Subject: [patch 0/3][NETNS45][V2] remove timewait sockets at namespace exit
Posted by Daniel Lezcano on Thu, 27 Sep 2007 11:04:00 GMT
View Forum Message <> Reply to Message

Denis Lunev spotted that using a reference to the network namespace
with the timewait sockets will be a waste of time because they
are pointless while we will remove the network stack at network
namespace exit.

The following patches do the following:
 - fix missing network namespace reference in timewait
 socket
 - do some changes in timewait socket code to prepare
 the next patch, especially split code taking a lock
 - do the effective timewait socket cleanup at network
 namespace exit.

The following code is a test program which creates 100 timewait
sockets.

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/poll.h>
#include <netinet/in.h>
#include <netinet/tcp.h>
#include <arpa/inet.h>

#include <unistd.h>

#define MAXCONN 100

int client(int *fds)
{
	int i;
	struct sockaddr_in addr;

	close(fds[1]);

	memset(&addr, 0, sizeof(addr));

	addr.sin_family = AF_INET;
	addr.sin_port =  htons(10000);
	addr.sin_addr.s_addr = inet_addr("127.0.0.1");

	if (read(fds[0], &i, sizeof(i)) == -1) {
		perror("read");
```

```
		return 1;
	}

	for (i = 0; i < MAXCONN; i++) {
		int fd = socket(PF_INET, SOCK_STREAM, 0);
		if (fd == -1) {
			perror("socket");
			return 1;
		}

		if (connect(fd, (const struct sockaddr *)&addr, sizeof(addr))) {
			perror("connect");
			return 1;
		}
	}

	return 0;
}

int server(int *fds)
{
	int i, fd, fdpoll[MAXCONN];
	struct sockaddr_in addr;
	socklen_t socklen = sizeof(addr);

	close(fds[0]);

	fd = socket(PF_INET, SOCK_STREAM, 0);
	if (fd == -1) {
		perror("socket");
		return 1;
	}

	memset(&addr, 0, sizeof(addr));

	addr.sin_family = AF_INET;
	addr.sin_port = htons(10000);
	addr.sin_addr.s_addr = inet_addr("127.0.0.1");

	if (bind(fd, (const struct sockaddr *)&addr, sizeof(addr))) {
		perror("bind");
		return 1;
	}

	if (listen(fd, MAXCONN)) {
		perror("listen");
		return 1;
	}
```

```
	if (write(fds[1], &i, sizeof(i)) == -1) {
		perror("write");
		return 1;
	}

	for (i = 0; i < MAXCONN; i++) {
		int f = accept(fd, (struct sockaddr *)&addr, &socklen);
		if (f == -1) {
			perror("accept");
			return 1;
		}
		fdpoll[i] = f;
	}

	return 0;
}

int main(int argc, char *argv[])
{
	int fds[2];
	int pid;

	if (pipe(fds)) {
		perror("pipe");
		return 1;
	}

	pid = fork();
	if (pid == -1) {
		perror("fork");
		return 1;
	}

	if (!pid)
		return client(fds);
	else
		return server(fds);
}

--
```

Subject: [patch 1/3][NETNS45][V2] add a reference to the netns for timewait
Posted by Daniel Lezcano on Thu, 27 Sep 2007 11:04:01 GMT
View Forum Message <> Reply to Message

From: Daniel Lezcano <dlezcano@fr.ibm.com>

When a socket changes to a timewait socket, the network namespace
is not copied from the original socket.

Here we hold a usage reference, not the ref count on the network
namespace, so the network namespace will be freed either the usage
reference is not 0. The network namespace cleanup function will
fail if there is any usage of it. In this case, we should ensure
there is no usage of the network namespace.

Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>
---
 include/net/inet_timewait_sock.h | 2 ++
 net/ipv4/inet_timewait_sock.c | 1 +
 2 files changed, 3 insertions(+)

Index: linux-2.6-netns/include/net/inet_timewait_sock.h
===
--- linux-2.6-netns.orig/include/net/inet_timewait_sock.h
+++ linux-2.6-netns/include/net/inet_timewait_sock.h
@@ -197,12 +197,14 @@ static inline void inet_twsk_put(struct
 {
  if (atomic_dec_and_test(&tw->tw_refcnt)) {
   struct module *owner = tw->tw_prot->owner;
+ struct net *net = tw->tw_net;
   twsk_destructor((struct sock *)tw);
 #ifdef SOCK_REFCNT_DEBUG
   printk(KERN_DEBUG "%s timewait_sock %p released\n",
          tw->tw_prot->name, tw);
 #endif
   kmem_cache_free(tw->tw_prot->twsk_prot->twsk_slab, tw);
+ release_net(net);
   module_put(owner);
  }
 }
Index: linux-2.6-netns/net/ipv4/inet_timewait_sock.c
===
--- linux-2.6-netns.orig/net/ipv4/inet_timewait_sock.c
+++ linux-2.6-netns/net/ipv4/inet_timewait_sock.c
@@ -108,6 +108,7 @@ struct inet_timewait_sock *inet_twsk_all
   tw->tw_hash   = sk->sk_hash;
   tw->tw_ipv6only   = 0;
   tw->tw_prot   = sk->sk_prot_creator;
+ tw->tw_net   = hold_net(sk->sk_net);

```
    atomic_set(&tw->tw_refcnt, 1);
    inet_twsk_dead_node_init(tw);
    __module_get(tw->tw_prot->owner);
```

--

____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: [patch 2/3][NETNS45][V2] make timewait unhash lock free
Posted by Daniel Lezcano on Thu, 27 Sep 2007 11:04:02 GMT
View Forum Message <> Reply to Message

From: Daniel Lezcano <dlezcano@fr.ibm.com>

The network namespace cleanup will remove all timewait sockets
related to it because there are pointless.

The problem is we need to browse the established hash table and
for that we need to take the lock. For each timesocket we call
inet_deschedule and this one take the established hash table lock
too.

The following patchset split the removing of the established hash
into two parts, one removing the node from the hash and another
taking the lock and calling the first one.

The network namespace cleanup can be done calling the lock free
function.

Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>

```
---
 include/net/inet_timewait_sock.h |   13 +++++++++++++
 net/ipv4/inet_timewait_sock.c    |   40 ++++++++++++++++++++++++++++------------
 2 files changed, 41 insertions(+), 12 deletions(-)

Index: linux-2.6-netns/net/ipv4/inet_timewait_sock.c
===================================================================
--- linux-2.6-netns.orig/net/ipv4/inet_timewait_sock.c
+++ linux-2.6-netns/net/ipv4/inet_timewait_sock.c
@@ -13,25 +13,28 @@
 #include <net/inet_timewait_sock.h>
 #include <net/ip.h>

-/* Must be called with locally disabled BHs. */
-static void __inet_twsk_kill(struct inet_timewait_sock *tw,
```

```
-       struct inet_hashinfo *hashinfo)
+static inline int inet_twsk_unehash(struct inet_timewait_sock *tw,
+        struct inet_hashinfo *hashinfo)
 {
- struct inet_bind_hashbucket *bhead;
- struct inet_bind_bucket *tb;
- /* Unlink from established hashes. */
- struct inet_ehash_bucket *ehead = inet_ehash_bucket(hashinfo, tw->tw_hash);
+ struct inet_ehash_bucket *ehead =
+  inet_ehash_bucket(hashinfo, tw->tw_hash);

  write_lock(&ehead->lock);
- if (hlist_unhashed(&tw->tw_node)) {
+ if (__inet_twsk_unehash(tw)) {
   write_unlock(&ehead->lock);
-  return;
+  return 1;
  }
- __hlist_del(&tw->tw_node);
- sk_node_init(&tw->tw_node);
  write_unlock(&ehead->lock);

- /* Disassociate with bind bucket. */
+ return 0;
+}
+
+void inet_twsk_unbhash(struct inet_timewait_sock *tw,
+        struct inet_hashinfo *hashinfo)
+{
+ struct inet_bind_hashbucket *bhead;
+ struct inet_bind_bucket *tb;
+
  bhead = &hashinfo->bhash[inet_bhashfn(tw->tw_net, tw->tw_num, hashinfo->bhash_size)];
  spin_lock(&bhead->lock);
  tb = tw->tw_tb;
@@ -39,6 +42,19 @@ static void __inet_twsk_kill(struct inet
  tw->tw_tb = NULL;
  inet_bind_bucket_destroy(hashinfo->bind_bucket_cachep, tb);
  spin_unlock(&bhead->lock);
+}
+
+/* Must be called with locally disabled BHs. */
+static void __inet_twsk_kill(struct inet_timewait_sock *tw,
+        struct inet_hashinfo *hashinfo)
+{
+ /* Unlink from established hashes. */
+ if (inet_twsk_unehash(tw, hashinfo))
+  return;
```

```
+
+ /* Disassociate with bind bucket. */
+ inet_twsk_unbhash(tw, hashinfo);
+
 #ifdef SOCK_REFCNT_DEBUG
  if (atomic_read(&tw->tw_refcnt) != 1) {
   printk(KERN_DEBUG "%s timewait_sock %p refcnt=%d\n",
Index: linux-2.6-netns/include/net/inet_timewait_sock.h
===================================================================
--- linux-2.6-netns.orig/include/net/inet_timewait_sock.h
+++ linux-2.6-netns/include/net/inet_timewait_sock.h
@@ -173,6 +173,19 @@ static inline int inet_twsk_del_dead_nod
  return 0;
 }

+static inline int __inet_twsk_unehash(struct inet_timewait_sock *tw)
+{
+ if (hlist_unhashed(&tw->tw_node))
+  return 1;
+ __hlist_del(&tw->tw_node);
+ sk_node_init(&tw->tw_node);
+
+ return 0;
+}
+
+extern void inet_twsk_unbhash(struct inet_timewait_sock *tw,
+         struct inet_hashinfo *hashinfo);
+
 #define inet_twsk_for_each(tw, node, head) \
  hlist_for_each_entry(tw, node, head, tw_node)


--
```

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: [patch 3/3][NETNS45][V2] remove timewait sockets at cleanup
Posted by Daniel Lezcano on Thu, 27 Sep 2007 11:04:03 GMT
View Forum Message <> Reply to Message

From: Daniel Lezcano <dlezcano@fr.ibm.com>

Denis Lunev spotted that if we take a reference to the network namespace
with the timewait sockets, we will need to wait for their expiration to
have the network namespace freed. This is a waste of time, the timewait

sockets are for avoiding to receive a duplicate packet from the network,
if the network namespace is freed, the network stack is removed, so no
chance to receive any packets from the outside world.

This patchset remove/destroy the timewait sockets when the
network namespace is freed.

Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>
---
 net/ipv4/tcp.c | 53 ++
 1 file changed, 53 insertions(+)

Index: linux-2.6-netns/net/ipv4/tcp.c
===
--- linux-2.6-netns.orig/net/ipv4/tcp.c
+++ linux-2.6-netns/net/ipv4/tcp.c
@@ -2432,8 +2432,61 @@ static int tcp_net_init(struct net *net)
  return 0;
 }

+/*
+ * Wipeout tcp timewait sockets, they are no longer needed
+ * because we destroy the network namespace, so no risk to
+ * have duplicate packet coming from the network
+ */
+static void tcp_net_exit(struct net *net)
+{
+ struct inet_timewait_sock *tw;
+ struct sock *sk;
+ struct hlist_node *node, *tmp;
+ int h;
+ int nbsock = 0;
+
+ /* Browse the the established hash table */
+ for (h = 0; h < (tcp_hashinfo.ehash_size); h++) {
+   struct inet_ehash_bucket *head =
+   inet_ehash_bucket(&tcp_hashinfo, h);
+
+ /* Take the look and disable bh */
+   write_lock_bh(&head->lock);
+
+ sk_for_each_safe(sk, node, tmp, &head->twchain) {
+
+   tw = inet_twsk(sk);
+   if (tw->tw_net != net)
+   continue;
+
+   /* deschedule the timewait socket */

```
+   spin_lock(&tcp_death_row.death_lock);
+   if (inet_twsk_del_dead_node(tw)) {
+    inet_twsk_put(tw);
+    if (--tcp_death_row.tw_count == 0)
+     del_timer(&tcp_death_row.tw_timer);
+   }
+   spin_unlock(&tcp_death_row.death_lock);
+
+   /* remove it from the established hash table */
+   __inet_twsk_unehash(tw);
+
+   /* remove it from the bind hash table */
+   inet_twsk_unbhash(tw, tcp_death_row.hashinfo);
+
+   /* last put */
+   inet_twsk_put(tw);
+
+   nbsock++;
+  }
+
+  write_unlock_bh(&head->lock);
+ }
+}
+
 static struct pernet_operations tcp_net_ops = {
   .init = tcp_net_init,
+ .exit = tcp_net_exit,
 };

 void __init tcp_init(void)

--
```

Subject: Re: [patch 1/3][NETNS45][V2] add a reference to the netns for timewait
Posted by den on Thu, 27 Sep 2007 12:34:51 GMT
View Forum Message <> Reply to Message

Daniel Lezcano wrote:
> From: Daniel Lezcano <dlezcano@fr.ibm.com>
>
> When a socket changes to a timewait socket, the network namespace
> is not copied from the original socket.
>

> Here we hold a usage reference, not the ref count on the network
> namespace, so the network namespace will be freed either the usage
> reference is not 0. The network namespace cleanup function will
> fail if there is any usage of it. In this case, we should ensure
> there is no usage of the network namespace.
>
> Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>
Acked-by: Denis V. Lunev <den@openvz.org>

____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [patch 2/3][NETNS45][V2] make timewait unhash lock free
Posted by den on Thu, 27 Sep 2007 12:46:28 GMT
View Forum Message <> Reply to Message

Sorry for a delay in answer. A was ill last three days.
Some stylistic comments inside

Daniel Lezcano wrote:
> From: Daniel Lezcano <dlezcano@fr.ibm.com>
>
> The network namespace cleanup will remove all timewait sockets
> related to it because there are pointless.
>
> The problem is we need to browse the established hash table and
> for that we need to take the lock. For each timesocket we call
> inet_deschedule and this one take the established hash table lock
> too.
>
> The following patchset split the removing of the established hash
> into two parts, one removing the node from the hash and another
> taking the lock and calling the first one.
>
> The network namespace cleanup can be done calling the lock free
> function.
>
> Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>
> ---
> include/net/inet_timewait_sock.h | 13 +++++++++++++
> net/ipv4/inet_timewait_sock.c | 40 +++++++++++++++++++++++++++++------------
> 2 files changed, 41 insertions(+), 12 deletions(-)
>
> Index: linux-2.6-netns/net/ipv4/inet_timewait_sock.c
> ===
> --- linux-2.6-netns.orig/net/ipv4/inet_timewait_sock.c

> +++ linux-2.6-netns/net/ipv4/inet_timewait_sock.c
> @@ -13,25 +13,28 @@
> #include <net/inet_timewait_sock.h>
> #include <net/ip.h>
>
> -/* Must be called with locally disabled BHs. */
> -static void __inet_twsk_kill(struct inet_timewait_sock *tw,
> -   struct inet_hashinfo *hashinfo)
> +static inline int inet_twsk_unehash(struct inet_timewait_sock *tw,
> +   struct inet_hashinfo *hashinfo)
> {
> - struct inet_bind_hashbucket *bhead;
> - struct inet_bind_bucket *tb;
> - /* Unlink from established hashes. */
> - struct inet_ehash_bucket *ehead = inet_ehash_bucket(hashinfo, tw->tw_hash);
> + struct inet_ehash_bucket *ehead =
> + inet_ehash_bucket(hashinfo, tw->tw_hash);
>
>   write_lock(&ehead->lock);
> - if (hlist_unhashed(&tw->tw_node)) {
> + if (__inet_twsk_unehash(tw)) {
>   write_unlock(&ehead->lock);
> - return;
> + return 1;
>   }
> - __hlist_del(&tw->tw_node);
> - sk_node_init(&tw->tw_node);
>   write_unlock(&ehead->lock);
>
> - /* Disassociate with bind bucket. */
> + return 0;
> +}
as far as I can understand the code, it will look better as below

```
struct inet_ehash_bucket *ehead =
    inet_ehash_bucket(hashinfo, tw->tw_hash);
int ret;

write_lock(&ehead->lock);
ret = __inet_twsk_unehash(tw);
write_unlock(&ehead->lock);
return ret;
```

> +
> +void inet_twsk_unbhash(struct inet_timewait_sock *tw,
> +   struct inet_hashinfo *hashinfo)
> +{

```
> + struct inet_bind_hashbucket *bhead;
> + struct inet_bind_bucket *tb;
> +
>   bhead = &hashinfo->bhash[inet_bhashfn(tw->tw_net, tw->tw_num, hashinfo->bhash_size)];
>   spin_lock(&bhead->lock);
>   tb = tw->tw_tb;
> @@ -39,6 +42,19 @@ static void __inet_twsk_kill(struct inet
>   tw->tw_tb = NULL;
>   inet_bind_bucket_destroy(hashinfo->bind_bucket_cachep, tb);
>   spin_unlock(&bhead->lock);
> +}
> +
> +/* Must be called with locally disabled BHs. */
> +static void __inet_twsk_kill(struct inet_timewait_sock *tw,
> +        struct inet_hashinfo *hashinfo)
> +{
> + /* Unlink from established hashes. */
> + if (inet_twsk_unehash(tw, hashinfo))
> +  return;
> +
> + /* Disassociate with bind bucket. */
> + inet_twsk_unbhash(tw, hashinfo);
> +
>  #ifdef SOCK_REFCNT_DEBUG
>   if (atomic_read(&tw->tw_refcnt) != 1) {
>    printk(KERN_DEBUG "%s timewait_sock %p refcnt=%d\n",
> Index: linux-2.6-netns/include/net/inet_timewait_sock.h
> ===================================================================
> --- linux-2.6-netns.orig/include/net/inet_timewait_sock.h
> +++ linux-2.6-netns/include/net/inet_timewait_sock.h
> @@ -173,6 +173,19 @@ static inline int inet_twsk_del_dead_nod
>   return 0;
>  }
>
> +static inline int __inet_twsk_unehash(struct inet_timewait_sock *tw)
> +{
> + if (hlist_unhashed(&tw->tw_node))
> +  return 1;
> + __hlist_del(&tw->tw_node);
> + sk_node_init(&tw->tw_node);
> +
>> see above about inet_twsk_unehash. We should insert
        /* Disassociate with bind bucket. */
>> here
> + return 0;
> +}
> +
> +extern void inet_twsk_unbhash(struct inet_timewait_sock *tw,
```

> + struct inet_hashinfo *hashinfo);
> +
> #define inet_twsk_for_each(tw, node, head) \
> hlist_for_each_entry(tw, node, head, tw_node)
>
>

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [patch 2/3][NETNS45][V2] make timewait unhash lock free
Posted by Daniel Lezcano on Thu, 27 Sep 2007 13:09:00 GMT
View Forum Message <> Reply to Message

Denis V. Lunev wrote:
> Sorry for a delay in answer. A was ill last three days.
> Some stylistic comments inside
>
> Daniel Lezcano wrote:
>> From: Daniel Lezcano <dlezcano@fr.ibm.com>
>>
>> The network namespace cleanup will remove all timewait sockets
>> related to it because there are pointless.
>>
>> The problem is we need to browse the established hash table and
>> for that we need to take the lock. For each timesocket we call
>> inet_deschedule and this one take the established hash table lock
>> too.
>>
>> The following patchset split the removing of the established hash
>> into two parts, one removing the node from the hash and another
>> taking the lock and calling the first one.
>>
>> The network namespace cleanup can be done calling the lock free
>> function.
>>
>> Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>
>> ---
>> include/net/inet_timewait_sock.h | 13 ++++++++++
>> net/ipv4/inet_timewait_sock.c | 40 ++++++++++++++++++++++++++++++----------
>> 2 files changed, 41 insertions(+), 12 deletions(-)
>>
>> Index: linux-2.6-netns/net/ipv4/inet_timewait_sock.c
>> ===
>> --- linux-2.6-netns.orig/net/ipv4/inet_timewait_sock.c

>> +++ linux-2.6-netns/net/ipv4/inet_timewait_sock.c
>> @@ -13,25 +13,28 @@
>> #include <net/inet_timewait_sock.h>
>> #include <net/ip.h>
>>
>> -/* Must be called with locally disabled BHs. */
>> -static void __inet_twsk_kill(struct inet_timewait_sock *tw,
>> -   struct inet_hashinfo *hashinfo)
>> +static inline int inet_twsk_unehash(struct inet_timewait_sock *tw,
>> +   struct inet_hashinfo *hashinfo)
>> {
>> - struct inet_bind_hashbucket *bhead;
>> - struct inet_bind_bucket *tb;
>> - /* Unlink from established hashes. */
>> - struct inet_ehash_bucket *ehead = inet_ehash_bucket(hashinfo, tw->tw_hash);
>> + struct inet_ehash_bucket *ehead =
>> + inet_ehash_bucket(hashinfo, tw->tw_hash);
>>
>>   write_lock(&ehead->lock);
>> - if (hlist_unhashed(&tw->tw_node)) {
>> + if (__inet_twsk_unehash(tw)) {
>>   write_unlock(&ehead->lock);
>> - return;
>> + return 1;
>>   }
>> - __hlist_del(&tw->tw_node);
>> - sk_node_init(&tw->tw_node);
>>   write_unlock(&ehead->lock);
>>
>> - /* Disassociate with bind bucket. */
>> + return 0;
>> +}
> as far as I can understand the code, it will look better as below
>
> struct inet_ehash_bucket *ehead =
>   inet_ehash_bucket(hashinfo, tw->tw_hash);
> int ret;
>
> write_lock(&ehead->lock);
> ret = __inet_twsk_unehash(tw);
> write_unlock(&ehead->lock);
> return ret;

Right. Will fix that. Thanks.

[cut]

>> +
>>> see above about inet_twsk_unehash. We should insert
> /* Disassociate with bind bucket. */
>>> here
>> + return 0;
>> +}

Is this comment for me ?

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [patch 2/3][NETNS45][V2] make timewait unhash lock free
Posted by den on Thu, 27 Sep 2007 13:18:54 GMT
View Forum Message <> Reply to Message

Daniel Lezcano wrote:
> [cut]
>>> +
>>>> see above about inet_twsk_unehash. We should insert
>> /* Disassociate with bind bucket. */
>>>> here
>>> + return 0;
>>> +}
>
> Is this comment for me ?
>

yes. It will be dropped in the upper part of the code and have to be
inserted here instead

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [patch 3/3][NETNS45][V2] remove timewait sockets at cleanup
Posted by den on Thu, 27 Sep 2007 13:21:57 GMT
View Forum Message <> Reply to Message

Daniel Lezcano wrote:
> From: Daniel Lezcano <dlezcano@fr.ibm.com>
>
> Denis Lunev spotted that if we take a reference to the network namespace
> with the timewait sockets, we will need to wait for their expiration to

> have the network namespace freed. This is a waste of time, the timewait
> sockets are for avoiding to receive a duplicate packet from the network,
> if the network namespace is freed, the network stack is removed, so no
> chance to receive any packets from the outside world.
>
> This patchset remove/destroy the timewait sockets when the
> network namespace is freed.
>
> Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>
> ---
> net/ipv4/tcp.c | 53 ++
> 1 file changed, 53 insertions(+)
>

[...]
This place seems non-trival and broken for me :(May be I am wrong.
> +   write_lock_bh(&head->lock);
> +
> + sk_for_each_safe(sk, node, tmp, &head->twchain) {
> +
> +   tw = inet_twsk(sk);
> +   if (tw->tw_net != net)
> +   continue;
> +
> +   /* deschedule the timewait socket */
> +   spin_lock(&tcp_death_row.death_lock);
> +   if (inet_twsk_del_dead_node(tw)) {
> +   inet_twsk_put(tw);
> +   if (--tcp_death_row.tw_count == 0)
> +   del_timer(&tcp_death_row.tw_timer);
There is a call inet_twsk_deschedule which do exactly what we need to

```
void inet_twsk_deschedule(struct inet_timewait_sock *tw,
                          struct inet_timewait_death_row *twdr)
{
        spin_lock(&twdr->death_lock);
        if (inet_twsk_del_dead_node(tw)) {
                inet_twsk_put(tw);
                if (--twdr->tw_count == 0)
                        del_timer(&twdr->tw_timer);
        }
        spin_unlock(&twdr->death_lock);
        __inet_twsk_kill(tw, twdr->hashinfo);
}
```

and, from my point of view, your patch [2] is even not needed. We should do

restart:

```
write_lock_bh(&head->lock);
sk_for_each_safe(sk, node, tmp, &head->twchain) {
    tw = inet_twsk(sk);
    if (tw->tw_net != net)
        continue;
    sock_hold(sk);
    write_unlock_bh(&head->lock);

    inet_twsk_deschedule(tw, &tcp_death_row);
    inet_twsk_put(tw);
    goto restart;
}
```

This removes serious locking issue. You have introduced dependency
between write_lock_bh(&head->lock); and
spin_lock(&tcp_death_row.death_lock);
This should be at least checked and documented in the headers. I am not
sure that this is correct.

If my approach is correct, your second patch is not needed.

It will also worth to local_bh_enable() at the very beginning and remove
_bh from write_lock.

Regards,
Den

Subject: Re: [patch 3/3][NETNS45][V2] remove timewait sockets at cleanup
Posted by Daniel Lezcano on Thu, 27 Sep 2007 14:38:16 GMT
View Forum Message <> Reply to Message

Denis V. Lunev wrote:
> Daniel Lezcano wrote:
>> From: Daniel Lezcano <dlezcano@fr.ibm.com>
>>
>> Denis Lunev spotted that if we take a reference to the network namespace
>> with the timewait sockets, we will need to wait for their expiration to
>> have the network namespace freed. This is a waste of time, the timewait
>> sockets are for avoiding to receive a duplicate packet from the network,
>> if the network namespace is freed, the network stack is removed, so no
>> chance to receive any packets from the outside world.
>>
>> This patchset remove/destroy the timewait sockets when the
>> network namespace is freed.
>>
>> Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>
>> ---

>> net/ipv4/tcp.c | 53 ++
>> 1 file changed, 53 insertions(+)
>>
>
> [...]
> This place seems non-trival and broken for me :(May be I am wrong.
>> +   write_lock_bh(&head->lock);
>> +
>> + sk_for_each_safe(sk, node, tmp, &head->twchain) {
>> +
>> +   tw = inet_twsk(sk);
>> +   if (tw->tw_net != net)
>> +   continue;
>> +
>> +   /* deschedule the timewait socket */
>> +   spin_lock(&tcp_death_row.death_lock);
>> +   if (inet_twsk_del_dead_node(tw)) {
>> +   inet_twsk_put(tw);
>> +   if (--tcp_death_row.tw_count == 0)
>> +   del_timer(&tcp_death_row.tw_timer);
> There is a call inet_twsk_deschedule which do exactly what we need to
>
> void inet_twsk_deschedule(struct inet_timewait_sock *tw,
>   struct inet_timewait_death_row *twdr)
> {
>   spin_lock(&twdr->death_lock);
>   if (inet_twsk_del_dead_node(tw)) {
>   inet_twsk_put(tw);
>   if (--twdr->tw_count == 0)
>   del_timer(&twdr->tw_timer);
>   }
>   spin_unlock(&twdr->death_lock);
>   __inet_twsk_kill(tw, twdr->hashinfo);
> }
>
> and, from my point of view, your patch [2] is even not needed. We should do
>
> restart:
>   write_lock_bh(&head->lock);
>   sk_for_each_safe(sk, node, tmp, &head->twchain) {
>   tw = inet_twsk(sk);
>   if (tw->tw_net != net)
>   continue;
>   sock_hold(sk);
>   write_unlock_bh(&head->lock);
>   inet_twsk_deschedule(tw, &tcp_death_row);
>   inet_twsk_put(tw);
>   goto restart;

> }
>
> This removes serious locking issue. You have introduced dependency
> between write_lock_bh(&head->lock); and
> spin_lock(&tcp_death_row.death_lock);
> This should be at least checked and documented in the headers. I am not
> sure that this is correct.
> If my approach is correct, your second patch is not needed.
>
> It will also worth to local_bh_enable() at the very beginning and remove
> _bh from write_lock.

local_bh_disable / local_bh_enable at the very beginning ?

like that ?

```
local_bh_disable();

/* Browse the the established hash table */
for (h = 0; h < (tcp_hashinfo.ehash_size); h++) {
        struct inet_ehash_bucket *head =
                inet_ehash_bucket(&tcp_hashinfo, h);
restart:
        write_lock(&head->lock);
        sk_for_each_safe(sk, node, tmp, &head->twchain) {
            tw = inet_twsk(sk);
            if (tw->tw_net != net)
                continue;
            sock_hold(sk);
            write_unlock(&head->lock);
            inet_twsk_deschedule(tw, &tcp_death_row);
            inet_twsk_put(tw);
            goto restart;
        }
}

local_bh_enable();
```

Subject: Re: [patch 3/3][NETNS45][V2] remove timewait sockets at cleanup
Posted by den on Fri, 28 Sep 2007 06:46:34 GMT
View Forum Message <> Reply to Message

Daniel Lezcano wrote:

```
> local_bh_disable();
>
> /* Browse the the established hash table */
> for (h = 0; h < (tcp_hashinfo.ehash_size); h++) {
>                 struct inet_ehash_bucket *head =
>                         inet_ehash_bucket(&tcp_hashinfo, h);
> restart:
>         write_lock(&head->lock);
>         sk_for_each_safe(sk, node, tmp, &head->twchain) {
>             tw = inet_twsk(sk);
>             if (tw->tw_net != net)
>                 continue;
>             sock_hold(sk);
>             write_unlock(&head->lock);
>             inet_twsk_deschedule(tw, &tcp_death_row);
>             inet_twsk_put(tw);
>             goto restart;
>         }
> }
>
> local_bh_enable();
```

yes :)