

[PATCH 2.6.23-rc6 Resending] NETWORKING : Edge Triggered EPOLLOUT events get missed for TCP sockets

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-09/msg05979.html>

- *From:* Nagendra Tomar <tomar_iisc@xxxxxxxx>
 - *Date:* Wed, 19 Sep 2007 15:37:09 -0700 (PDT)
-

The `tcp_check_space()` function calls `tcp_new_space()` only if the `SOCK_NOSPACE` bit is set in the socket flags. This is causing Edge Triggered EPOLLOUT events to be missed for TCP sockets, as the `ep_poll_callback()` is not called from the wakeup routine.

The `SOCK_NOSPACE` bit indicates the user's intent to perform writes on that socket (set in `tcp_sendmsg` and `tcp_poll`). I believe the idea behind the `SOCK_NOSPACE` check is to optimize away the `tcp_new_space` call in cases when user is not interested in writing to the socket. These two take care of all possible scenarios in which a user can convey his intent to write on that socket.

Case 1: `tcp_sendmsg` detects lack of `sndbuf` space

Case 2: `tcp_poll` returns not writable

This is fine if we do not deal with `epoll`'s Edge Triggered events (EPOLLET). With ET events we can have a scenario where the `SOCK_NOSPACE` bit is not set, as the user has neither done a `sendmsg` nor a `poll/epoll` call that returned with the `POLLOUT` condition not set.

In this case the user will *never* get an ET `POLLOUT` event since `tcp_check_space()` will not call `tcp_new_space()` (as the `SOCK_NOSPACE` bit is not set), which does the real work. **THIS IS AGAINST THE EPOLL ET PROMISE OF DELIVERING AN EVENT WHENEVER THE EVENT ACTUALLY HAPPENS.**

This ET event will be very helpful to implement user level memory management for `mmap+sendfile` zero copy Tx. So typically the application does this

```
void *alloc_sendfile_buf(void)
{
while(!next_free_buffer)
{
/*
* No free buffers (all are dispatched to sendfile and are
* in use). Wait for one or more buffers to become free
* The socket fd is registered with EPOLLET|EPOLLOUT events.
```

[PATCH 2.6.23-rc6 Resending] NETWORKING : Edge Triggered EPOLLOUT events get missed for TCP sockets

```
* EPOLLET enables us to check for SIOCOUTQ only when some
* more space becomes available.
*
* One would expect the ET EPOLLOUT event to be notified
* when TCP space is freed due to some ack coming in.
*/
epoll_wait(...); /* wait for some incoming ack to free some
buffer from the retransmit queue */
ioctl(fd, SIOCOUTQ, &in_outq);
/*
* see if we can mark some more "complete" buffers free
* If it can mark one or more buffer free, it will set
* next_free_buffer to point to the available buffer to use
*/
rehash_free_buffers(in_outq);
}
return next_free_buffer;
}
```

With the SOCK_NOSPACE check in tcp_check_space(), this epoll_wait call will not return, even when the incoming acks free the buffers. Note that this patch assumes that the SOCK_NOSPACE check in tcp_check_space is a trivial optimization which can be safely removed.

Thanx,
Tomar

Signed-off-by: Nagendra Singh Tomar <nagendra_tomar@xxxxxxxxxxx>

```
--- linux-2.6.23-rc6/net/ipv4/tcp_input.c.orig 2007-09-19 13:58:44.000000000 +0530
+++ linux-2.6.23-rc6/net/ipv4/tcp_input.c 2007-09-19 10:17:36.000000000 +0530
@@ -3929,8 +3929,7 @@ static void tcp_check_space(struct sock
{
if (sock_flag(sk, SOCK_QUEUE_SHRUNK)) {
sock_reset_flag(sk, SOCK_QUEUE_SHRUNK);
- if (sk->sk_socket &&
- test_bit(SOCK_NOSPACE, &sk->sk_socket->flags))
+ if (sk->sk_socket)
tcp_new_space(sk);
}
}
```

Yahoo! Answers – Got a question? Someone out there knows the answer. Try it now.

<http://uk.answers.yahoo.com/>

–

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in

[PATCH 2.6.23-rc6 Resending] NETWORKING : Edge Triggered EPOLLOUT events get missed for TCP sockets

[PATCH 2.6.23-rc6 Resending] NETWORKING : Edge Triggered EPOLLOUT events get missed for TCP sockets

the body of a message to majordomo@xxxxxxxxxxxxxxxxx

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>

[PATCH 2.6.23-rc6 Resending] NETWORKING : Edge Triggered EPOLLOUT events get missed for TCP sockets