
Subject: [PATCH RFC 12/13] LockD: pass actual network namespace to grace period management functions

Posted by [Stanislav Kinsbursky](#) on Sat, 05 May 2012 17:22:58 GMT

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

Passed network namespace replaced hard-coded init_net

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>

```
---
fs/lockd/grace.c      |  6 ++----
fs/lockd/svc.c        | 16 ++++++++-----
fs/lockd/svc4proc.c   | 13 ++++++-----
fs/lockd/svcclock.c   | 16 ++++++++-----
fs/lockd/svcproc.c    | 15 ++++++++-----
fs/nfsd/nfs4proc.c    | 18 ++++++++-----
fs/nfsd/nfs4state.c   | 30 ++++++++-----
fs/nfsd/state.h       |  3 +-
include/linux/fs.h    |  5 +++--
include/linux/lockd/lockd.h |  4 ++--
10 files changed, 68 insertions(+), 58 deletions(-)
```

```
diff --git a/fs/lockd/grace.c b/fs/lockd/grace.c
```

```
index 8dbaff7..6d1ee72 100644
```

```
--- a/fs/lockd/grace.c
```

```
+++ b/fs/lockd/grace.c
```

```
@@ -21,9 +21,8 @@ static DEFINE_SPINLOCK(grace_lock);
```

```
*
```

```
* This function is called to start a grace period.
```

```
*/
```

```
-void locks_start_grace(struct lock_manager *lm)
```

```
+void locks_start_grace(struct net *net, struct lock_manager *lm)
```

```
{
```

```
- struct net *net = &init_net;
```

```
struct lockd_net *ln = net_generic(net, lockd_net_id);
```

```
spin_lock(&grace_lock);
```

```
@@ -57,9 +56,8 @@ EXPORT_SYMBOL_GPL(locks_end_grace);
```

```
* to answer ordinary lock requests, and when they should accept only
```

```
* lock reclaims.
```

```
*/
```

```
-int locks_in_grace(void)
```

```
+int locks_in_grace(struct net *net)
```

```
{
```

```
- struct net *net = &init_net;
```

```
struct lockd_net *ln = net_generic(net, lockd_net_id);
```

```
return !list_empty(&ln->grace_list);
```

```
diff --git a/fs/lockd/svc.c b/fs/lockd/svc.c
```

index 834dfe2..68271c2 100644

--- a/fs/lockd/svc.c

+++ b/fs/lockd/svc.c

```
@@ -97,12 +97,12 @@ static void grace_ender(struct work_struct *grace)
    locks_end_grace(&ln->lockd_manager);
}
```

```
-static void set_grace_period(void)
```

```
+static void set_grace_period(struct net *net)
```

```
{
    unsigned long grace_period = get_lockd_grace_period();
- struct lockd_net *ln = net_generic(&init_net, lockd_net_id);
+ struct lockd_net *ln = net_generic(net, lockd_net_id);
```

```
- locks_start_grace(&ln->lockd_manager);
+ locks_start_grace(net, &ln->lockd_manager);
    cancel_delayed_work_sync(&ln->grace_period_end);
    schedule_delayed_work(&ln->grace_period_end, grace_period);
}
```

```
@@ -110,12 +110,13 @@ static void set_grace_period(void)
static void restart_grace(void)
```

```
{
    if (nlmsvc_ops) {
- struct lockd_net *ln = net_generic(&init_net, lockd_net_id);
+ struct net *net = &init_net;
+ struct lockd_net *ln = net_generic(net, lockd_net_id);
```

```
    cancel_delayed_work_sync(&ln->grace_period_end);
    locks_end_grace(&ln->lockd_manager);
    nlmsvc_invalidate_all();
- set_grace_period();
+ set_grace_period(net);
    }
}
```

```
@@ -127,7 +128,8 @@ lockd(void *vrqstp)
```

```
{
    int err = 0, preverr = 0;
    struct svc_rqst *rqstp = vrqstp;
- struct lockd_net *ln = net_generic(&init_net, lockd_net_id);
+ struct net *net = &init_net;
+ struct lockd_net *ln = net_generic(net, lockd_net_id);
```

```
    /* try_to_freeze() is called from svc_rcv() */
    set_freezable();
```

```
@@ -141,7 +143,7 @@ lockd(void *vrqstp)
```

```
    nlm_timeout = LOCKD_DFLT_TIMEO;
    nlmsvc_timeout = nlm_timeout * HZ;
```

```

- set_grace_period();
+ set_grace_period(net);

/*
 * The main request loop. We don't terminate until the last
diff --git a/fs/lockd/svc4proc.c b/fs/lockd/svc4proc.c
index 9a41fdc..4a43d25 100644
--- a/fs/lockd/svc4proc.c
+++ b/fs/lockd/svc4proc.c
@@ -11,6 +11,7 @@
#include <linux/time.h>
#include <linux/lockd/lockd.h>
#include <linux/lockd/share.h>
+#include <linux/sunrpc/svc_xprt.h>

#define NLMDBG_FACILITY NLMDBG_CLIENT

@@ -151,7 +152,7 @@ nlm4svc_proc_cancel(struct svc_rqst *rqstp, struct nlm_args *argp,
    resp->cookie = argp->cookie;

/* Don't accept requests during grace period */
- if (locks_in_grace()) {
+ if (locks_in_grace(SVC_NET(rqstp))) {
    resp->status = nlm_lck_denied_grace_period;
    return rpc_success;
}
@@ -161,7 +162,7 @@ nlm4svc_proc_cancel(struct svc_rqst *rqstp, struct nlm_args *argp,
    return resp->status == nlm_drop_reply ? rpc_drop_reply : rpc_success;

/* Try to cancel request. */
- resp->status = nlmsvc_cancel_blocked(file, &argp->lock);
+ resp->status = nlmsvc_cancel_blocked(SVC_NET(rqstp), file, &argp->lock);

dprintk("lockd: CANCEL      status %d\n", ntohl(resp->status));
nlmsvc_release_host(host);
@@ -184,7 +185,7 @@ nlm4svc_proc_unlock(struct svc_rqst *rqstp, struct nlm_args *argp,
    resp->cookie = argp->cookie;

/* Don't accept new lock requests during grace period */
- if (locks_in_grace()) {
+ if (locks_in_grace(SVC_NET(rqstp))) {
    resp->status = nlm_lck_denied_grace_period;
    return rpc_success;
}
@@ -194,7 +195,7 @@ nlm4svc_proc_unlock(struct svc_rqst *rqstp, struct nlm_args *argp,
    return resp->status == nlm_drop_reply ? rpc_drop_reply : rpc_success;

```

```

/* Now try to remove the lock */
- resp->status = nlmsvc_unlock(file, &argp->lock);
+ resp->status = nlmsvc_unlock(SVC_NET(rqstp), file, &argp->lock);

dprintk("lockd: UNLOCK      status %d\n", ntohl(resp->status));
nlmsvc_release_host(host);
@@ -321,7 +322,7 @@ nlm4svc_proc_share(struct svc_rqst *rqstp, struct nlm_args *argp,
    resp->cookie = argp->cookie;

/* Don't accept new lock requests during grace period */
- if (locks_in_grace() && !argp->reclaim) {
+ if (locks_in_grace(SVC_NET(rqstp)) && !argp->reclaim) {
    resp->status = nlm_lck_denied_grace_period;
    return rpc_success;
}
@@ -354,7 +355,7 @@ nlm4svc_proc_unshare(struct svc_rqst *rqstp, struct nlm_args *argp,
    resp->cookie = argp->cookie;

/* Don't accept requests during grace period */
- if (locks_in_grace()) {
+ if (locks_in_grace(SVC_NET(rqstp))) {
    resp->status = nlm_lck_denied_grace_period;
    return rpc_success;
}
diff --git a/fs/lockd/svcllock.c b/fs/lockd/svcllock.c
index e46353f..afe4488 100644
--- a/fs/lockd/svcllock.c
+++ b/fs/lockd/svcllock.c
@@ -26,7 +26,7 @@
#include <linux/kernel.h>
#include <linux/sched.h>
#include <linux/sunrpc/clnt.h>
-#include <linux/sunrpc/svc.h>
+#include <linux/sunrpc/svc_xprt.h>
#include <linux/lockd/nlm.h>
#include <linux/lockd/lockd.h>
#include <linux/kthread.h>
@@ -447,11 +447,11 @@ nlm4svc_lock(struct svc_rqst *rqstp, struct nlm_file *file,
    goto out;
}

- if (locks_in_grace() && !reclaim) {
+ if (locks_in_grace(SVC_NET(rqstp)) && !reclaim) {
    ret = nlm_lck_denied_grace_period;
    goto out;
}
- if (reclaim && !locks_in_grace()) {
+ if (reclaim && !locks_in_grace(SVC_NET(rqstp))) {

```

```

    ret = nlm_lck_denied_grace_period;
    goto out;
}
@@ -559,7 +559,7 @@ nlmsvc_testlock(struct svc_rqst *rqstp, struct nlm_file *file,
    goto out;
}

- if (locks_in_grace()) {
+ if (locks_in_grace(SVC_NET(rqstp))) {
    ret = nlm_lck_denied_grace_period;
    goto out;
}
@@ -603,7 +603,7 @@ out:
    * must be removed.
    */
    __be32
-nlmsvc_unlock(struct nlm_file *file, struct nlm_lock *lock)
+nlmsvc_unlock(struct net *net, struct nlm_file *file, struct nlm_lock *lock)
{
    int error;

@@ -615,7 +615,7 @@ nlmsvc_unlock(struct nlm_file *file, struct nlm_lock *lock)
    (long long)lock->fl.fl_end);

    /* First, cancel any lock that might be there */
- nlmsvc_cancel_blocked(file, lock);
+ nlmsvc_cancel_blocked(net, file, lock);

    lock->fl.fl_type = F_UNLCK;
    error = vfs_lock_file(file->f_file, F_SETLK, &lock->fl, NULL);
@@ -631,7 +631,7 @@ nlmsvc_unlock(struct nlm_file *file, struct nlm_lock *lock)
    * The calling procedure must check whether the file can be closed.
    */
    __be32
-nlmsvc_cancel_blocked(struct nlm_file *file, struct nlm_lock *lock)
+nlmsvc_cancel_blocked(struct net *net, struct nlm_file *file, struct nlm_lock *lock)
{
    struct nlm_block *block;
    int status = 0;
@@ -643,7 +643,7 @@ nlmsvc_cancel_blocked(struct nlm_file *file, struct nlm_lock *lock)
    (long long)lock->fl.fl_start,
    (long long)lock->fl.fl_end);

- if (locks_in_grace())
+ if (locks_in_grace(net))
    return nlm_lck_denied_grace_period;

    mutex_lock(&file->f_mutex);

```

```

diff --git a/fs/lockd/svcproc.c b/fs/lockd/svcproc.c
index d27aab1..de8f2ca 100644
--- a/fs/lockd/svcproc.c
+++ b/fs/lockd/svcproc.c
@@ -11,6 +11,7 @@
#include <linux/time.h>
#include <linux/lockd/lockd.h>
#include <linux/lockd/share.h>
+#include <linux/sunrpc/svc_xprt.h>

#define NLMDBG_FACILITY NLMDBG_CLIENT

@@ -175,13 +176,14 @@ nlmsvc_proc_cancel(struct svc_rqst *rqstp, struct nlm_args *argp,
{
    struct nlm_host *host;
    struct nlm_file *file;
+ struct net *net = SVC_NET(rqstp);

    dprintk("lockd: CANCEL      called\n");

    resp->cookie = argp->cookie;

    /* Don't accept requests during grace period */
- if (locks_in_grace()) {
+ if (locks_in_grace(net)) {
    resp->status = nlm_lck_denied_grace_period;
    return rpc_success;
}
@@ -191,7 +193,7 @@ nlmsvc_proc_cancel(struct svc_rqst *rqstp, struct nlm_args *argp,
    return resp->status == nlm_drop_reply ? rpc_drop_reply : rpc_success;

    /* Try to cancel request. */
- resp->status = cast_status(nlmsvc_cancel_blocked(file, &argp->lock));
+ resp->status = cast_status(nlmsvc_cancel_blocked(net, file, &argp->lock));

    dprintk("lockd: CANCEL      status %d\n", ntohl(resp->status));
    nlmsvc_release_host(host);
@@ -208,13 +210,14 @@ nlmsvc_proc_unlock(struct svc_rqst *rqstp, struct nlm_args *argp,
{
    struct nlm_host *host;
    struct nlm_file *file;
+ struct net *net = SVC_NET(rqstp);

    dprintk("lockd: UNLOCK      called\n");

    resp->cookie = argp->cookie;

    /* Don't accept new lock requests during grace period */

```

```

- if (locks_in_grace()) {
+ if (locks_in_grace(net)) {
    resp->status = nlm_lck_denied_grace_period;
    return rpc_success;
}
@@ -224,7 +227,7 @@ nlmsvc_proc_unlock(struct svc_rqst *rqstp, struct nlm_args *argp,
    return resp->status == nlm_drop_reply ? rpc_drop_reply : rpc_success;

    /* Now try to remove the lock */
- resp->status = cast_status(nlmsvc_unlock(file, &argp->lock));
+ resp->status = cast_status(nlmsvc_unlock(net, file, &argp->lock));

    dprintk("lockd: UNLOCK      status %d\n", ntohl(resp->status));
    nlmsvc_release_host(host);
@@ -361,7 +364,7 @@ nlmsvc_proc_share(struct svc_rqst *rqstp, struct nlm_args *argp,
    resp->cookie = argp->cookie;

    /* Don't accept new lock requests during grace period */
- if (locks_in_grace() && !argp->reclaim) {
+ if (locks_in_grace(SVC_NET(rqstp)) && !argp->reclaim) {
    resp->status = nlm_lck_denied_grace_period;
    return rpc_success;
}
@@ -394,7 +397,7 @@ nlmsvc_proc_unshare(struct svc_rqst *rqstp, struct nlm_args *argp,
    resp->cookie = argp->cookie;

    /* Don't accept requests during grace period */
- if (locks_in_grace()) {
+ if (locks_in_grace(SVC_NET(rqstp))) {
    resp->status = nlm_lck_denied_grace_period;
    return rpc_success;
}
diff --git a/fs/nfsd/nfs4proc.c b/fs/nfsd/nfs4proc.c
index 087065b..7353e1f 100644
--- a/fs/nfsd/nfs4proc.c
+++ b/fs/nfsd/nfs4proc.c
@@ -352,10 +352,10 @@ nfsd4_open(struct svc_rqst *rqstp, struct nfsd4_compound_state
*cstate,
    /* Openowner is now set, so sequence id will get bumped. Now we need
    * these checks before we do any creates: */
    status = nfserr_grace;
- if (locks_in_grace() && open->op_claim_type != NFS4_OPEN_CLAIM_PREVIOUS)
+ if (locks_in_grace(SVC_NET(rqstp)) && open->op_claim_type !=
NFS4_OPEN_CLAIM_PREVIOUS)
    goto out;
    status = nfserr_no_grace;
- if (!locks_in_grace() && open->op_claim_type == NFS4_OPEN_CLAIM_PREVIOUS)
+ if (!locks_in_grace(SVC_NET(rqstp)) && open->op_claim_type ==

```

```

NFS4_OPEN_CLAIM_PREVIOUS)
    goto out;

    switch (open->op_claim_type) {
@@ -690,7 +690,8 @@ nfsd4_read(struct svc_rqst *rqstp, struct nfsd4_compound_state *cstate,

    nfs4_lock_state();
    /* check stateid */
- if ((status = nfs4_preprocess_stateid_op(cstate, &read->rd_stateid,
+ if ((status = nfs4_preprocess_stateid_op(SVC_NET(rqstp),
+     cstate, &read->rd_stateid,
+     RD_STATE, &read->rd_filp))) {
    dprintk("NFSD: nfsd4_read: couldn't process stateid!\n");
    goto out;
@@ -745,7 +746,7 @@ nfsd4_remove(struct svc_rqst *rqstp, struct nfsd4_compound_state
*cstate,
{
    __be32 status;

- if (locks_in_grace())
+ if (locks_in_grace(SVC_NET(rqstp)))
    return nfserr_grace;
    status = nfsd_unlink(rqstp, &cstate->current_fh, 0,
        remove->rm_name, remove->rm_namelen);
@@ -764,8 +765,8 @@ nfsd4_rename(struct svc_rqst *rqstp, struct nfsd4_compound_state
*cstate,

    if (!cstate->save_fh.fh_dentry)
        return status;
- if (locks_in_grace() && !(cstate->save_fh.fh_export->ex_flags
-     & NFSEXP_NOSUBTREECHECK))
+ if (locks_in_grace(SVC_NET(rqstp)) &&
+     !(cstate->save_fh.fh_export->ex_flags & NFSEXP_NOSUBTREECHECK))
    return nfserr_grace;
    status = nfsd_rename(rqstp, &cstate->save_fh, rename->rn_sname,
        rename->rn_snamelen, &cstate->current_fh,
@@ -848,7 +849,7 @@ nfsd4_setattr(struct svc_rqst *rqstp, struct nfsd4_compound_state
*cstate,

    if (setattr->sa_iattr.ia_valid & ATTR_SIZE) {
        nfs4_lock_state();
-     status = nfs4_preprocess_stateid_op(cstate,
+     status = nfs4_preprocess_stateid_op(SVC_NET(rqstp), cstate,
+     &setattr->sa_stateid, WR_STATE, NULL);
        nfs4_unlock_state();
        if (status) {
@@ -893,7 +894,8 @@ nfsd4_write(struct svc_rqst *rqstp, struct nfsd4_compound_state *cstate,
    return nfserr_inval;

```

```

    nfs4_lock_state();
- status = nfs4_preprocess_stateid_op(cstate, stateid, WR_STATE, &filp);
+ status = nfs4_preprocess_stateid_op(SVC_NET(rqstp),
+   cstate, stateid, WR_STATE, &filp);
    if (filp)
        get_file(filp);
    nfs4_unlock_state();
diff --git a/fs/nfsd/nfs4state.c b/fs/nfsd/nfs4state.c
index c6c1be9..2f940c1 100644
--- a/fs/nfsd/nfs4state.c
+++ b/fs/nfsd/nfs4state.c
@@ -2951,7 +2951,9 @@ static void nfsd4_open_deleg_none_ext(struct nfsd4_open *open, int
status)
    * Attempt to hand out a delegation.
    */
    static void
-nfs4_open_delegation(struct svc_fh *fh, struct nfsd4_open *open, struct nfs4_ol_stateid *stp,
struct svc_fh *parent)
+nfs4_open_delegation(struct net *net, struct svc_fh *fh,
+   struct nfsd4_open *open, struct nfs4_ol_stateid *stp,
+   struct svc_fh *parent)
    {
        struct nfs4_delegation *dp;
        struct nfs4_openowner *oo = container_of(stp->st_stateowner, struct nfs4_openowner,
oo_owner);
@@ -2972,7 +2974,7 @@ nfs4_open_delegation(struct svc_fh *fh, struct nfsd4_open *open,
struct nfs4_ol_
    case NFS4_OPEN_CLAIM_NULL:
        /* Let's not give out any delegations till everyone's
        * had the chance to reclaim theirs.... */
-   if (locks_in_grace())
+   if (locks_in_grace(net))
        goto out;
        if (!cb_up || !(oo->oo_flags & NFS4_OO_CONFIRMED))
            goto out;
@@ -3108,7 +3110,7 @@ nfsd4_process_open2(struct svc_rqst *rqstp, struct svc_fh *current_fh,
struct nf
    * Attempt to hand out a delegation. No error return, because the
    * OPEN succeeds even if we fail.
    */
-   nfs4_open_delegation(current_fh, open, stp, parent);
+   nfs4_open_delegation(SVC_NET(rqstp), current_fh, open, stp, parent);
    nodeleg:
        status = nfs_ok;

@@ -3347,11 +3349,11 @@ out:
    }

```

```

static inline __be32
-check_special_stateids(svc_fh *current_fh, stateid_t *stateid, int flags)
+check_special_stateids(struct net *net, svc_fh *current_fh, stateid_t *stateid, int flags)
{
    if (ONE_STATEID(stateid) && (flags & RD_STATE))
        return nfs_ok;
- else if (locks_in_grace()) {
+ else if (locks_in_grace(net)) {
    /* Answer in remaining cases depends on existence of
     * conflicting state; so we must wait out the grace period. */
    return nfserr_grace;
@@ -3368,9 +3370,9 @@ check_special_stateids(svc_fh *current_fh, stateid_t *stateid, int flags)
    * that are not able to provide mandatory locking.
    */
    static inline int
-grace_disallows_io(struct inode *inode)
+grace_disallows_io(struct net *net, struct inode *inode)
{
- return locks_in_grace() && mandatory_lock(inode);
+ return locks_in_grace(net) && mandatory_lock(inode);
}

/* Returns true iff a is later than b: */
@@ -3453,7 +3455,7 @@ static __be32 nfsd4_lookup_stateid(stateid_t *stateid, unsigned char
typemask, s
    * Checks for stateid operations
    */
    __be32
-nfs4_preprocess_stateid_op(struct nfsd4_compound_state *cstate,
+nfs4_preprocess_stateid_op(struct net *net, struct nfsd4_compound_state *cstate,
    stateid_t *stateid, int flags, struct file **filpp)
{
    struct nfs4_stid *s;
@@ -3466,11 +3468,11 @@ nfs4_preprocess_stateid_op(struct nfsd4_compound_state *cstate,
    if (filpp)
        *filpp = NULL;

- if (grace_disallows_io(ino))
+ if (grace_disallows_io(net, ino))
    return nfserr_grace;

    if (ZERO_STATEID(stateid) || ONE_STATEID(stateid))
- return check_special_stateids(current_fh, stateid, flags);
+ return check_special_stateids(net, current_fh, stateid, flags);

    status = nfsd4_lookup_stateid(stateid,
NFS4_DELEG_STID|NFS4_OPEN_STID|NFS4_LOCK_STID, &s);

```

```

    if (status)
@@ -4168,10 +4170,10 @@ nfsd4_lock(struct svc_rqst *rqstp, struct nfsd4_compound_state
*cstate,
    goto out;

    status = nfserr_grace;
- if (locks_in_grace() && !lock->lk_reclaim)
+ if (locks_in_grace(SVC_NET(rqstp)) && !lock->lk_reclaim)
    goto out;
    status = nfserr_no_grace;
- if (!locks_in_grace() && lock->lk_reclaim)
+ if (!locks_in_grace(SVC_NET(rqstp)) && lock->lk_reclaim)
    goto out;

    locks_init_lock(&file_lock);
@@ -4274,7 +4276,7 @@ nfsd4_lockt(struct svc_rqst *rqstp, struct nfsd4_compound_state
*cstate,
    int error;
    __be32 status;

- if (locks_in_grace())
+ if (locks_in_grace(SVC_NET(rqstp)))
    return nfserr_grace;

    if (check_lock_length(lockt->lt_offset, lockt->lt_length))
@@ -4758,7 +4760,7 @@ nfs4_state_start(void)
    get_net(net);
    nfsd4_client_tracking_init(net);
    boot_time = get_seconds();
- locks_start_grace(&nn->nfsd4_manager);
+ locks_start_grace(net, &nn->nfsd4_manager);
    grace_ended = false;
    printk(KERN_INFO "NFSD: starting %ld-second grace period\n",
        nfsd4_grace);
diff --git a/fs/nfsd/state.h b/fs/nfsd/state.h
index 89ab137..29d9f0b 100644
--- a/fs/nfsd/state.h
+++ b/fs/nfsd/state.h
@@ -452,7 +452,8 @@ static inline struct nfs4_ol_stateid *openlockstateid(struct nfs4_stid *s)

struct nfsd4_compound_state;

-extern __be32 nfs4_preprocess_stateid_op(struct nfsd4_compound_state *cstate,
+extern __be32 nfs4_preprocess_stateid_op(struct net *net,
+ struct nfsd4_compound_state *cstate,
    stateid_t *stateid, int flags, struct file **filp);
extern void nfs4_lock_state(void);
extern void nfs4_unlock_state(void);

```

```

diff --git a/include/linux/fs.h b/include/linux/fs.h
index 99cbb1f..02a3eb2 100644
--- a/include/linux/fs.h
+++ b/include/linux/fs.h
@@ -1129,9 +1129,10 @@ struct lock_manager {
    struct list_head list;
};

-void locks_start_grace(struct lock_manager *);
+struct net;
+void locks_start_grace(struct net *, struct lock_manager *);
void locks_end_grace(struct lock_manager *);
-int locks_in_grace(void);
+int locks_in_grace(struct net *);

/* that will die - we need it for nfs_lock_info */
#include <linux/nfs_fs_i.h>
diff --git a/include/linux/lockd/lockd.h b/include/linux/lockd/lockd.h
index 50e31a2..f5a051a 100644
--- a/include/linux/lockd/lockd.h
+++ b/include/linux/lockd/lockd.h
@@ -262,11 +262,11 @@ typedef int  (*nlm_host_match_fn_t)(void *cur, struct nlm_host *ref);
__be32  nlmsvc_lock(struct svc_rqst *, struct nlm_file *,
    struct nlm_host *, struct nlm_lock *, int,
    struct nlm_cookie *, int);
-__be32  nlmsvc_unlock(struct nlm_file *, struct nlm_lock *);
+__be32  nlmsvc_unlock(struct net *net, struct nlm_file *, struct nlm_lock *);
__be32  nlmsvc_testlock(struct svc_rqst *, struct nlm_file *,
    struct nlm_host *, struct nlm_lock *,
    struct nlm_lock *, struct nlm_cookie *);
-__be32  nlmsvc_cancel_blocked(struct nlm_file *, struct nlm_lock *);
+__be32  nlmsvc_cancel_blocked(struct net *net, struct nlm_file *, struct nlm_lock *);
unsigned long  nlmsvc_retry_blocked(void);
void  nlmsvc_traverse_blocks(struct nlm_host *, struct nlm_file *,
    nlm_host_match_fn_t match);

```
