sock *__inet_lookup_listener(struct net *net,
    struct inet_hashinfo *hashinfo,
@@ -425,7 +392,7 @@ extern int __inet_hash_connect(struct inet_timewait_death_row
*death_row,
    struct sock *sk,
    int (*check_established)(struct inet_timewait_death_row *,
    struct sock *, __u16, struct inet_timewait_sock **),
- void (*hash)(struct inet_hashinfo *, struct sock *));
+ void (*hash)(struct sock *sk));
extern int inet_hash_connect(struct inet_timewait_death_row *death_row,
    struct sock *sk);
#endif /* _INET_HASHTABLES_H */
diff --git a/include/net/sock.h b/include/net/sock.h
index e3fb4c0..8a7889b 100644
--- a/include/net/sock.h
+++ b/include/net/sock.h
@@ -496,6 +496,7 @@ extern int sk_wait_data(struct sock *sk, long *timeo);

struct request_sock_ops;
struct timewait_sock_ops;
+struct inet_hashinfo;

/* Networking protocol blocks we attach to sockets.
 * socket layer -> transport layer interface
@@ -578,6 +579,8 @@ struct proto {
    struct request_sock_ops *rsk_prot;

```

```

struct timewait_sock_ops *twsk_prot;

+ struct inet_hashinfo *hashinfo;
+
+ struct module *owner;

char name[32];
diff --git a/net/dccp/dccp.h b/net/dccp/dccp.h
index ebe59d9..287a62b 100644
--- a/net/dccp/dccp.h
+++ b/net/dccp/dccp.h
@@ -271,8 +271,6 @@ extern struct sk_buff *dccp_make_response(struct sock *sk,

extern int dccp_connect(struct sock *sk);
extern int dccp_disconnect(struct sock *sk, int flags);
-extern void dccp_hash(struct sock *sk);
-extern void dccp_unhash(struct sock *sk);
extern int dccp_getsockopt(struct sock *sk, int level, int optname,
                           char __user *optval, int __user *optlen);
extern int dccp_setsockopt(struct sock *sk, int level, int optname,
diff --git a/net/dccp/ipv4.c b/net/dccp/ipv4.c
index c982ad8..474075a 100644
--- a/net/dccp/ipv4.c
+++ b/net/dccp/ipv4.c
@@ -38,12 +38,6 @@
 */
static struct socket *dccp_v4_ctl_socket;

-static int dccp_v4_get_port(struct sock *sk, const unsigned short snum)
-{
- return inet_csk_get_port(&dccp_hashinfo, sk, snum,
- inet_csk_bind_conflict);
-}
-
int dccp_v4_connect(struct sock *sk, struct sockaddr *uaddr, int addr_len)
{
struct inet_sock *inet = inet_sk(sk);
@@ -408,8 +402,8 @@ struct sock *dccp_v4_request_recv_sock(struct sock *sk, struct sk_buff
*skb,

dccp_sync_mss(newsk, dst_mtu(dst));

- __inet_hash_nolisten(&dccp_hashinfo, newsk);
- __inet_inherit_port(&dccp_hashinfo, sk, newsk);
+ __inet_hash_nolisten(newsk);
+ __inet_inherit_port(sk, newsk);

return newsk;

```

```

@@ -898,6 +892,7 @@ static struct inet_connection_sock_af_ops dccp_ipv4_af_ops = {
    .getsockopt    = ip_getsockopt,
    .addr2sockaddr = inet_csk_addr2sockaddr,
    .sockaddr_len  = sizeof(struct sockaddr_in),
+ .bind_conflict  = inet_csk_bind_conflict,
#ifdef CONFIG_COMPAT
    .compat_setsockopt = compat_ip_setsockopt,
    .compat_getsockopt = compat_ip_getsockopt,
@@ -937,10 +932,10 @@ static struct proto dccp_v4_prot = {
    .sendmsg      = dccp_sendmsg,
    .recvmsg      = dccp_recvmsg,
    .backlog_rcv  = dccp_v4_do_rcv,
- .hash          = dccp_hash,
- .unhash        = dccp_unhash,
+ .hash          = inet_hash,
+ .unhash        = inet_unhash,
    .accept       = inet_csk_accept,
- .get_port      = dccp_v4_get_port,
+ .get_port      = inet_csk_get_port,
    .shutdown     = dccp_shutdown,
    .destroy       = dccp_destroy_sock,
    .orphan_count = &dccp_orphan_count,
@@ -948,6 +943,7 @@ static struct proto dccp_v4_prot = {
    .obj_size      = sizeof(struct dccp_sock),
    .rsk_prot      = &dccp_request_sock_ops,
    .twsk_prot     = &dccp_timewait_sock_ops,
+ .hashinfo       = &dccp_hashinfo,
#ifdef CONFIG_COMPAT
    .compat_setsockopt = compat_dccp_setsockopt,
    .compat_getsockopt = compat_dccp_getsockopt,
diff --git a/net/dccp/ipv6.c b/net/dccp/ipv6.c
index ed0a005..490333d 100644
--- a/net/dccp/ipv6.c
+++ b/net/dccp/ipv6.c
@@ -39,21 +39,15 @@ static struct socket *dccp_v6_ctl_socket;
static struct inet_connection_sock_af_ops dccp_ipv6_mapped;
static struct inet_connection_sock_af_ops dccp_ipv6_af_ops;

-static int dccp_v6_get_port(struct sock *sk, unsigned short snum)
-{
- return inet_csk_get_port(&dccp_hashinfo, sk, snum,
-    inet6_csk_bind_conflict);
-}
-
static void dccp_v6_hash(struct sock *sk)
{
    if (sk->sk_state != DCCP_CLOSED) {

```

```

    if (inet_csk(sk)->icsk_af_ops == &dccp_ipv6_mapped) {
-   dccp_hash(sk);
+   inet_hash(sk);
        return;
    }
    local_bh_disable();
-   __inet6_hash(&dccp_hashinfo, sk);
+   __inet6_hash(sk);
    local_bh_enable();
}
}
@@ -630,8 +624,8 @@ static struct sock *dccp_v6_request_rcv_sock(struct sock *sk,

    newinet->daddr = newinet->saddr = newinet->rcv_saddr = LOOPBACK4_IPV6;

-   __inet6_hash(&dccp_hashinfo, newsk);
-   inet_inherit_port(&dccp_hashinfo, sk, newsk);
+   __inet6_hash(newsk);
+   inet_inherit_port(sk, newsk);

    return newsk;

@@ -1054,6 +1048,7 @@ static struct inet_connection_sock_af_ops dccp_ipv6_af_ops = {
    .getsockopt    = ipv6_getsockopt,
    .addr2sockaddr = inet6_csk_addr2sockaddr,
    .sockaddr_len  = sizeof(struct sockaddr_in6),
+   .bind_conflict = inet6_csk_bind_conflict,
#ifdef CONFIG_COMPAT
    .compat_setsockopt = compat_ipv6_setsockopt,
    .compat_getsockopt = compat_ipv6_getsockopt,
@@ -1123,9 +1118,9 @@ static struct proto dccp_v6_prot = {
    .recvmmsg      = dccp_recvmmsg,
    .backlog_rcv   = dccp_v6_do_rcv,
    .hash          = dccp_v6_hash,
-   .unhash        = dccp_unhash,
+   .unhash        = inet_unhash,
    .accept        = inet_csk_accept,
-   .get_port      = dccp_v6_get_port,
+   .get_port      = inet_csk_get_port,
    .shutdown      = dccp_shutdown,
    .destroy       = dccp_v6_destroy_sock,
    .orphan_count  = &dccp_orphan_count,
@@ -1133,6 +1128,7 @@ static struct proto dccp_v6_prot = {
    .obj_size      = sizeof(struct dccp6_sock),
    .rsk_prot      = &dccp6_request_sock_ops,
    .twsk_prot     = &dccp6_timewait_sock_ops,
+   .hashinfo      = &dccp_hashinfo,
#ifdef CONFIG_COMPAT

```

```

.compat_setsockopt = compat_dccp_setsockopt,
.compat_getsockopt = compat_dccp_getsockopt,
diff --git a/net/dccp/proto.c b/net/dccp/proto.c
index 0bed4a6..e3f5d37 100644
--- a/net/dccp/proto.c
+++ b/net/dccp/proto.c
@@ -78,7 +78,7 @@ void dccp_set_state(struct sock *sk, const int state)
     sk->sk_prot->unhash(sk);
     if (inet_csk(sk)->icsk_bind_hash != NULL &&
         !(sk->sk_userlocks & SOCK_BINDPORT_LOCK))
-    inet_put_port(&dccp_hashinfo, sk);
+    inet_put_port(sk);
     /* fall through */
     default:
         if (oldstate == DCCP_OPEN)
@@ -173,20 +173,6 @@ const char *dccp_state_name(const int state)

EXPORT_SYMBOL_GPL(dccp_state_name);

-void dccp_hash(struct sock *sk)
-{
-    inet_hash(&dccp_hashinfo, sk);
-}
-
-EXPORT_SYMBOL_GPL(dccp_hash);
-
-void dccp_unhash(struct sock *sk)
-{
-    inet_unhash(&dccp_hashinfo, sk);
-}
-
-EXPORT_SYMBOL_GPL(dccp_unhash);
-
int dccp_init_sock(struct sock *sk, const __u8 ctl_sock_initialized)
{
    struct dccp_sock *dp = dccp_sk(sk);
@@ -268,7 +268,7 @@ int dccp_destroy_sock(struct sock *sk)

    /* Clean up a referenced DCCP bind bucket. */
    if (inet_csk(sk)->icsk_bind_hash != NULL)
-    inet_put_port(&dccp_hashinfo, sk);
+    inet_put_port(sk);

    kfree(dp->dccps_service_list);
    dp->dccps_service_list = NULL;
diff --git a/net/ipv4/inet_connection_sock.c b/net/ipv4/inet_connection_sock.c
index de5a41d..b189278 100644
--- a/net/ipv4/inet_connection_sock.c

```



```

+++ b/net/ipv4/inet_connection_sock.c
@@ -78,11 +78,9 @@ EXPORT_SYMBOL_GPL(inet_csk_bind_conflict);
/* Obtain a reference to a local port for the given sock,
 * if snum is zero it means select any available local port.
 */
-int inet_csk_get_port(struct inet_hashinfo *hashinfo,
-    struct sock *sk, unsigned short snum,
-    int (*bind_conflict)(const struct sock *sk,
-    const struct inet_bind_bucket *tb))
+int inet_csk_get_port(struct sock *sk, unsigned short snum)
{
+ struct inet_hashinfo *hashinfo = sk->sk_prot->hashinfo;
+ struct inet_bind_hashbucket *head;
+ struct hlist_node *node;
+ struct inet_bind_bucket *tb;
@@ -142,7 +140,7 @@ tb_found:
    goto success;
} else {
    ret = 1;
- if (bind_conflict(sk, tb))
+ if (inet_csk(sk)->icsk_af_ops->bind_conflict(sk, tb))
    goto fail_unlock;
}
}
diff --git a/net/ipv4/inet_hashtables.c b/net/ipv4/inet_hashtables.c
index 48d4500..90f422c 100644
--- a/net/ipv4/inet_hashtables.c
+++ b/net/ipv4/inet_hashtables.c
@@ -66,8 +66,9 @@ void inet_bind_hash(struct sock *sk, struct inet_bind_bucket *tb,
/*
 * Get rid of any references to a local port held by the given sock.
 */
-static void __inet_put_port(struct inet_hashinfo *hashinfo, struct sock *sk)
+static void __inet_put_port(struct sock *sk)
{
+ struct inet_hashinfo *hashinfo = sk->sk_prot->hashinfo;
+ const int bhash = inet_bhashfn(inet_sk(sk)->num, hashinfo->bhash_size);
+ struct inet_bind_hashbucket *head = &hashinfo->bhash[bhash];
+ struct inet_bind_bucket *tb;
@@ -81,10 +82,10 @@ static void __inet_put_port(struct inet_hashinfo *hashinfo, struct sock
*sk)
    spin_unlock(&head->lock);
}

-void inet_put_port(struct inet_hashinfo *hashinfo, struct sock *sk)
+void inet_put_port(struct sock *sk)
{
    local_bh_disable();

```

```

- __inet_put_port(hashinfo, sk);
+ __inet_put_port(sk);
  local_bh_enable();
}

@@ -317,8 +318,9 @@ static inline u32 inet_sk_port_offset(const struct sock *sk)
    inet->dport);
}

-void __inet_hash_nolisten(struct inet_hashinfo *hashinfo, struct sock *sk)
+void __inet_hash_nolisten(struct sock *sk)
{
+ struct inet_hashinfo *hashinfo = sk->sk_prot->hashinfo;
  struct hlist_head *list;
  rwlock_t *lock;
  struct inet_eshash_bucket *head;
@@ -337,13 +339,14 @@ void __inet_hash_nolisten(struct inet_hashinfo *hashinfo, struct sock
*sk)
}
EXPORT_SYMBOL_GPL(__inet_hash_nolisten);

-void __inet_hash(struct inet_hashinfo *hashinfo, struct sock *sk)
+static void __inet_hash(struct sock *sk)
{
+ struct inet_hashinfo *hashinfo = sk->sk_prot->hashinfo;
  struct hlist_head *list;
  rwlock_t *lock;

  if (sk->sk_state != TCP_LISTEN) {
- __inet_hash_nolisten(hashinfo, sk);
+ __inet_hash_nolisten(sk);
  return;
}

@@ -357,13 +360,48 @@ void __inet_hash(struct inet_hashinfo *hashinfo, struct sock *sk)
  write_unlock(lock);
  wake_up(&hashinfo->lhash_wait);
}
-EXPORT_SYMBOL_GPL(__inet_hash);
+
+void inet_hash(struct sock *sk)
+{
+ if (sk->sk_state != TCP_CLOSE) {
+  local_bh_disable();
+  __inet_hash(sk);
+  local_bh_enable();
+ }
+}

```

```

+EXPORT_SYMBOL_GPL(inet_hash);
+
+void inet_unhash(struct sock *sk)
+{
+ rwlock_t *lock;
+ struct inet_hashinfo *hashinfo = sk->sk_prot->hashinfo;
+
+ if (sk_unhashed(sk))
+  goto out;
+
+ if (sk->sk_state == TCP_LISTEN) {
+  local_bh_disable();
+  inet_listen_wlock(hashinfo);
+  lock = &hashinfo->lhash_lock;
+ } else {
+  lock = inet_eshash_lockp(hashinfo, sk->sk_hash);
+  write_lock_bh(lock);
+ }
+
+ if (__sk_del_node_init(sk))
+  sock_prot_inuse_add(sk->sk_prot, -1);
+ write_unlock_bh(lock);
+out:
+ if (sk->sk_state == TCP_LISTEN)
+  wake_up(&hashinfo->lhash_wait);
+}
+EXPORT_SYMBOL_GPL(inet_unhash);

int __inet_hash_connect(struct inet_timewait_death_row *death_row,
    struct sock *sk,
    int (*check_established)(struct inet_timewait_death_row *,
        struct sock *, __u16, struct inet_timewait_sock **),
- void (*hash)(struct inet_hashinfo *, struct sock *))
+ void (*hash)(struct sock *sk))
{
    struct inet_hashinfo *hinfo = death_row->hashinfo;
    const unsigned short snum = inet_sk(sk)->num;
@@ -427,7 +465,7 @@ ok:
    inet_bind_hash(sk, tb, port);
    if (sk_unhashed(sk)) {
        inet_sk(sk)->sport = htons(port);
-    hash(hinfo, sk);
+    hash(sk);
    }
    spin_unlock(&head->lock);

@@ -444,7 +482,7 @@ ok:
    tb = inet_csk(sk)->icsk_bind_hash;

```

```

spin_lock_bh(&head->lock);
if (sk_head(&tb->owners) == sk && !sk->sk_bind_node.next) {
- hash(hinfo, sk);
+ hash(sk);
    spin_unlock_bh(&head->lock);
    return 0;
} else {
diff --git a/net/ipv4/tcp.c b/net/ipv4/tcp.c
index a0d373b..071e83a 100644
--- a/net/ipv4/tcp.c
+++ b/net/ipv4/tcp.c
@@ -1669,7 +1669,7 @@ void tcp_set_state(struct sock *sk, int state)
    sk->sk_prot->unhash(sk);
    if (inet_csk(sk)->icsk_bind_hash &&
        !(sk->sk_userlocks & SOCK_BINDPORT_LOCK))
- inet_put_port(&tcp_hashinfo, sk);
+ inet_put_port(sk);
    /* fall through */
    default:
        if (oldstate==TCP_ESTABLISHED)
diff --git a/net/ipv4/tcp_ipv4.c b/net/ipv4/tcp_ipv4.c
index 77c1939..63414ea 100644
--- a/net/ipv4/tcp_ipv4.c
+++ b/net/ipv4/tcp_ipv4.c
@@ -108,22 +108,6 @@ struct inet_hashinfo __cacheline_aligned tcp_hashinfo = {
    .lhash_wait = __WAIT_QUEUE_HEAD_INITIALIZER(tcp_hashinfo.lhash_wait),
};

-static int tcp_v4_get_port(struct sock *sk, unsigned short snum)
-{
- return inet_csk_get_port(&tcp_hashinfo, sk, snum,
-    inet_csk_bind_conflict);
-}
-
-static void tcp_v4_hash(struct sock *sk)
-{
- inet_hash(&tcp_hashinfo, sk);
-}
-
-void tcp_unhash(struct sock *sk)
-{
- inet_unhash(&tcp_hashinfo, sk);
-}
-
static inline __u32 tcp_v4_init_sequence(struct sk_buff *skb)
{
    return secure_tcp_sequence_number(ip_hdr(skb)->daddr,
@@ -1478,8 +1462,8 @@ struct sock *tcp_v4_syn_recv_sock(struct sock *sk, struct sk_buff

```

```

*skb,
}
#endif

- __inet_hash_nolisten(&tcp_hashinfo, newsk);
- __inet_inherit_port(&tcp_hashinfo, sk, newsk);
+ __inet_hash_nolisten(newsk);
+ __inet_inherit_port(sk, newsk);

return newsk;

@@ -1827,6 +1811,7 @@ struct inet_connection_sock_af_ops ipv4_specific = {
    .getsockopt = ip_getsockopt,
    .addr2sockaddr = inet_csk_addr2sockaddr,
    .sockaddr_len = sizeof(struct sockaddr_in),
+ .bind_conflict = inet_csk_bind_conflict,
#ifdef CONFIG_COMPAT
    .compat_setsockopt = compat_ip_setsockopt,
    .compat_getsockopt = compat_ip_getsockopt,
@@ -1926,7 +1911,7 @@ int tcp_v4_destroy_sock(struct sock *sk)

    /* Clean up a referenced TCP bind bucket. */
    if (inet_csk(sk)->icsk_bind_hash)
- inet_put_port(&tcp_hashinfo, sk);
+ inet_put_port(sk);

    /*
     * If sendmsg cached page exists, toss it.
@@ -2435,9 +2420,9 @@ struct proto tcp_prot = {
    .getsockopt = tcp_getsockopt,
    .recvmsg = tcp_recvmsg,
    .backlog_rcv = tcp_v4_do_rcv,
- .hash = tcp_v4_hash,
- .unhash = tcp_unhash,
- .get_port = tcp_v4_get_port,
+ .hash = inet_hash,
+ .unhash = inet_unhash,
+ .get_port = inet_csk_get_port,
    .enter_memory_pressure = tcp_enter_memory_pressure,
    .sockets_allocated = &tcp_sockets_allocated,
    .orphan_count = &tcp_orphan_count,
@@ -2450,6 +2435,7 @@ struct proto tcp_prot = {
    .obj_size = sizeof(struct tcp_sock),
    .twsk_prot = &tcp_timewait_sock_ops,
    .rsk_prot = &tcp_request_sock_ops,
+ .hashinfo = &tcp_hashinfo,
#ifdef CONFIG_COMPAT
    .compat_setsockopt = compat_tcp_setsockopt,

```

```

.compat_getsockopt = compat_tcp_getsockopt,
@@ -2467,7 +2453,6 @@ void __init tcp_v4_init(struct net_proto_family *ops)
EXPORT_SYMBOL(ipv4_specific);
EXPORT_SYMBOL(tcp_hashinfo);
EXPORT_SYMBOL(tcp_prot);
-EXPORT_SYMBOL(tcp_unhash);
EXPORT_SYMBOL(tcp_v4_conn_request);
EXPORT_SYMBOL(tcp_v4_connect);
EXPORT_SYMBOL(tcp_v4_do_rcv);
diff --git a/net/ipv6/inet6_hashtables.c b/net/ipv6/inet6_hashtables.c
index d325a99..43f3993 100644
--- a/net/ipv6/inet6_hashtables.c
+++ b/net/ipv6/inet6_hashtables.c
@@ -22,9 +22,9 @@
#include <net/inet6_hashtables.h>
#include <net/ip.h>

-void __inet6_hash(struct inet_hashinfo *hashinfo,
- struct sock *sk)
+void __inet6_hash(struct sock *sk)
{
+ struct inet_hashinfo *hashinfo = sk->sk_prot->hashinfo;
  struct hlist_head *list;
  rwlock_t *lock;

diff --git a/net/ipv6/tcp_ipv6.c b/net/ipv6/tcp_ipv6.c
index 59d0029..12750f2 100644
--- a/net/ipv6/tcp_ipv6.c
+++ b/net/ipv6/tcp_ipv6.c
@@ -86,12 +86,6 @@ static struct tcp_sock_af_ops tcp_sock_ipv6_specific;
static struct tcp_sock_af_ops tcp_sock_ipv6_mapped_specific;
#endif

-static int tcp_v6_get_port(struct sock *sk, unsigned short snum)
-{
- return inet_csk_get_port(&tcp_hashinfo, sk, snum,
- inet6_csk_bind_conflict);
-}
-
static void tcp_v6_hash(struct sock *sk)
{
  if (sk->sk_state != TCP_CLOSE) {
@@ -100,7 +94,7 @@ static void tcp_v6_hash(struct sock *sk)
  return;
}
local_bh_disable();
- __inet6_hash(&tcp_hashinfo, sk);
+ __inet6_hash(sk);

```

```

    local_bh_enable();
}
}
@@ -1504,8 +1498,8 @@ static struct sock * tcp_v6_syn_recv_sock(struct sock *sk, struct
sk_buff *skb,
}
#endif

- __inet6_hash(&tcp_hashinfo, newsk);
- inet_inherit_port(&tcp_hashinfo, sk, newsk);
+ __inet6_hash(newsk);
+ inet_inherit_port(sk, newsk);

return newsk;

@@ -1833,6 +1827,7 @@ static struct inet_connection_sock_af_ops ipv6_specific = {
.getsockopt = ipv6_getsockopt,
.addr2sockaddr = inet6_csk_addr2sockaddr,
.sockaddr_len = sizeof(struct sockaddr_in6),
+ .bind_conflict = inet6_csk_bind_conflict,
#ifdef CONFIG_COMPAT
.compat_setsockopt = compat_ipv6_setsockopt,
.compat_getsockopt = compat_ipv6_getsockopt,
@@ -1864,6 +1859,7 @@ static struct inet_connection_sock_af_ops ipv6_mapped = {
.getsockopt = ipv6_getsockopt,
.addr2sockaddr = inet6_csk_addr2sockaddr,
.sockaddr_len = sizeof(struct sockaddr_in6),
+ .bind_conflict = inet6_csk_bind_conflict,
#ifdef CONFIG_COMPAT
.compat_setsockopt = compat_ipv6_setsockopt,
.compat_getsockopt = compat_ipv6_getsockopt,
@@ -2127,8 +2123,8 @@ struct proto tcpv6_prot = {
.recvmsg = tcp_recvmsg,
.backlog_rcv = tcp_v6_do_rcv,
.hash = tcp_v6_hash,
- .unhash = tcp_unhash,
- .get_port = tcp_v6_get_port,
+ .unhash = inet_unhash,
+ .get_port = inet_csk_get_port,
.enter_memory_pressure = tcp_enter_memory_pressure,
.sockets_allocated = &tcp_sockets_allocated,
.memory_allocated = &tcp_memory_allocated,
@@ -2141,6 +2137,7 @@ struct proto tcpv6_prot = {
.obj_size = sizeof(struct tcp6_sock),
.twsk_prot = &tcp6_timewait_sock_ops,
.rsk_prot = &tcp6_request_sock_ops,
+ .hashinfo = &tcp_hashinfo,
#ifdef CONFIG_COMPAT

```

```
.compat_setsockopt = compat_tcp_setsockopt,  
.compat_getsockopt = compat_tcp_getsockopt,
```
