

[RFC][PATCH] netconsole: avoid deadlock on printk from driver code

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-08/msg05683.html>

- *From:* Vegard Nossum <vegardno@xxxxxxxxxx>
 - *Date:* Wed, 13 Aug 2008 11:53:24 +0200
-

From b96d44099cc28d084bc1e86aba45465505362298 Mon Sep 17 00:00:00 2001

From: Vegard Nossum <vegard.nossum@xxxxxxxxxx>
Date: Wed, 13 Aug 2008 11:52:21 +0200
Subject: [PATCH] netconsole: avoid deadlock on printk from driver code

I encountered a hard-to-debug deadlock when I pulled out the plug of my RealTek 8139 which was also running netconsole: The driver wants to print a "link down" message. However, this triggers netconsole, which wants to print the message using the same device. Here is a backtrace:

```
[<c05916b6>] _spin_lock_irqsave+0x76/0x90
[<c035b255>] rtl8139_start_xmit+0x65/0x130 <-- spin_lock(&tp->lock)
[<c04c5e28>] netpoll_send_skb+0x158/0x1a0
[<c04c62fb>] netpoll_send_udp+0x1db/0x1f0
[<c037c70c>] write_msg+0x8c/0xc0
[<c0135883>] __call_console_drivers+0x53/0x60
[<c01358db>] _call_console_drivers+0x4b/0x90
[<c0135a25>] release_console_sem+0xc5/0x1f0
[<c0135f0b>] vprintk+0x1ab/0x3e0
[<c013615b>] printk+0x1b/0x20
[<c0349736>] mii_check_media+0x196/0x1e0
[<c03597f4>] rtl_check_media+0x24/0x30
[<c035a0ea>] rtl8139_interrupt+0x42a/0x4a0 <-- spin_lock(&tp->lock)
[<c01716d8>] handle_IRQ_event+0x28/0x70
[<c0172d9b>] handle_fasteoi_irq+0x6b/0xe0
[<c0107128>] do_IRQ+0x48/0xa0
```

The least invasive fix is to detect that we're trying to re-enter the driver code. We provide a netdev_busy() function which can be used to determine whether a deadlock can occur if we try to transmit another packet.

Note that this may lead to lost messages if the driver is active on another CPU while we try to use the same device for netconsole. It would probably be best to set a "lost messages" flag in this case and add it to the stream when the device becomes ready again.

[RFC][PATCH] netconsole: avoid deadlock on printk from driver code

The only extra overhead in non-netconsole code paths is the fact that we need another callback in struct net_device. However, all drivers must be checked for the possibility of a deadlock and implement the ->busy() callback as necessary.

(Patch is untested.)

Cc: Alexey Dobriyan <adobriyan@xxxxxxxxxx>

Cc: Jeff Garzik <jgarzik@xxxxxxxxxx>

Signed-off-by: Vegard Nossum <vegard.nossum@xxxxxxxxxx>

drivers/net/8139too.c | 7 +++++++
drivers/net/netconsole.c | 5 ++++-
include/linux/netdevice.h | 19 ++++++++
3 files changed, 30 insertions(+), 1 deletions(-)

```
diff --git a/drivers/net/8139too.c b/drivers/net/8139too.c
index 8a5b0d2..c1af142 100644
--- a/drivers/net/8139too.c
+++ b/drivers/net/8139too.c
@@ -636,6 +636,7 @@ static void rtl8139_tx_timeout (struct net_device *dev);
static void rtl8139_init_ring (struct net_device *dev);
static int rtl8139_start_xmit (struct sk_buff *skb,
struct net_device *dev);
+static bool rtl8139_busy (struct net_device *dev);
#ifdef CONFIG_NET_POLL_CONTROLLER
static void rtl8139_poll_controller(struct net_device *dev);
#endif
@@ -979,6 +980,7 @@ static int __devinit rtl8139_init_one (struct pci_dev *pdev,
/* The Rtl8139-specific entries in the device structure. */
dev->open = rtl8139_open;
dev->hard_start_xmit = rtl8139_start_xmit;
+ dev->busy = rtl8139_busy;
netif_napi_add(dev, &tp->napi, rtl8139_poll, 64);
dev->stop = rtl8139_close;
dev->get_stats = rtl8139_get_stats;
@@ -1741,6 +1743,11 @@ static int rtl8139_start_xmit (struct sk_buff *skb, struct net_device *dev)
return 0;
}

+static bool rtl8139_busy (struct net_device *dev)
+{
+ struct rtl8139_private *tp = netdev_priv(dev);
+ return spin_is_locked(&tp->lock);
+}

static void rtl8139_tx_interrupt (struct net_device *dev,
struct rtl8139_private *tp,
```

```
diff --git a/drivers/net/netconsole.c b/drivers/net/netconsole.c
index 9681618..b06d7ef 100644
--- a/drivers/net/netconsole.c
```

[RFC][PATCH] netconsole: avoid deadlock on printk from driver code

```
+++ b/drivers/net/netconsole.c
@@ -707,8 +707,11 @@ static void write_msg(struct console *con, const char *msg, unsigned int len)

spin_lock_irqsave(&target_list_lock, flags);
list_for_each_entry(nt, &target_list, list) {
+ struct net_device *dev;
+
netconsole_target_get(nt);
- if (nt->enabled && netif_running(nt->np.dev)) {
+ dev = nt->np.dev;
+ if (nt->enabled && netif_running(dev) && !netdev_busy(dev)) {
/*
* We nest this inside the for-each-target loop above
* so that we're able to get as much logging out to
diff --git a/include/linux/netdevice.h b/include/linux/netdevice.h
index 488c56e..8865a17 100644
--- a/include/linux/netdevice.h
+++ b/include/linux/netdevice.h
@@ -642,6 +642,10 @@ struct net_device
void *priv; /* pointer to private data */
int (*hard_start_xmit) (struct sk_buff *skb,
struct net_device *dev);
+
+ /* Don't use directly; see netdev_busy() below. */
+ bool (*busy) (struct net_device *dev);
+
/* These may be needed for future network-power-down code. */
unsigned long trans_start; /* Time (in jiffies) of last Tx */

@@ -809,6 +813,21 @@ static inline void *netdev_priv(const struct net_device *dev)
& ~NETDEV_ALIGN_CONST);
}

+/**
+ * netdev_busy -r returns false if ->hard_start_xmit() may deadlock
+ * @dev: network device
+ *
+ * More specifically, the driver callback must return true if any of the
+ * locks that are needed for ->hard_start_xmit() are locked. This may look
+ * like a race (because somebody else can take the locks in-between), but
+ * that's fine; we only want to avoid the situation where we can deadlock
+ * against ourselves.
+ */
+static inline bool netdev_busy(struct net_device *dev)
+{
+ return dev->busy && dev->busy(dev);
+}
+
/* Set the sysfs physical device reference for the network logical device
* if set prior to registration will cause a symlink during initialization.
*/
```

[RFC][PATCH] netconsole: avoid deadlock on printk from driver code

--
1.5.5.1

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