

[lock validator] drivers/net/8139too.c: deadlock?

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-01/msg10050.html>

- *From:* Ingo Molnar <mingo@xxxxxxx>
 - *Date:* Thu, 26 Jan 2006 23:43:12 +0100
-

the lock validator i'm working on found the following scenario in the rtl8139 driver, which it flagged as a deadlock:

```
----->
NETDEV WATCHDOG: eth0: transmit timed out
eth0: Transmit timeout, status 0d 0000 c07f media 80.
eth0: Tx queue start entry 164281 dirty entry 164277.
eth0: Tx descriptor 0 is 000805ea.
eth0: Tx descriptor 1 is 00080042. (queue head)
eth0: Tx descriptor 2 is 00080042.
eth0: Tx descriptor 3 is 000805ea.
```

```
=====
[ BUG: circular locking deadlock detected! ]
-----
```

```
hackbench/9560 [2] is trying to acquire lock {&tp->rx_lock} at:
[<c033e460>] rtl8139_tx_timeout+0x110/0x1f0
but task is already holding lock {&dev->xmit_lock}, acquired at:
[<c045294b>] dev_watchdog+0x1b/0xc0
which lock already depends on the new lock,
which could lead to circular deadlocks!
```

the dependency chain (in reverse order) is:

```
-> #4 {&dev->xmit_lock}: [<c045294b>] dev_watchdog+0x1b/0xc0
-> #3 {&dev->queue_lock}: [<c0447154>] dev_queue_xmit+0x64/0x290
-> #2 {&((sk)->sk_lock.slock)}: [<c043eb66>] sk_clone+0x66/0x200
-> #1 {&((sk)->sk_lock.slock)}: [<c047a116>] tcp_v4_rcv+0x726/0x9d0
-> #0 {&tp->rx_lock}: [<c033e460>] rtl8139_tx_timeout+0x110/0x1f0
```

other info that might help us debug this:

locks held by hackbench/9560:

```
#0: {net/unix/af_unix.c:&u->readlock} [<c0490e6f>] unix_stream_recvmsg+0xbf/0x4f0
#1: {&dev->xmit_lock} [<c045294b>] dev_watchdog+0x1b/0xc0
```

stack backtrace:

```
[<c010432d>] show_trace+0xd/0x10
[<c0104347>] dump_stack+0x17/0x20
[<c0137d60>] print_circular_bug_tail+0x40/0x50
```

[lock validator] drivers/net/8139too.c: deadlock?

```
[<c013922f>] debug_lock_chain+0x74f/0xd40
[<c013985d>] debug_lock_chain_spin+0x3d/0x60
[<c0266add>] _raw_spin_lock+0x2d/0x90
[<c04d9d18>] _spin_lock+0x8/0x10
[<c033e460>] rtl8139_tx_timeout+0x110/0x1f0
[<c04529e8>] dev_watchdog+0xb8/0xc0
[<c0127615>] run_timer_softirq+0xf5/0x1f0
[<c0122f67>] __do_softirq+0x97/0x130
[<c0105519>] do_softirq+0x69/0x100
=====
[<c0122c19>] irq_exit+0x39/0x50
[<c010f4cc>] smp_apic_timer_interrupt+0x4c/0x50
[<c010393b>] apic_timer_interrupt+0x27/0x2c
[<c0441829>] skb_release_data+0x59/0xa0
[<c0441b43>] kfree_skbmem+0x13/0xe0
[<c0441c58>] __kfree_skb+0x48/0xc0
[<c0490f7c>] unix_stream_recvmsg+0x1cc/0x4f0
[<c043d2d5>] do_sock_read+0x95/0xd0
[<c043d475>] sock_aio_read+0x75/0x80
[<c016499b>] do_sync_read+0xbb/0x110
[<c0164e88>] vfs_read+0x148/0x150
[<c01658bd>] sys_read+0x3d/0x70
[<c0102df7>] sysenter_past_esp+0x54/0x8d
eth0: link up, 100Mbps, full-duplex, lpa 0xC5E1
<-----
```

i'm wondering, is this a genuine deadlock, or a false positive? The dependency chain is quite complex, but looks realistic:

```
-> #4 {&dev->xmit_lock}: [<c045294b>] dev_watchdog+0x1b/0xc0
-> #3 {&dev->queue_lock}: [<c0447154>] dev_queue_xmit+0x64/0x290
-> #2 {&((sk)->sk_lock.slock)}: [<c043eb66>] sk_clone+0x66/0x200
-> #1 {&((sk)->sk_lock.slock)}: [<c047a116>] tcp_v4_rcv+0x726/0x9d0
-> #0 {&tp->rx_lock}: [<c033e460>] rtl8139_tx_timeout+0x110/0x1f0
```

and rtl8139_tx_timeout() is rare enough to not cause real lockups in practice all that often.

explanation of the validator output: the above dependency chain does not mean it actually occurred in one single call sequence – it is a cumulative (and full) dependency graph the validator is maintaining, to prove locking correctness. So it can easily be multiple tasks, at distinct points in time, on different CPUs, which built this dependency chain. The first (#0) and the last (#4) entry is the current locking sequence's fingerprint – so do not understand the above to be an actual locking stack – it cannot possibly occur in this order.

Ingo

–

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxx

[lock validator] drivers/net/8139too.c: deadlock?

[lock validator] drivers/net/8139too.c: deadlock?

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

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

- *Follow-Ups:*

- ◆ **Re: [lock validator] drivers/net/8139too.c: deadlock?**

- ◇ *From:* Francois Romieu

- Prev by Date: **Re: [patch 8/9] slab – Add * mempool slab variants**
- Next by Date: **[1/10] remove ISA legacy functions: drivers/char/toshiba.c**
- Previous by thread: **[2.6 patch: 0/10] remove ISA legacy functions**
- Next by thread: **Re: [lock validator] drivers/net/8139too.c: deadlock?**
- Index(es):
 - ◆ **Date**
 - ◆ **Thread**