NAME

xnb - Xen Paravirtualized Backend Ethernet Driver

SYNOPSIS

To compile this driver into the kernel, place the following lines in your kernel configuration file:

```
options XENHVM
device xenpci
```

DESCRIPTION

The **xnb** driver provides the back half of a paravirtualized xen(4) network connection. The netback and netfront drivers appear to their respective operating systems as Ethernet devices linked by a crossover cable. Typically, **xnb** will run on Domain 0 and the netfront driver will run on a guest domain. However, it is also possible to run **xnb** on a guest domain. It may be bridged or routed to provide the netfront domain access to other guest domains or to a physical network.

In most respects, the **xnb** device appears to the OS as any other Ethernet device. It can be configured at runtime entirely with ifconfig(8). In particular, it supports MAC changing, arbitrary MTU sizes, checksum offload for IP, UDP, and TCP for both receive and transmit, and TSO. However, see *CAVEATS* before enabling txcsum, rxcsum, or tso.

SYSCTL VARIABLES

The following read-only variables are available via sysctl(8):

dev.xnb.%d.dump_rings
: Displays information about the ring buffers used to pass requests between the netfront and netback. Mostly useful for debugging, but can also be used to get traffic statistics.

dev.xnb.%d.unit_test_results
: Runs a builtin suite of unit tests and displays the results. Does not affect the operation of the driver in any way. Note that the test suite simulates error conditions; this will result in error messages being printed to the system log.

SEE ALSO

arp(4), netintro(4), ng_ether(4), xen(4), ifconfig(8)

HISTORY

The **xnb** device driver first appeared in FreeBSD 10.0.

AUTHORS

The **xnb** driver was written by Alan Somers <*asomers@FreeBSD.org*> and John Suykerbuyk.

CAVEATS

Packets sent through Xennet pass over shared memory, so the protocol includes no form of link-layer checksum or CRC. Furthermore, Xennet drivers always report to their hosts that they support receive and transmit checksum offloading. They "offload" the checksum calculation by simply skipping it. That works fine for packets that are exchanged between two domains on the same machine. However, when a Xennet interface is bridged to a physical interface, a correct checksum must be attached to any packets bound for that physical interface. Currently, FreeBSD lacks any mechanism for an Ethernet device to inform the OS that newly received packets are valid even though their checksums are not. So if the netfront driver is configured to offload checksum calculations, it will pass non-checksumed packets to **xnb**, which must then calculate the checksum in software before passing the packet to the OS.

For this reason, it is recommended that if **xnb** is bridged to a physical interface, then transmit checksum offloading should be disabled on the netfront. The Xennet protocol does not have any mechanism for the netback to request the netfront to do this; the operator must do it manually.

BUGS

The **xnb** driver does not properly checksum UDP datagrams that span more than one Ethernet frame. Nor does it correctly checksum IPv6 packets. To workaround that bug, disable transmit checksum offloading on the netfront driver.