Subject: [PATCH v3 0/2] SUNRPC: separate per-net data creation from service
Posted by Stanislav Kinsbursky on Wed, 02 May 2012 12:08:29 GMT
View Forum Message <> Reply to Message

creation

v3: "SUNRPC: new svc_bind() routine introduced" patch was squashed with the
"SUNRPC: check rpcbind clients usage counter before decrement" patch.

v2: Increase per-net usage counted in lockd_up_net().

This is a cleanup patch set.
It will be followed my LockD start/stop cleanup patch set and NFS callback service containerization patch set (yes, I forgot to implement it).

Today per-net data is created with service, and then is service is starting in other network namespace. And thus it's destroyed with service too. Moreover, network context for destroying of per-net data is taken from current process. This is correct, but code looks ugly.
This patch set separates per-net data allocation from service allocation and destruction.
IOW, per-net data have to be destroyed by service users - not service itself.

BTW, NFSd code become uglier with this patch set. Sorry.
But I assume, that these new ugly parts will be replaced later by NFSd service containerization code.

The following series implements...

Stanislav Kinsbursky (2):
SUNRPC: new svc_bind() routine introduced
SUNRPC: move per-net operations from svc_destroy()

```
fs/lockd/svc.c            |   33 ++++++++++++++++++++++----------
fs/nfs/callback.c         |   11 ++++++++++
fs/nfsd/nfsctl.c          |    4 ++++
fs/nfsd/nfssvc.c          |   16 ++++++++++++++++
include/linux/sunrpc/svc.h |    1 +
net/sunrpc/rpcb_clnt.c    |   12 +++++++-----
net/sunrpc/svc.c          |   23 +++++++++++++++++------
7 files changed, 70 insertions(+), 30 deletions(-)
```

Subject: [PATCH v3 1/2] SUNRPC: new svc_bind() routine introduced

v3: this patch was squashed with the "SUNRPC: check rpcbind clients usage counter before decrement" patch.

New routine is responsible for service registration in specified network context.
The idea is to separate service creation from per-net operations.
Since registering service with svc_bind() can fail, then service will be destroyed and during destruction it will try to unregister itself from rpcbind. In this case unregister have to be skipped.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
 fs/lockd/svc.c | 6 ++++++
 fs/nfs/callback.c | 8 ++++++++
 fs/nfsd/nfssvc.c | 9 +++++++++
 include/linux/sunrpc/svc.h | 1 +
 net/sunrpc/rpcb_clnt.c | 12 +++++++-----
 net/sunrpc/svc.c | 19 +++++++++++++---------
 6 files changed, 41 insertions(+), 14 deletions(-)

diff --git a/fs/lockd/svc.c b/fs/lockd/svc.c
index 1ead075..b7e92ed 100644
--- a/fs/lockd/svc.c
+++ b/fs/lockd/svc.c
@@ -324,6 +324,12 @@ int lockd_up(struct net *net)
   goto out;
  }

+ error = svc_bind(serv, net);
+ if (error < 0) {
+ printk(KERN_WARNING "lockd_up: bind service failed\n");
+ goto destroy_and_out;
+ }
+
  error = make_socks(serv, net);
  if (error < 0)
   goto destroy_and_out;
diff --git a/fs/nfs/callback.c b/fs/nfs/callback.c
index eb95f50..26b38fb 100644
--- a/fs/nfs/callback.c
+++ b/fs/nfs/callback.c
@@ -17,6 +17,7 @@
 #include <linux/kthread.h>
 #include <linux/sunrpc/svcauth_gss.h>
 #include <linux/sunrpc/bc_xprt.h>
+#include <linux/nsproxy.h>

```
 #include <net/inet_sock.h>

@@ -253,6 +254,7 @@ int nfs_callback_up(u32 minorversion, struct rpc_xprt *xprt)
  char svc_name[12];
  int ret = 0;
  int minorversion_setup;
+ struct net *net = current->nsproxy->net_ns;

  mutex_lock(&nfs_callback_mutex);
  if (cb_info->users++ || cb_info->task != NULL) {
@@ -265,6 +267,12 @@ int nfs_callback_up(u32 minorversion, struct rpc_xprt *xprt)
   goto out_err;
  }

+ ret = svc_bind(serv, net);
+ if (ret < 0) {
+  printk(KERN_WARNING "NFS: bind callback service failed\n");
+  goto out_err;
+ }
+
  minorversion_setup =  nfs_minorversion_callback_svc_setup(minorversion,
      serv, xprt, &rqstp, &callback_svc);
  if (!minorversion_setup) {
diff --git a/fs/nfsd/nfssvc.c b/fs/nfsd/nfssvc.c
index cb4d51d..0762f3c 100644
--- a/fs/nfsd/nfssvc.c
+++ b/fs/nfsd/nfssvc.c
@@ -11,6 +11,7 @@
 #include <linux/module.h>
 #include <linux/fs_struct.h>
 #include <linux/swap.h>
+#include <linux/nsproxy.h>

 #include <linux/sunrpc/stats.h>
 #include <linux/sunrpc/svcsock.h>
@@ -330,6 +331,8 @@ static int nfsd_get_default_max_blksize(void)

 int nfsd_create_serv(void)
 {
+ int error;
+
  WARN_ON(!mutex_is_locked(&nfsd_mutex));
  if (nfsd_serv) {
   svc_get(nfsd_serv);
@@ -343,6 +346,12 @@ int nfsd_create_serv(void)
  if (nfsd_serv == NULL)
   return -ENOMEM;
```

```
+ error = svc_bind(nfsd_serv, current->nsproxy->net_ns);
+ if (error < 0) {
+  svc_destroy(nfsd_serv);
+  return error;
+ }
+
  set_max_drc();
  do_gettimeofday(&nfssvc_boot);  /* record boot time */
  return 0;
diff --git a/include/linux/sunrpc/svc.h b/include/linux/sunrpc/svc.h
index 51b29ac..2b43e02 100644
--- a/include/linux/sunrpc/svc.h
+++ b/include/linux/sunrpc/svc.h
@@ -416,6 +416,7 @@ struct svc_procedure {
  */
 int svc_rpcb_setup(struct svc_serv *serv, struct net *net);
 void svc_rpcb_cleanup(struct svc_serv *serv, struct net *net);
+int svc_bind(struct svc_serv *serv, struct net *net);
 struct svc_serv *svc_create(struct svc_program *, unsigned int,
        void (*shutdown)(struct svc_serv *, struct net *net));
 struct svc_rqst *svc_prepare_thread(struct svc_serv *serv,
diff --git a/net/sunrpc/rpcb_clnt.c b/net/sunrpc/rpcb_clnt.c
index 78ac39f..4c38b33 100644
--- a/net/sunrpc/rpcb_clnt.c
+++ b/net/sunrpc/rpcb_clnt.c
@@ -180,14 +180,16 @@ void rpcb_put_local(struct net *net)
  struct sunrpc_net *sn = net_generic(net, sunrpc_net_id);
  struct rpc_clnt *clnt = sn->rpcb_local_clnt;
  struct rpc_clnt *clnt4 = sn->rpcb_local_clnt4;
- int shutdown;
+ int shutdown = 0;

  spin_lock(&sn->rpcb_clnt_lock);
- if (--sn->rpcb_users == 0) {
-  sn->rpcb_local_clnt = NULL;
-  sn->rpcb_local_clnt4 = NULL;
+ if (sn->rpcb_users) {
+  if (--sn->rpcb_users == 0) {
+   sn->rpcb_local_clnt = NULL;
+   sn->rpcb_local_clnt4 = NULL;
+  }
+  shutdown = !sn->rpcb_users;
  }
- shutdown = !sn->rpcb_users;
  spin_unlock(&sn->rpcb_clnt_lock);

  if (shutdown) {
```

```
diff --git a/net/sunrpc/svc.c b/net/sunrpc/svc.c
index 4153846..e6d542c 100644
--- a/net/sunrpc/svc.c
+++ b/net/sunrpc/svc.c
@@ -407,6 +407,14 @@ static int svc_uses_rpcbind(struct svc_serv *serv)
  return 0;
 }

+int svc_bind(struct svc_serv *serv, struct net *net)
+{
+ if (!svc_uses_rpcbind(serv))
+  return 0;
+ return svc_rpcb_setup(serv, net);
+}
+EXPORT_SYMBOL_GPL(svc_bind);
+
 /*
  * Create an RPC service
  */
@@ -471,15 +479,8 @@ __svc_create(struct svc_program *prog, unsigned int bufsize, int
npools,
   spin_lock_init(&pool->sp_lock);
  }

- if (svc_uses_rpcbind(serv)) {
-  if (svc_rpcb_setup(serv, current->nsproxy->net_ns) < 0) {
-   kfree(serv->sv_pools);
-   kfree(serv);
-   return NULL;
-  }
-  if (!serv->sv_shutdown)
-   serv->sv_shutdown = svc_rpcb_cleanup;
- }
+ if (svc_uses_rpcbind(serv) && (!serv->sv_shutdown))
+  serv->sv_shutdown = svc_rpcb_cleanup;

  return serv;
 }
```

Subject: [PATCH v3 2/2] SUNRPC: move per-net operations from svc_destroy()
Posted by Stanislav Kinsbursky on Wed, 02 May 2012 12:08:45 GMT
View Forum Message <> Reply to Message

v2:
1) Increase per-net usage counted in lockd_up_net(), because of note 2.

The idea is to separate service destruction and per-net operations, because

these are two different things and it's mix looks ugly.

Notes:
1) For NFS server this patch looks ugly (sorry for that). But these place will be
rewritten soon during NFSd containerization.
2) LockD per-net counter increase int lockd_up() was moved prior to
make_socks() to make lockd_down_net() call safe in case of error.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
 fs/lockd/svc.c | 27 +++++++++++++++++++----------
 fs/nfs/callback.c | 3 +++
 fs/nfsd/nfsctl.c | 4 ++++
 fs/nfsd/nfssvc.c | 7 +++++++
 net/sunrpc/svc.c | 4 ----
 5 files changed, 29 insertions(+), 16 deletions(-)

diff --git a/fs/lockd/svc.c b/fs/lockd/svc.c
index b7e92ed..3250f28 100644
--- a/fs/lockd/svc.c
+++ b/fs/lockd/svc.c
@@ -257,7 +257,7 @@ static int lockd_up_net(struct net *net)
  struct svc_serv *serv = nlmsvc_rqst->rq_server;
  int error;

- if (ln->nlmsvc_users)
+ if (ln->nlmsvc_users++)
   return 0;

  error = svc_rpcb_setup(serv, net);
@@ -272,6 +272,7 @@ static int lockd_up_net(struct net *net)
  svc_rpcb_cleanup(serv, net);
 err_rpcb:
+ ln->nlmsvc_users--;
  return error;
 }

@@ -299,6 +300,7 @@ int lockd_up(struct net *net)
 {
  struct svc_serv *serv;
  int error = 0;
+ struct lockd_net *ln = net_generic(net, lockd_net_id);

  mutex_lock(&nlmsvc_mutex);
  /*
@@ -330,9 +332,11 @@ int lockd_up(struct net *net)
   goto destroy_and_out;

```
  }

+ ln->nlmsvc_users++;
+
  error = make_socks(serv, net);
  if (error < 0)
-  goto destroy_and_out;
+  goto err_start;

  /*
   * Create the kernel thread and wait for it to start.
@@ -344,7 +348,7 @@ int lockd_up(struct net *net)
   printk(KERN_WARNING
    "lockd_up: svc_rqst allocation failed, error=%d\n",
    error);
-  goto destroy_and_out;
+  goto err_start;
  }

  svc_sock_update_bufs(serv);
@@ -358,7 +362,7 @@ int lockd_up(struct net *net)
   nlmsvc_rqst = NULL;
   printk(KERN_WARNING
    "lockd_up: kthread_run failed, error=%d\n", error);
-  goto destroy_and_out;
+  goto err_start;
  }

  /*
@@ -368,14 +372,14 @@ int lockd_up(struct net *net)
 destroy_and_out:
  svc_destroy(serv);
 out:
- if (!error) {
-  struct lockd_net *ln = net_generic(net, lockd_net_id);
-
-  ln->nlmsvc_users++;
+ if (!error)
   nlmsvc_users++;
- }
  mutex_unlock(&nlmsvc_mutex);
  return error;
+
+err_start:
+ lockd_down_net(net);
+ goto destroy_and_out;
 }
 EXPORT_SYMBOL_GPL(lockd_up);
```

```
@@ -386,11 +390,10 @@ void
 lockd_down(struct net *net)
 {
  mutex_lock(&nlmsvc_mutex);
+ lockd_down_net(net);
  if (nlmsvc_users) {
-  if (--nlmsvc_users) {
-   lockd_down_net(net);
+  if (--nlmsvc_users)
    goto out;
-  }
  } else {
   printk(KERN_ERR "lockd_down: no users! task=%p\n",
    nlmsvc_task);
diff --git a/fs/nfs/callback.c b/fs/nfs/callback.c
index 26b38fb..cff3940 100644
--- a/fs/nfs/callback.c
+++ b/fs/nfs/callback.c
@@ -314,6 +314,8 @@ out_err:
  dprintk("NFS: Couldn't create callback socket or server thread; "
   "err = %d\n", ret);
  cb_info->users--;
+ if (serv)
+  svc_shutdown_net(serv, net);
  goto out;
 }

@@ -328,6 +330,7 @@ void nfs_callback_down(int minorversion)
  cb_info->users--;
  if (cb_info->users == 0 && cb_info->task != NULL) {
   kthread_stop(cb_info->task);
+  svc_shutdown_net(cb_info->serv, current->nsproxy->net_ns);
   svc_exit_thread(cb_info->rqst);
   cb_info->serv = NULL;
   cb_info->rqst = NULL;
diff --git a/fs/nfsd/nfsctl.c b/fs/nfsd/nfsctl.c
index 7269988..efb3818 100644
--- a/fs/nfsd/nfsctl.c
+++ b/fs/nfsd/nfsctl.c
@@ -672,6 +672,8 @@ static ssize_t __write_ports_addfd(char *buf)

  err = svc_addsock(nfsd_serv, fd, buf, SIMPLE_TRANSACTION_LIMIT);
  if (err < 0) {
+  if (nfsd_serv->sv_nrthreads == 1)
+   svc_shutdown_net(nfsd_serv, current->nsproxy->net_ns);
   svc_destroy(nfsd_serv);
   return err;
```

```
  }
@@ -740,6 +742,8 @@ out_close:
    svc_xprt_put(xprt);
  }
 out_err:
+ if (nfsd_serv->sv_nrthreads == 1)
+  svc_shutdown_net(nfsd_serv, current->nsproxy->net_ns);
  svc_destroy(nfsd_serv);
  return err;
 }
diff --git a/fs/nfsd/nfssvc.c b/fs/nfsd/nfssvc.c
index 0762f3c..3fffe6c 100644
--- a/fs/nfsd/nfssvc.c
+++ b/fs/nfsd/nfssvc.c
@@ -426,6 +426,9 @@ int nfsd_set_nrthreads(int n, int *nthreads)
    if (err)
     break;
  }
+
+ if (nfsd_serv->sv_nrthreads == 1)
+  svc_shutdown_net(nfsd_serv, current->nsproxy->net_ns);
  svc_destroy(nfsd_serv);

  return err;
@@ -473,6 +476,8 @@ out_shutdown:
  if (error < 0 && !nfsd_up_before)
   nfsd_shutdown();
 out_destroy:
+ if (nfsd_serv->sv_nrthreads == 1)
+  svc_shutdown_net(nfsd_serv, current->nsproxy->net_ns);
  svc_destroy(nfsd_serv);  /* Release server */
 out:
  mutex_unlock(&nfsd_mutex);
@@ -670,6 +675,8 @@ int nfsd_pool_stats_release(struct inode *inode, struct file *file)
  int ret = seq_release(inode, file);
  mutex_lock(&nfsd_mutex);
  /* this function really, really should have been called svc_put() */
+ if (nfsd_serv->sv_nrthreads == 1)
+  svc_shutdown_net(nfsd_serv, current->nsproxy->net_ns);
  svc_destroy(nfsd_serv);
  mutex_unlock(&nfsd_mutex);
  return ret;
diff --git a/net/sunrpc/svc.c b/net/sunrpc/svc.c
index e6d542c..b7210f5 100644
--- a/net/sunrpc/svc.c
+++ b/net/sunrpc/svc.c
@@ -537,8 +537,6 @@ EXPORT_SYMBOL_GPL(svc_shutdown_net);
 void
```

```
 svc_destroy(struct svc_serv *serv)
 {
- struct net *net = current->nsproxy->net_ns;
-
  dprintk("svc: svc_destroy(%s, %d)\n",
     serv->sv_program->pg_name,
     serv->sv_nrthreads);
@@ -553,8 +551,6 @@ svc_destroy(struct svc_serv *serv)

  del_timer_sync(&serv->sv_temptimer);

- svc_shutdown_net(serv, net);
-
  /*
   * The last user is gone and thus all sockets have to be destroyed to
   * the point. Check this.
```

Subject: Re: [PATCH v3 0/2] SUNRPC: separate per-net data creation from service
Posted by bfields on Wed, 02 May 2012 21:58:37 GMT
View Forum Message <> Reply to Message

Thanks, applying.

--b.

On Wed, May 02, 2012 at 04:08:29PM +0400, Stanislav Kinsbursky wrote:
> creation
>
> v3: "SUNRPC: new svc_bind() routine introduced" patch was squashed with the
>   "SUNRPC: check rpcbind clients usage counter before decrement" patch.
>
> v2: Increase per-net usage counted in lockd_up_net().
>
> This is a cleanup patch set.
> It will be followed my LockD start/stop cleanup patch set and NFS callback
> service containerization patch set (yes, I forgot to implement it).
>
> Today per-net data is created with service, and then is service is starting in
> other network namespace. And thus it's destroyed with service too. Moreover,
> network context for destroying of per-net data is taken from current process.
> This is correct, but code looks ugly.
> This patch set separates per-net data allocation from service allocation and
> destruction.
> IOW, per-net data have to be destroyed by service users - not service itself.
>
> BTW, NFSd code become uglier with this patch set. Sorry.
> But I assume, that these new ugly parts will be replaced later by NFSd service

> containerization code.
>
> The following series implements...
>
> ---
>
> Stanislav Kinsbursky (2):
> SUNRPC: new svc_bind() routine introduced
> SUNRPC: move per-net operations from svc_destroy()
>
>
> fs/lockd/svc.c | 33 ++++++++++++++++++++++----------
> fs/nfs/callback.c | 11 +++++++++++
> fs/nfsd/nfsctl.c | 4 ++++
> fs/nfsd/nfssvc.c | 16 ++++++++++++++++
> include/linux/sunrpc/svc.h | 1 +
> net/sunrpc/rpcb_clnt.c | 12 +++++++-----
> net/sunrpc/svc.c | 23 ++++++++++----------
> 7 files changed, 70 insertions(+), 30 deletions(-)
>

Subject: Re: [PATCH v3 0/2] SUNRPC: separate per-net data creation from service
Posted by bfields on Thu, 03 May 2012 14:27:35 GMT
View Forum Message <> Reply to Message

On Wed, May 02, 2012 at 05:58:37PM -0400, J. Bruce Fields wrote:
> Thanks, applying.

Whoops, no--I'm hitting the BUG at net/sunrpc/svc.c:558 (the first of
the two BUG_ON()s in svc_destroy()) when restarting nfsd. Could you
look into this?

--b.

>
> --b.
>
> On Wed, May 02, 2012 at 04:08:29PM +0400, Stanislav Kinsbursky wrote:
> > creation
> >
> > v3: "SUNRPC: new svc_bind() routine introduced" patch was squashed with the
> > "SUNRPC: check rpcbind clients usage counter before decrement" patch.
> >
> > v2: Increase per-net usage counted in lockd_up_net().
> >
> > This is a cleanup patch set.
> > It will be followed my LockD start/stop cleanup patch set and NFS callback

> > service containerization patch set (yes, I forgot to implement it).
> >
> > Today per-net data is created with service, and then is service is starting in
> > other network namespace. And thus it's destroyed with service too. Moreover,
> > network context for destroying of per-net data is taken from current process.
> > This is correct, but code looks ugly.
> > This patch set separates per-net data allocation from service allocation and
> > destruction.
> > IOW, per-net data have to be destroyed by service users - not service itself.
> >
> > BTW, NFSd code become uglier with this patch set. Sorry.
> > But I assume, that these new ugly parts will be replaced later by NFSd service
> > containerization code.
> >
> > The following series implements...
> >
> > ---
> >
> > Stanislav Kinsbursky (2):
> > SUNRPC: new svc_bind() routine introduced
> > SUNRPC: move per-net operations from svc_destroy()
> >
> >
> > fs/lockd/svc.c | 33 ++++++++++++++++++++++----------
> > fs/nfs/callback.c | 11 +++++++++++
> > fs/nfsd/nfsctl.c | 4 ++++
> > fs/nfsd/nfssvc.c | 16 ++++++++++++++++
> > include/linux/sunrpc/svc.h | 1 +
> > net/sunrpc/rpcb_clnt.c | 12 +++++++-----
> > net/sunrpc/svc.c | 23 ++++++++++-------------
> > 7 files changed, 70 insertions(+), 30 deletions(-)
> >

Subject: Re: [PATCH v3 0/2] SUNRPC: separate per-net data creation from service
Posted by Stanislav Kinsbursky on Fri, 04 May 2012 08:43:06 GMT
View Forum Message <> Reply to Message

On 03.05.2012 18:27, J. Bruce Fields wrote:
> On Wed, May 02, 2012 at 05:58:37PM -0400, J. Bruce Fields wrote:
>> Thanks, applying.
>
> Whoops, no--I'm hitting the BUG at net/sunrpc/svc.c:558 (the first of
> the two BUG_ON()s in svc_destroy()) when restarting nfsd. Could you
> look into this?
>

My fault. Sorry. Investigating.

> --b.
>
>>
>> --b.
>>
>> On Wed, May 02, 2012 at 04:08:29PM +0400, Stanislav Kinsbursky wrote:
>>> creation
>>>
>>> v3: "SUNRPC: new svc_bind() routine introduced" patch was squashed with the
>>> "SUNRPC: check rpcbind clients usage counter before decrement" patch.
>>>
>>> v2: Increase per-net usage counted in lockd_up_net().
>>>
>>> This is a cleanup patch set.
>>> It will be followed my LockD start/stop cleanup patch set and NFS callback
>>> service containerization patch set (yes, I forgot to implement it).
>>>
>>> Today per-net data is created with service, and then is service is starting in
>>> other network namespace. And thus it's destroyed with service too. Moreover,
>>> network context for destroying of per-net data is taken from current process.
>>> This is correct, but code looks ugly.
>>> This patch set separates per-net data allocation from service allocation and
>>> destruction.
>>> IOW, per-net data have to be destroyed by service users - not service itself.
>>>
>>> BTW, NFSd code become uglier with this patch set. Sorry.
>>> But I assume, that these new ugly parts will be replaced later by NFSd service
>>> containerization code.
>>>
>>> The following series implements...
>>>
>>> ---
>>>
>>> Stanislav Kinsbursky (2):
>>> SUNRPC: new svc_bind() routine introduced
>>> SUNRPC: move per-net operations from svc_destroy()
>>>
>>>
>>> fs/lockd/svc.c | 33 ++++++++++++++++++++++------------
>>> fs/nfs/callback.c | 11 +++++++++++
>>> fs/nfsd/nfsctl.c | 4 ++++
>>> fs/nfsd/nfssvc.c | 16 ++++++++++++++++
>>> include/linux/sunrpc/svc.h | 1 +
>>> net/sunrpc/rpcb_clnt.c | 12 +++++++-----
>>> net/sunrpc/svc.c | 23 ++++++++++-------------
>>> 7 files changed, 70 insertions(+), 30 deletions(-)
>>>

--
Best regards,
Stanislav Kinsbursky

Subject: [PATCH v4] SUNRPC: move per-net operations from svc_destroy()
Posted by Stanislav Kinsbursky on Fri, 04 May 2012 08:49:41 GMT
View Forum Message <> Reply to Message

v4:
1) Added per-net data shutdown by the last kernel thread in nfsd().

v2:
1) Increase per-net usage counted in lockd_up_net(), because of note 2.

The idea is to separate service destruction and per-net operations, because
these are two different things and it's mix looks ugly.

Notes:
1) For NFS server this patch looks ugly (sorry for that). But these place will be
rewritten soon during NFSd containerization.
2) LockD per-net counter increase int lockd_up() was moved prior to
make_socks() to make lockd_down_net() call safe in case of error.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
 fs/lockd/svc.c | 27 +++++++++++++++------------
 fs/nfs/callback.c | 3 +++
 fs/nfsd/nfsctl.c | 12 +++++++++---
 fs/nfsd/nfssvc.c | 14 ++++++++++++++
 net/sunrpc/svc.c | 4 ----
 5 files changed, 41 insertions(+), 19 deletions(-)

diff --git a/fs/lockd/svc.c b/fs/lockd/svc.c
index b7e92ed..3250f28 100644
--- a/fs/lockd/svc.c
+++ b/fs/lockd/svc.c
@@ -257,7 +257,7 @@ static int lockd_up_net(struct net *net)
  struct svc_serv *serv = nlmsvc_rqst->rq_server;
  int error;

- if (ln->nlmsvc_users)
+ if (ln->nlmsvc_users++)
   return 0;

  error = svc_rpcb_setup(serv, net);

```
@@ -272,6 +272,7 @@ static int lockd_up_net(struct net *net)
 err_socks:
  svc_rpcb_cleanup(serv, net);
 err_rpcb:
+ ln->nlmsvc_users--;
  return error;
 }

@@ -299,6 +300,7 @@ int lockd_up(struct net *net)
 {
  struct svc_serv *serv;
  int  error = 0;
+ struct lockd_net *ln = net_generic(net, lockd_net_id);

  mutex_lock(&nlmsvc_mutex);
  /*
@@ -330,9 +332,11 @@ int lockd_up(struct net *net)
   goto destroy_and_out;
  }

+ ln->nlmsvc_users++;
+
  error = make_socks(serv, net);
  if (error < 0)
-  goto destroy_and_out;
+  goto err_start;

  /*
   * Create the kernel thread and wait for it to start.
@@ -344,7 +348,7 @@ int lockd_up(struct net *net)
    printk(KERN_WARNING
     "lockd_up: svc_rqst allocation failed, error=%d\n",
     error);
-  goto destroy_and_out;
+  goto err_start;
  }

  svc_sock_update_bufs(serv);
@@ -358,7 +362,7 @@ int lockd_up(struct net *net)
   nlmsvc_rqst = NULL;
   printk(KERN_WARNING
    "lockd_up: kthread_run failed, error=%d\n", error);
-  goto destroy_and_out;
+  goto err_start;
  }

  /*
@@ -368,14 +372,14 @@ int lockd_up(struct net *net)
```

```
 destroy_and_out:
  svc_destroy(serv);
 out:
- if (!error) {
-  struct lockd_net *ln = net_generic(net, lockd_net_id);
-
-  ln->nlmsvc_users++;
+ if (!error)
    nlmsvc_users++;
- }
  mutex_unlock(&nlmsvc_mutex);
  return error;
+
+err_start:
+ lockd_down_net(net);
+ goto destroy_and_out;
 }
 EXPORT_SYMBOL_GPL(lockd_up);

@@ -386,11 +390,10 @@ void
 lockd_down(struct net *net)
 {
  mutex_lock(&nlmsvc_mutex);
+ lockd_down_net(net);
  if (nlmsvc_users) {
-  if (--nlmsvc_users) {
-   lockd_down_net(net);
+  if (--nlmsvc_users)
    goto out;
-  }
  } else {
   printk(KERN_ERR "lockd_down: no users! task=%p\n",
    nlmsvc_task);
diff --git a/fs/nfs/callback.c b/fs/nfs/callback.c
index 26b38fb..cff3940 100644
--- a/fs/nfs/callback.c
+++ b/fs/nfs/callback.c
@@ -314,6 +314,8 @@ out_err:
  dprintk("NFS: Couldn't create callback socket or server thread; "
   "err = %d\n", ret);
  cb_info->users--;
+ if (serv)
+  svc_shutdown_net(serv, net);
  goto out;
 }

@@ -328,6 +330,7 @@ void nfs_callback_down(int minorversion)
  cb_info->users--;
```

```
  if (cb_info->users == 0 && cb_info->task != NULL) {
   kthread_stop(cb_info->task);
+  svc_shutdown_net(cb_info->serv, current->nsproxy->net_ns);
   svc_exit_thread(cb_info->rqst);
   cb_info->serv = NULL;
   cb_info->rqst = NULL;
diff --git a/fs/nfsd/nfsctl.c b/fs/nfsd/nfsctl.c
index 7269988..c55298e 100644
--- a/fs/nfsd/nfsctl.c
+++ b/fs/nfsd/nfsctl.c
@@ -661,6 +661,7 @@ static ssize_t __write_ports_addfd(char *buf)
 {
  char *mesg = buf;
  int fd, err;
+ struct net *net = &init_net;

  err = get_int(&mesg, &fd);
  if (err != 0 || fd < 0)
@@ -672,6 +673,8 @@ static ssize_t __write_ports_addfd(char *buf)

  err = svc_addsock(nfsd_serv, fd, buf, SIMPLE_TRANSACTION_LIMIT);
  if (err < 0) {
+  if (nfsd_serv->sv_nrthreads == 1)
+   svc_shutdown_net(nfsd_serv, net);
   svc_destroy(nfsd_serv);
   return err;
  }
@@ -709,6 +712,7 @@ static ssize_t __write_ports_addxprt(char *buf)
  char transport[16];
  struct svc_xprt *xprt;
  int port, err;
+ struct net *net = &init_net;

  if (sscanf(buf, "%15s %4u", transport, &port) != 2)
   return -EINVAL;
@@ -720,12 +724,12 @@ static ssize_t __write_ports_addxprt(char *buf)
  if (err != 0)
   return err;

- err = svc_create_xprt(nfsd_serv, transport, &init_net,
+ err = svc_create_xprt(nfsd_serv, transport, net,
     PF_INET, port, SVC_SOCK_ANONYMOUS);
  if (err < 0)
   goto out_err;

- err = svc_create_xprt(nfsd_serv, transport, &init_net,
+ err = svc_create_xprt(nfsd_serv, transport, net,
     PF_INET6, port, SVC_SOCK_ANONYMOUS);
```

```
  if (err < 0 && err != -EAFNOSUPPORT)
   goto out_close;
@@ -734,12 +738,14 @@ static ssize_t __write_ports_addxprt(char *buf)
  nfsd_serv->sv_nrthreads--;
  return 0;
 out_close:
- xprt = svc_find_xprt(nfsd_serv, transport, &init_net, PF_INET, port);
+ xprt = svc_find_xprt(nfsd_serv, transport, net, PF_INET, port);
  if (xprt != NULL) {
   svc_close_xprt(xprt);
   svc_xprt_put(xprt);
  }
 out_err:
+ if (nfsd_serv->sv_nrthreads == 1)
+  svc_shutdown_net(nfsd_serv, net);
  svc_destroy(nfsd_serv);
  return err;
 }
diff --git a/fs/nfsd/nfssvc.c b/fs/nfsd/nfssvc.c
index 0762f3c..ee709fc 100644
--- a/fs/nfsd/nfssvc.c
+++ b/fs/nfsd/nfssvc.c
@@ -382,6 +382,7 @@ int nfsd_set_nrthreads(int n, int *nthreads)
  int i = 0;
  int tot = 0;
  int err = 0;
+ struct net *net = &init_net;

  WARN_ON(!mutex_is_locked(&nfsd_mutex));

@@ -426,6 +427,9 @@ int nfsd_set_nrthreads(int n, int *nthreads)
   if (err)
    break;
  }
+
+ if (nfsd_serv->sv_nrthreads == 1)
+  svc_shutdown_net(nfsd_serv, net);
  svc_destroy(nfsd_serv);

  return err;
@@ -441,6 +445,7 @@ nfsd_svc(unsigned short port, int nrservs)
 {
  int error;
  bool nfsd_up_before;
+ struct net *net = &init_net;

  mutex_lock(&nfsd_mutex);
  dprintk("nfsd: creating service\n");
```

```
@@ -473,6 +478,8 @@ out_shutdown:
  if (error < 0 && !nfsd_up_before)
   nfsd_shutdown();
 out_destroy:
+ if (nfsd_serv->sv_nrthreads == 1)
+  svc_shutdown_net(nfsd_serv, net);
  svc_destroy(nfsd_serv);  /* Release server */
 out:
  mutex_unlock(&nfsd_mutex);
@@ -556,6 +563,9 @@ nfsd(void *vrqstp)
  nfsdstats.th_cnt --;

 out:
+ if (rqstp->rq_server->sv_nrthreads == 1)
+  svc_shutdown_net(rqstp->rq_server, &init_net);
+
  /* Release the thread */
  svc_exit_thread(rqstp);

@@ -668,8 +678,12 @@ int nfsd_pool_stats_open(struct inode *inode, struct file *file)
 int nfsd_pool_stats_release(struct inode *inode, struct file *file)
 {
  int ret = seq_release(inode, file);
+ struct net *net = &init_net;
+
  mutex_lock(&nfsd_mutex);
  /* this function really, really should have been called svc_put() */
+ if (nfsd_serv->sv_nrthreads == 1)
+  svc_shutdown_net(nfsd_serv, net);
  svc_destroy(nfsd_serv);
  mutex_unlock(&nfsd_mutex);
  return ret;
diff --git a/net/sunrpc/svc.c b/net/sunrpc/svc.c
index e6d542c..b7210f5 100644
--- a/net/sunrpc/svc.c
+++ b/net/sunrpc/svc.c
@@ -537,8 +537,6 @@ EXPORT_SYMBOL_GPL(svc_shutdown_net);
 void
 svc_destroy(struct svc_serv *serv)
 {
- struct net *net = current->nsproxy->net_ns;
-
  dprintk("svc: svc_destroy(%s, %d)\n",
     serv->sv_program->pg_name,
     serv->sv_nrthreads);
@@ -553,8 +551,6 @@ svc_destroy(struct svc_serv *serv)

  del_timer_sync(&serv->sv_temptimer);
```

```
- svc_shutdown_net(serv, net);
-
  /*
   * The last user is gone and thus all sockets have to be destroyed to
   * the point. Check this.
```