

[RFC PATCH 34/35] Add the Xen virtual network device driver.

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-03/msg07361.html>

- *From:* Chris Wright <chrisw@xxxxxxxxxxxxx>
- *Date:* Tue, 21 Mar 2006 22:31:14 -0800

The network device frontend driver allows the kernel to access network devices exported by a virtual machine containing a physical network device driver.

Signed-off-by: Ian Pratt <ian.pratt@xxxxxxxxxxxxx>
 Signed-off-by: Christian Limpach <Christian.Limpach@xxxxxxxxxxxxx>
 Signed-off-by: Chris Wright <chrisw@xxxxxxxxxxxxx>

```

---
drivers/net/Kconfig | 2
drivers/xen/Kconfig.net | 14
drivers/xen/Makefile | 3
drivers/xen/net_driver_util.c | 57 +
drivers/xen/netfront/Makefile | 4
drivers/xen/netfront/netfront.c | 1500 +++++
include/xen/net_driver_util.h | 46 +
7 files changed, 1626 insertions(+)

```

```

--- xen-subarch-2.6.orig/drivers/net/Kconfig
+++ xen-subarch-2.6/drivers/net/Kconfig
@@ -2326,6 +2326,8 @@ source "drivers/atm/Kconfig"

```

```

source "drivers/s390/net/Kconfig"

```

```

+source "drivers/xen/Kconfig.net"
+
config ISERIES_VETH
tristate "iSeries Virtual Ethernet driver support"
depends on PPC_ISERIES
--- xen-subarch-2.6.orig/drivers/xen/Makefile
+++ xen-subarch-2.6/drivers/xen/Makefile
@@ -1,5 +1,8 @@

```

```

+obj-y += net_driver_util.o
obj-y += util.o

```

```

obj-y += console/
obj-y += xenbus/
+

```

[RFC PATCH 34/35] Add the Xen virtual network device driver.

```
+obj-$(CONFIG_XEN_NETDEV_FRONTEND) += netfront/
--- /dev/null
+++ xen-subarch-2.6/drivers/xen/Kconfig.net
@@ -0,0 +1,14 @@
+menu "Xen network device drivers"
+depends on NETDEVICES && XEN
+
+config XEN_NETDEV_FRONTEND
+tristate "Network-device frontend driver"
+depends on XEN
+default y
+help
+The network-device frontend driver allows the kernel to access
+network interfaces within another guest OS. Unless you are building a
+dedicated device-driver domain, or your master control domain
+(domain 0), then you almost certainly want to say Y here.
+
+endmenu
--- /dev/null
+++ xen-subarch-2.6/drivers/xen/net_driver_util.c
@@ -0,0 +1,57 @@
+/******
+ *
+ * Utility functions for Xen network devices.
+ *
+ * Copyright (c) 2005 XenSource Ltd.
+ *
+ * This file may be distributed separately from the Linux kernel, or
+ * incorporated into other software packages, subject to the following
+ * license:
+ *
+ * Permission is hereby granted, free of charge, to any person obtaining a
+ * +}
+
+
+/* ** Driver registration ** */
+
+
+static struct xenbus_device_id netfront_ids[] = {
+ { "vif" },
+ { "" }
+};
+
+
+static struct xenbus_driver netfront = {
+ .name = "vif",
+ .owner = THIS_MODULE,
+ .ids = netfront_ids,
+ .probe = netfront_probe,
+ .remove = netfront_remove,
+ .resume = netfront_resume,
```

```
+ .otherend_changed = backend_changed,
+};
+
+
+static struct notifier_block notifier_inetdev = {
+ .notifier_call = inetdev_notify,
+ .next = NULL,
+ .priority = 0
+};
+
+static int __init netif_init(void)
+{
+ int err = 0;
+
+ if (xen_start_info->flags & SIF_INITDOMAIN)
+ return 0;
+
+ if ((err = xennet_proc_init()) != 0)
+ return err;
+
+ IPRINTK("Initialising virtual ethernet driver.\n");
+
+ (void)register_inetaddr_notifier(&notifier_inetdev);
+
+ return xenbus_register_frontend(&netfront);
+}
+module_init(netif_init);
+
+
+static void netif_exit(void)
+{
+ unregister_inetaddr_notifier(&notifier_inetdev);
+
+ return xenbus_unregister_driver(&netfront);
+}
+module_exit(netif_exit);
+
+MODULE_LICENSE("Dual BSD/GPL");
+
+
+/* ** /proc **/
+
+
+#if defined(CONFIG_PROC_FS) && defined(XEN_XENNET_PROC_INTERFACE)
+
+#define TARGET_MIN 0UL
+#define TARGET_MAX 1UL
+#define TARGET_CUR 2UL
+
+static int xennet_proc_read(
+ char *page, char **start, off_t off, int count, int *eof, void *data)
```

```

+{
+ struct net_device *dev =
+ (struct net_device *)((unsigned long)data & ~3UL);
+ struct netfront_info *np = netdev_priv(dev);
+ int len = 0, which_target = (long)data & 3;
+
+ switch (which_target) {
+ case TARGET_MIN:
+ len = sprintf(page, "%d\n", np->rx_min_target);
+ break;
+ case TARGET_MAX:
+ len = sprintf(page, "%d\n", np->rx_max_target);
+ break;
+ case TARGET_CUR:
+ len = sprintf(page, "%d\n", np->rx_target);
+ break;
+ }
+
+ *eof = 1;
+ return len;
+}
+
+static int xennet_proc_write(
+ struct file *file, const char __user *buffer,
+ unsigned long count, void *data)
+{
+ struct net_device *dev =
+ (struct net_device *)((unsigned long)data & ~3UL);
+ struct netfront_info *np = netdev_priv(dev);
+ int which_target = (long)data & 3;
+ char string[64];
+ long target;
+
+ if (!capable(CAP_SYS_ADMIN))
+ return -EPERM;
+
+ if (count <= 1)
+ return -EBADMSG; /* runt */
+ if (count > sizeof(string))
+ return -EFBIG; /* too long */
+
+ if (copy_from_user(string, buffer, count))
+ return -EFAULT;
+ string[sizeof(string)-1] = '\0';
+
+ target = simple_strtol(string, NULL, 10);
+ if (target < RX_MIN_TARGET)
+ target = RX_MIN_TARGET;
+ if (target > RX_MAX_TARGET)
+ target = RX_MAX_TARGET;
+
+

```

[RFC PATCH 34/35] Add the Xen virtual network device driver.

```
+ spin_lock(&np->rx_lock);
+
+ switch (which_target) {
+ case TARGET_MIN:
+ if (target > np->rx_max_target)
+ np->rx_max_target = target;
+ np->rx_min_target = target;
+ if (target > np->rx_target)
+ np->rx_target = target;
+ break;
+ case TARGET_MAX:
+ if (target < np->rx_min_target)
+ np->rx_min_target = target;
+ np->rx_max_target = target;
+ if (target < np->rx_target)
+ np->rx_target = target;
+ break;
+ case TARGET_CUR:
+ break;
+ }
+
+ network_alloc_rx_buffers(dev);
+
+ spin_unlock(&np->rx_lock);
+
+ return count;
+}
+
+static int xennet_proc_init(void)
+{
+ if (proc_mkdir("xen/net", NULL) == NULL)
+ return -ENOMEM;
+ return 0;
+}
+
+static int xennet_proc_addif(str
```