
Subject: [PATCH 3/4] SUNRPC: service destruction in network namespace context
Posted by [Stanislav Kinsbursky](#) on Wed, 25 Jan 2012 13:47:26 GMT
[View Forum Message](#) <> [Reply to Message](#)

This patch introduces network namespace filter for service destruction function.

Nothing special here - just do exactly the same operations, but only for transports in passed networks namespace context.

BTW, BUG_ON() checks for empty service transports lists were returned into svc_destroy() function. This is because of swithing generic svc_close_all() to networks namespace dependable svc_close_net().

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

```
include/linux/sunrpc/svcsock.h | 2 +-
net/sunrpc/svc.c                | 9 ++++++--
net/sunrpc/svc_xprt.c           | 27 ++++++++-----
3 files changed, 25 insertions(+), 13 deletions(-)
```

```
diff --git a/include/linux/sunrpc/svcsock.h b/include/linux/sunrpc/svcsock.h
index c84e974..cb4ac69 100644
```

```
--- a/include/linux/sunrpc/svcsock.h
+++ b/include/linux/sunrpc/svcsock.h
@@ -34,7 +34,7 @@ struct svc_sock {
/*
 * Function prototypes.
 */
```

```
-void svc_close_all(struct svc_serv *);
+void svc_close_net(struct svc_serv *, struct net *);
int svc_recv(struct svc_rqst *, long);
int svc_send(struct svc_rqst *);
void svc_drop(struct svc_rqst *);
```

```
diff --git a/net/sunrpc/svc.c b/net/sunrpc/svc.c
index a8b49a0..279bbd8 100644
```

```
--- a/net/sunrpc/svc.c
+++ b/net/sunrpc/svc.c
@@ -517,6 +517,8 @@ EXPORT_SYMBOL_GPL(svc_create_pooled);
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_nthreads);
@@ -539,10 +541,13 @@ svc_destroy(struct svc_serv *serv)
 * caller is using--nfsd_mutex in the case of nfsd). So it's
```

```

    * safe to traverse those lists and shut everything down:
    */
- svc_close_all(serv);
+ svc_close_net(serv, net);
+
+ BUG_ON(!list_empty(&serv->sv_permsocks));
+ BUG_ON(!list_empty(&serv->sv_tempsocks));

    if (serv->sv_shutdown)
- serv->sv_shutdown(serv, current->nsproxy->net_ns);
+ serv->sv_shutdown(serv, net);

    cache_clean_deferred(serv);

diff --git a/net/sunrpc/svc_xprt.c b/net/sunrpc/svc_xprt.c
index 493e70b..4bda09d 100644
--- a/net/sunrpc/svc_xprt.c
+++ b/net/sunrpc/svc_xprt.c
@@ -922,17 +922,19 @@ void svc_close_xprt(struct svc_xprt *xprt)
}
EXPORT_SYMBOL_GPL(svc_close_xprt);

-static void svc_close_list(struct list_head *xprt_list)
+static void svc_close_list(struct list_head *xprt_list, struct net *net)
{
    struct svc_xprt *xprt;

    list_for_each_entry(xprt, xprt_list, xprt_list) {
+ if (xprt->xprt_net != net)
+ continue;
        set_bit(XPT_CLOSE, &xprt->xprt_flags);
        set_bit(XPT_BUSY, &xprt->xprt_flags);
    }
}

-static void svc_clear_pools(struct svc_serv *serv)
+static void svc_clear_pools(struct svc_serv *serv, struct net *net)
{
    struct svc_pool *pool;
    struct svc_xprt *xprt;
@@ -944,36 +946,41 @@ static void svc_clear_pools(struct svc_serv *serv)

    spin_lock_bh(&pool->sp_lock);
    list_for_each_entry_safe(xprt, tmp, &pool->sp_sockets, xprt_ready) {
+ if (xprt->xprt_net != net)
+ continue;
        list_del_init(&xprt->xprt_ready);
    }
}

```

```

    spin_unlock_bh(&pool->sp_lock);
}
}

-static void svc_clear_list(struct list_head *xprt_list)
+static void svc_clear_list(struct list_head *xprt_list, struct net *net)
{
    struct svc_xprt *xprt;
    struct svc_xprt *tmp;

    list_for_each_entry_safe(xprt, tmp, xprt_list, xprt_list) {
+ if (xprt->xprt_net != net)
+ continue;
    svc_delete_xprt(xprt);
    }
- BUG_ON(!list_empty(xprt_list));
+ list_for_each_entry(xprt, xprt_list, xprt_list)
+ BUG_ON(xprt->xprt_net == net);
    }

-void svc_close_all(struct svc_serv *serv)
+void svc_close_net(struct svc_serv *serv, struct net *net)
{
- svc_close_list(&serv->sv_tempsocks);
- svc_close_list(&serv->sv_permsocks);
+ svc_close_list(&serv->sv_tempsocks, net);
+ svc_close_list(&serv->sv_permsocks, net);

- svc_clear_pools(serv);
+ svc_clear_pools(serv, net);
/*
 * At this point the sp_sockets lists will stay empty, since
 * svc_enqueue will not add new entries without taking the
 * sp_lock and checking XPT_BUSY.
 */
- svc_clear_list(&serv->sv_tempsocks);
- svc_clear_list(&serv->sv_permsocks);
+ svc_clear_list(&serv->sv_tempsocks, net);
+ svc_clear_list(&serv->sv_permsocks, net);
    }

/*

```
