
Subject: [PATCH] net: Add network namespace clone & unshare support.

Posted by [ebiederm](#) on Wed, 26 Sep 2007 23:49:54 GMT

[View Forum Message](#) <> [Reply to Message](#)

This patch allows you to create a new network namespace using sys_clone, or sys_unshare.

As the network namespace is still experimental and under development clone and unshare support is only made available when CONFIG_NET_NS is selected at compile time.

As this patch introduces network namespace support into code paths that exist when the CONFIG_NET is not selected there are a few additions made to net_namespace.h to allow a few more functions to be used when the networking stack is not compiled in.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

```
include/linux/sched.h      |  1 +
include/net/net_namespace.h | 18 ++++++
kernel/fork.c              |  3 +-
kernel/nsproxy.c           | 15 ++++++
net/Kconfig                |  8 ++++++
net/core/net_namespace.c   | 43 ++++++
6 files changed, 83 insertions(+), 5 deletions(-)
```

diff --git a/include/linux/sched.h b/include/linux/sched.h

index a01ac6d..e10a0a8 100644

--- a/include/linux/sched.h

+++ b/include/linux/sched.h

@@ -27,6 +27,7 @@

#define CLONE_NEWUTS 0x04000000 /* New utsname group? */

#define CLONE_NEWIPC 0x08000000 /* New ipc */

#define CLONE_NEWUSER 0x10000000 /* New user namespace */

+#define CLONE_NEWNET 0x20000000 /* New network namespace */

/*

* Scheduling policies

diff --git a/include/net/net_namespace.h b/include/net/net_namespace.h

index ac8f830..3ea4194 100644

--- a/include/net/net_namespace.h

+++ b/include/net/net_namespace.h

@@ -38,11 +38,23 @@ extern struct net init_net;

extern struct list_head net_namespace_list;

+#ifdef CONFIG_NET

+extern struct net *copy_net_ns(unsigned long flags, struct net *net_ns);

```

+ #else
+ static inline struct net *copy_net_ns(unsigned long flags, struct net *net_ns)
+ {
+ /* There is nothing to copy so this is a noop */
+ return net_ns;
+ }
+ #endif
+
+ extern void __put_net(struct net *net);

static inline struct net *get_net(struct net *net)
{
+ #ifdef CONFIG_NET
+ atomic_inc(&net->count);
+ #endif
+ return net;
+ }

@@ -60,19 +72,25 @@ static inline struct net *maybe_get_net(struct net *net)

static inline void put_net(struct net *net)
{
+ #ifdef CONFIG_NET
+ if (atomic_dec_and_test(&net->count))
+ __put_net(net);
+ #endif
+ }

static inline struct net *hold_net(struct net *net)
{
+ #ifdef CONFIG_NET
+ atomic_inc(&net->use_count);
+ #endif
+ return net;
+ }

static inline void release_net(struct net *net)
{
+ #ifdef CONFIG_NET
+ atomic_dec(&net->use_count);
+ #endif
+ }

extern void net_lock(void);
diff --git a/kernel/fork.c b/kernel/fork.c
index 33f12f4..5e67f90 100644
--- a/kernel/fork.c
+++ b/kernel/fork.c

```

```

@@ -1608,7 +1608,8 @@ asmlinkage long sys_unshare(unsigned long unshare_flags)
    err = -EINVAL;
    if (unshare_flags & ~(CLONE_THREAD|CLONE_FS|CLONE_NEWNS|CLONE_SIGHAND|
        CLONE_VM|CLONE_FILES|CLONE_SYSVSEM|
-   CLONE_NEWUTS|CLONE_NEWIPC|CLONE_NEWUSER))
+   CLONE_NEWUTS|CLONE_NEWIPC|CLONE_NEWUSER|
+   CLONE_NEWNET))
        goto bad_unshare_out;

    if ((err = unshare_thread(unshare_flags)))
diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c
index a4fb7d4..f1dec2 100644
--- a/kernel/nsproxy.c
+++ b/kernel/nsproxy.c
@@ -20,6 +20,7 @@
#include <linux/mnt_namespace.h>
#include <linux/utsname.h>
#include <linux/pid_namespace.h>
+#include <net/net_namespace.h>

static struct kmem_cache *nsproxy_cache;

@@ -98,8 +99,17 @@ static struct nsproxy *create_new_namespaces(unsigned long flags,
    goto out_user;
}

+ new_nsp->net_ns = copy_net_ns(flags, tsk->nsproxy->net_ns);
+ if (IS_ERR(new_nsp->net_ns)) {
+   err = PTR_ERR(new_nsp->net_ns);
+   goto out_net;
+ }
+
    return new_nsp;

+out_net:
+ if (new_nsp->user_ns)
+   put_user_ns(new_nsp->user_ns);
+out_user:
+ if (new_nsp->pid_ns)
+   put_pid_ns(new_nsp->pid_ns);
@@ -132,7 +142,7 @@ int copy_namespaces(unsigned long flags, struct task_struct *tsk)

    get_nsproxy(old_ns);

- if (!(flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC | CLONE_NEWUSER)))
+ if (!(flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC | CLONE_NEWUSER |
    CLONE_NEWNET)))
    return 0;

```

```

    if (!capable(CAP_SYS_ADMIN)) {
@@ -164,6 +174,7 @@ void free_nsproxy(struct nsproxy *ns)
    put_pid_ns(ns->pid_ns);
    if (ns->user_ns)
        put_user_ns(ns->user_ns);
+ put_net(ns->net_ns);
    kmem_cache_free(nsproxy_cachep, ns);
    }

@@ -177,7 +188,7 @@ int unshare_nsproxy_namespaces(unsigned long unshare_flags,
    int err = 0;

    if (!(unshare_flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC |
-        CLONE_NEWUSER)))
+        CLONE_NEWUSER | CLONE_NEWNET)))
        return 0;

    if (!capable(CAP_SYS_ADMIN))
diff --git a/net/Kconfig b/net/Kconfig
index cdba08c..ab4e6da 100644
--- a/net/Kconfig
+++ b/net/Kconfig
@@ -27,6 +27,14 @@ if NET

menu "Networking options"

+config NET_NS
+ bool "Network namespace support"
+ default n
+ depends on EXPERIMENTAL && !SYSFS
+ help
+   Allow user space to create what appear to be multiple instances
+   of the network stack.
+
    source "net/packet/Kconfig"
    source "net/unix/Kconfig"
    source "net/xfrm/Kconfig"
diff --git a/net/core/net_namespace.c b/net/core/net_namespace.c
index 0e6cb02..e478e35 100644
--- a/net/core/net_namespace.c
+++ b/net/core/net_namespace.c
@@ -4,6 +4,7 @@
#include <linux/slab.h>
#include <linux/list.h>
#include <linux/delay.h>
+#include <linux/sched.h>
#include <net/net_namespace.h>

```

```

/*
@@ -32,12 +33,10 @@ void net_unlock(void)
    mutex_unlock(&net_list_mutex);
}

-#if 0
static struct net *net_alloc(void)
{
    return kmem_cache_alloc(net_cachep, GFP_KERNEL);
}
-#endif

static void net_free(struct net *net)
{
@@ -128,6 +127,46 @@ out_undo:
    goto out;
}

+struct net *copy_net_ns(unsigned long flags, struct net *old_net)
+{
+ struct net *new_net = NULL;
+ int err;
+
+ get_net(old_net);
+
+ if (!(flags & CLONE_NEWNET))
+     return old_net;
+
+ #ifndef CONFIG_NET_NS
+ return ERR_PTR(-EINVAL);
+ #endif
+
+ err = -ENOMEM;
+ new_net = net_alloc();
+ if (!new_net)
+     goto out;
+
+ mutex_lock(&net_mutex);
+ err = setup_net(new_net);
+ if (err)
+     goto out_unlock;
+
+ net_lock();
+ list_add_tail(&new_net->list, &net_namespace_list);
+ net_unlock();
+
+
+

```

```
+out_unlock:
+ mutex_unlock(&net_mutex);
+out:
+ put_net(old_net);
+ if (err) {
+  net_free(new_net);
+  new_net = ERR_PTR(err);
+ }
+ return new_net;
+}
+
+static int __init net_ns_init(void)
+{
+  int err;
+}
+
+--
1.5.3.rc6.17.g1911
```

Containers mailing list
Containers@lists.linux-foundation.org
<https://lists.linux-foundation.org/mailman/listinfo/containers>
