
Subject: Re: [RFC PATCH] SUNRPC: protect service sockets lists during per-net shutdown

Posted by [bfields](#) on Wed, 16 May 2012 16:34:32 GMT

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

On Fri, May 11, 2012 at 03:41:56PM +0400, Stanislav Kinsbursky wrote:

> Service sv_tempsocks and sv_permsocks lists are accessible by tasks with
> different network namespaces, and thus per-net service destruction must be
> protected.
> These lists are protected by service sv_lock. So lets wrap list munipulations
> with this lock and move tranports destruction outside wrapped area to prevent
> deadlocks.

The comment I originally quoted is still wrong now:

```
/*  
 * The set of xprts (contained in the sv_tempsocks and  
 * sv_permsocks lists) is now constant, since it is modified  
 * only by accepting new sockets (done by service threads in  
 * svc_recv) or aging old ones (done by sv_temptimer), or  
 * configuration changes (excluded by whatever locking the  
 * caller is using--nfsd_mutex in the case of nfsd). So it's  
 * safe to traverse those lists and shut everything down:  
 */
```

And I think that's still a problem.

A server thread could be running svc_recv(), handling a new connection on a listening socket in the network namespace that we're shutting down.

And then I'm not sure exactly what happens, but it doesn't look right. At best we end up adding a connection from the new network namespace after we thought we'd got rid of them all. More likely we crash somewhere.

--b.

```
>  
> Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>  
> ---  
> net/sunrpc/svc_xprt.c | 29 ++++++-----  
> 1 files changed, 25 insertions(+), 4 deletions(-)  
>  
> diff --git a/net/sunrpc/svc_xprt.c b/net/sunrpc/svc_xprt.c  
> index 8195c6a..233f993 100644  
> --- a/net/sunrpc/svc_xprt.c  
> +++ b/net/sunrpc/svc_xprt.c
```

```

> @@ -954,7 +954,8 @@ static void svc_clear_pools(struct svc_serv *serv, struct net *net)
> }
> }
>
> -static void svc_clear_list(struct list_head *xprt_list, struct net *net)
> +static void svc_clear_list(struct list_head *xprt_list, struct net *net,
> +    struct list_head *kill_list)
> {
>     struct svc_xprt *xprt;
>     struct svc_xprt *tmp;
> @@ -962,7 +963,8 @@ static void svc_clear_list(struct list_head *xprt_list, struct net *net)
>     list_for_each_entry_safe(xprt, tmp, xprt_list, xprt_list) {
>         if (xprt->xprt_net != net)
>             continue;
>         - svc_delete_xprt(xprt);
>         + list_move(&xprt->xprt_list, kill_list);
>         + set_bit(XPT_DETACHED, &xprt->xprt_flags);
>     }
>     list_for_each_entry(xprt, xprt_list, xprt_list)
>         BUG_ON(xprt->xprt_net == net);
> @@ -970,6 +972,15 @@ static void svc_clear_list(struct list_head *xprt_list, struct net *net)
>
> void svc_close_net(struct svc_serv *serv, struct net *net)
> {
>     + struct svc_xprt *xprt;
>     + LIST_HEAD(kill_list);
>     +
>     + /*
>     + * Protect the lists, since they can be by tasks with different network
>     + * namespace contexts.
>     + */
>     + spin_lock(&serv->sv_lock);
>     +
>     + svc_close_list(&serv->sv_tempsocks, net);
>     + svc_close_list(&serv->sv_permsocks, net);
>
> @@ -979,8 +990,18 @@ void svc_close_net(struct svc_serv *serv, struct net *net)
>     * svc_enqueue will not add new entries without taking the
>     * sp_lock and checking XPT_BUSY.
>     */
>     - svc_clear_list(&serv->sv_tempsocks, net);
>     - svc_clear_list(&serv->sv_permsocks, net);
>     + svc_clear_list(&serv->sv_tempsocks, net, &kill_list);
>     + svc_clear_list(&serv->sv_permsocks, net, &kill_list);
>     +
>     + spin_unlock(&serv->sv_lock);
>     +
>     + /*

```

```
> + * Destroy collected transports.  
> + * Note: transports has been marked as XPT_DETACHED on svc_clear_list(),  
> + * so no need to protect against list_del() in svc_delete_xprt().  
> + */  
> + list_for_each_entry(xprt, &kill_list, xpt_list)  
> + svc_delete_xprt(xprt);  
> }  
>  
> /*  
>
```
