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

Posted by [Stanislav Kinsbursky](#) on Fri, 11 May 2012 11:41:56 GMT

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

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.

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);
+
+    for (xprt = list_first_entry(&serv->xprt_list, struct svc_xprt, xprt_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: tranports 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);
+ }

/*

```

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, xpt_list) {
>      if (xprt->xpt_net != net)
>        continue;
> -    svc_delete_xprt(xprt);
> +    list_move(&xprt->xpt_list, kill_list);

```

```

> + set_bit(XPT_DETACHED, &xprt->xpt_flags);
> }
> list_for_each_entry(xprt, xprt_list, xpt_list)
> BUG_ON(xprt->xpt_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);
> }
>
> /*
>

```

Subject: [RFC PATCH v2] SUNRPC: protect service sockets lists during per-net shutdown

Posted by [Stanislav Kinsbursky](#) on Mon, 21 May 2012 08:51:47 GMT

v2: destruction of currently processing transport added:

- 1) Added marking of currently processing transports with XPT_CLOSE on per-net shutdown. These transports will be destroyed in svc_xprt_enqueue() (instead of enqueueing).
- 2) newly created temporary transport in svc_recv() will be destroyed, if it's "parent" was marked with currently processing transports with XPT_CLOSE.
- 3) spin_lock(&serv->sv_lock) was replaced by spin_lock_bh() in svc_close_net(&serv->sv_lock).

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 let's wrap list manipulations with this lock and move transports destruction outside wrapped area to prevent deadlocks.

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

net/sunrpc/svc_xprt.c | 56 +-----
1 files changed, 52 insertions(+), 4 deletions(-)

diff --git a/net/sunrpc/svc_xprt.c b/net/sunrpc/svc_xprt.c

index 37a1f66..947d3cb 100644

--- a/net/sunrpc/svc_xprt.c

+++ b/net/sunrpc/svc_xprt.c

@@ -320,6 +320,7 @@ void svc_xprt_enqueue(struct svc_xprt *xprt)

struct svc_pool *pool;

struct svc_rqst *rqstp;

int cpu;

+ int destroy = 0;

if (!svc_xprt_has_something_to_do(xprt))

return;

@@ -338,6 +339,17 @@ void svc_xprt_enqueue(struct svc_xprt *xprt)

pool->sp_stats.packets++;

+ /*

+ * Check transport close flag. It could be marked as closed on per-net

+ * service shutdown.

+ */

+ if (test_bit(XPT_CLOSE, &xprt->xpt_flags)) {

+ /* Don't enqueue transport if it has to be destroyed. */

+ dprintk("svc: transport %p have to be closed\n", xprt);

+ destroy++;

+ goto out_unlock;

+ }

```

+
/* Mark transport as busy. It will remain in this state until
 * the provider calls svc_xprt_received. We update XPT_BUSY
 * atomically because it also guards against trying to enqueue
@@ -374,6 +386,8 @@ void svc_xprt_enqueue(struct svc_xprt *xprt)

out_unlock:
    spin_unlock_bh(&pool->sp_lock);
+ if (destroy)
+ svc_delete_xprt(xprt);
}
EXPORT_SYMBOL_GPL(svc_xprt_enqueue);

@@ -717,6 +731,13 @@ int svc_rcv(struct svc_rqst *rqstp, long timeout)
    __module_get(newxpt->xpt_class->xcl_owner);
    svc_check_conn_limits(xprt->xpt_server);
    spin_lock_bh(&serv->sv_lock);
+ if (test_bit(XPT_CLOSE, &xprt->xpt_flags)) {
+ dprintk("svc_rcv: found XPT_CLOSE on listener\n");
+ set_bit(XPT_DETACHED, &newxpt->xpt_flags);
+ spin_unlock_bh(&pool->sp_lock);
+ svc_delete_xprt(newxpt);
+ goto out_closed;
+ }
    set_bit(XPT_TEMP, &newxpt->xpt_flags);
    list_add(&newxpt->xpt_list, &serv->sv_tempsocks);
    serv->sv_tmprcnt++;
@@ -742,6 +763,7 @@ int svc_rcv(struct svc_rqst *rqstp, long timeout)
    len = xprt->xpt_ops->xpo_recvfrom(rqstp);
    dprintk("svc: got len=%d\n", len);
}
+out_closed:
    svc_xprt_received(xprt);

/* No data, incomplete (TCP) read, or accept() */
@@ -939,6 +961,7 @@ static void svc_clear_pools(struct svc_serv *serv, struct net *net)
    struct svc_pool *pool;
    struct svc_xprt *xprt;
    struct svc_xprt *tmp;
+ struct svc_rqst *rqstp;
    int i;

    for (i = 0; i < serv->sv_nrpoools; i++) {
@@ -950,11 +973,16 @@ static void svc_clear_pools(struct svc_serv *serv, struct net *net)
        continue;
        list_del_init(&xprt->xpt_ready);
    }
+ list_for_each_entry(rqstp, &pool->sp_all_threads, rq_all) {

```

```

+ if (rqstp->rq_xprt && rqstp->rq_xprt->xpt_net == net)
+ set_bit(XPT_CLOSE, &rqstp->rq_xprt->xpt_flags);
+ }
  spin_unlock_bh(&pool->sp_lock);
}
}

-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 +990,8 @@ static void svc_clear_list(struct list_head *xprt_list, struct net *net)
  list_for_each_entry_safe(xprt, tmp, xprt_list, xpt_list) {
    if (xprt->xpt_net != net)
      continue;
- svc_delete_xprt(xprt);
+ list_move(&xprt->xpt_list, kill_list);
+ set_bit(XPT_DETACHED, &xprt->xpt_flags);
  }
  list_for_each_entry(xprt, xprt_list, xpt_list)
    BUG_ON(xprt->xpt_net == net);
@@ -970,6 +999,15 @@ 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, *tmp;
+ LIST_HEAD(kill_list);
+
+ /*
+  * Protect the lists, since they can be by tasks with different network
+  * namespace contexts.
+  */
+ spin_lock_bh(&serv->sv_lock);
+
  svc_close_list(&serv->sv_tempsocks, net);
  svc_close_list(&serv->sv_permsocks, net);

@@ -979,8 +1017,18 @@ void svc_close_net(struct svc_serv *serv, struct net *net)
  * svc_xprt_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_bh(&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_safe(xprt, tmp, &kill_list, xpt_list)
+   svc_delete_xprt(xprt);
+ }

/*
```
