Linux Base Driver for 10 Gigabit Intel(R) Ethernet Network Connection

October 1, 2018

Contents

In This Release

This file describes the ixgb Linux Base Driver for the 10 Gigabit Intel(R) Network Connection. This driver includes support for Itanium(R)2-based systems.

For questions related to hardware requirements, refer to the documentation supplied with your 10 Gigabit adapter. All hardware requirements listed apply to use with Linux.

The following features are available in this kernel:

- Native VLANs
- Channel Bonding (teaming)
- SNMP

Channel Bonding documentation can be found in the Linux kernel source: /Documentation/networking/bonding.rst

The driver information previously displayed in the /proc filesystem is not supported in this release. Alternatively, you can use ethtool (version 1.6 or later), lspci, and iproute2 to obtain the same information.

Instructions on updating ethtool can be found in the section "Additional Configurations" later in this document.

Identifying Your Adapter

The following Intel network adapters are compatible with the drivers in this release:

Controller	Adapter Name	Physical Layer
82597EX	Intel(R) PRO/10GbE LR/SR/CX4 Server Adapters	10G Base-LR (fiber) 10G Base-SR (fiber) 10G Base-CX4 (copper)

For more information on how to identify your adapter, go to the Adapter & Driver ID Guide at:

https://support.intel.com

Command Line Parameters

If the driver is built as a module, the following optional parameters are used by entering them on the command line with the modprobe command using this syntax:

```
modprobe ixgb [<option>=<VAL1>,<VAL2>,...]
```

For example, with two 10GbE PCI adapters, entering:

```
modprobe ixgb TxDescriptors=80,128
```

loads the ixgb driver with 80 TX resources for the first adapter and 128 TX resources for the second adapter.

The default value for each parameter is generally the recommended setting, unless otherwise noted.

Copybreak

Valid Range:	0-XXXX
Default Value:	256

This is the maximum size of packet that is copied to a new buffer on receive.

Debug

Valid Range: 0-16 (0=none,...,16=all)

Default Value: 0

This parameter adjusts the level of debug messages displayed in the system logs.

FlowControl

Valid Range: 0-3 (0=none, 1=Rx only, 2=Tx only, 3=Rx&Tx)

Default Value: 1 if no EEPROM, otherwise read from EEPROM

This parameter controls the automatic generation(Tx) and response(Rx) to Ethernet PAUSE frames. There are hardware bugs associated with enabling Tx flow control so beware.

RxDescriptors

Valid Range: 64-4096

Default Value: 1024

This value is the number of receive descriptors allocated by the driver. Increasing this value allows the driver to buffer more incoming packets. Each descriptor is 16 bytes. A receive buffer is also allocated for each descriptor and can be either 2048, 4056, 8192, or 16384 bytes, depending on the MTU setting. When the MTU size is 1500 or less, the receive buffer size is 2048 bytes. When the MTU is greater than 1500 the receive buffer size will be either 4056, 8192, or 16384 bytes. The maximum MTU size is 16114.

TxDescriptors

Valid Range: 64-4096

Default Value: 256

This value is the number of transmit descriptors allocated by the driver. Increasing this value allows the driver to queue more transmits. Each descriptor is 16 bytes.

RxIntDelay

Valid Range: 0-65535 (0=off)

Default Value: 72

This value delays the generation of receive interrupts in units of 0.8192 microseconds. Receive interrupt reduction can improve CPU efficiency if properly tuned for specific network traffic. Increasing this value adds extra latency to frame reception and can end up decreasing the throughput of TCP traffic. If the system is reporting dropped receives, this value may be set too high, causing the driver to run out of available receive descriptors.

TxIntDelay

Valid Range: 0-65535 (0=off)

Default Value: 32

This value delays the generation of transmit interrupts in units of 0.8192 microseconds. Transmit interrupt reduction can improve CPU efficiency if properly tuned for specific network traffic. Increasing this value adds extra latency to frame transmission and can end up decreasing the throughput of TCP traffic. If this value is set too high, it will cause the driver to run out of available transmit descriptors.

XsumRX

Valid Range: 0-1

Default Value: 1

A value of '1' indicates that the driver should enable IP checksum offload for received packets (both UDP and TCP) to the adapter hardware.

RxFCHighThresh

Valid Range: 1,536-262,136 (0x600 - 0x3FFF8, 8 byte granularity)

Default Value: 196,608 (0x30000)

Receive Flow control high threshold (when we send a pause frame)

RxFCLowThresh

Valid Range: 64-262,136 (0x40 - 0x3FFF8, 8 byte granularity)

Default Value: 163,840 (0x28000)

Receive Flow control low threshold (when we send a resume frame)

FCReqTimeout

Valid Range: 1-65535

Default Value: 65535

Flow control request timeout (how long to pause the link partner's tx)

IntDelayEnable

Value Range: 0,1

Default Value: 1

Interrupt Delay, 0 disables transmit interrupt delay and 1 enables it.

Improving Performance

With the 10 Gigabit server adapters, the default Linux configuration will very likely limit the total available throughput artificially. There is a set of configuration changes that, when applied together, will increase the ability of Linux to transmit and receive data. The following enhancements were originally acquired from settings published at https://www.spec.org/web99/ for various submitted results using Linux.

NOTE:

> These changes are only suggestions, and serve as a starting point for tuning your network performance.

The changes are made in three major ways, listed in order of greatest effect:

- Use ip link to modify the mtu (maximum transmission unit) and the txqueuelen parameter.
- Use sysctl to modify /proc parameters (essentially kernel tuning)
- Use setpci to modify the MMRBC field in PCI-X configuration space to increase transmit burst lengths on the bus.

NOTE:

> setpci modifies the adapter's configuration registers to allow it to read up to 4k bytes at a time (for transmits). However, for some systems the behavior after modifying this register may be undefined (possibly errors of some kind). A power-cycle, hard reset or explicitly setting the e6 register back to 22 (setpci -d 8086:1a48 e6.b=22) may be required to get back to a stable configuration.

- COPY these lines and paste them into ixgb_perf.sh:

```
#!/bin/bash
echo "configuring network performance , edit this file to change the interface
or device ID of 10GbE card"
# set mmrbc to 4k reads, modify only Intel 10GbE device IDs
# replace 1a48 with appropriate 10GbE device's ID installed on the system,
# if needed.
setpci -d 8086:1a48 e6.b=2e
# set the MTU (max transmission unit) - it requires your switch and clients
# to change as well.
# set the txqueuelen
# your ixgb adapter should be loaded as eth1 for this to work, change if needed
ip li set dev eth1 mtu 9000 txqueuelen 1000 up
# call the sysctl utility to modify /proc/sys entries
sysctl -p ./sysctl_ixgb.conf
```

- COPY these lines and paste them into sysctl_ixgb.conf:

```
# some of the defaults may be different for your kernel
# call this file with sysctl -p <this file>
# these are just suggested values that worked well to increase throughput in
# several network benchmark tests, your mileage may vary

### IPV4 specific settings
# turn TCP timestamp support off, default 1, reduces CPU use
net.ipv4.tcp_timestamps = 0
# turn SACK support off, default on
# on systems with a VERY fast bus -> memory interface this is the big gainer
net.ipv4.tcp_sack = 0
# set min/default/max TCP read buffer, default 4096 87380 174760
```

```
net.ipv4.tcp_rmem = 10000000 10000000 10000000
# set min/pressure/max TCP write buffer, default 4096 16384 131072
net.ipv4.tcp_wmem = 10000000 10000000 10000000
# set min/pressure/max TCP buffer space, default 31744 32256 32768
net.ipv4.tcp_mem = 10000000 10000000 10000000

### CORE settings (mostly for socket and UDP effect)
# set maximum receive socket buffer size, default 131071
net.core.rmem_max = 524287
# set maximum send socket buffer size, default 131071
net.core.wmem_max = 524287
# set default receive socket buffer size, default 65535
net.core.rmem_default = 524287
# set default send socket buffer size, default 65535
net.core.wmem_default = 524287
# set maximum amount of option memory buffers, default 10240
net.core.optmem_max = 524287
# set number of unprocessed input packets before kernel starts dropping them; default 300
net.core.netdev_max_backlog = 300000
```

Edit the ixgb_perf.sh script if necessary to change eth1 to whatever interface your ixgb driver is using and/or replace '1a48' with appropriate 10GbE device's ID installed on the system.

NOTE:

Unless these scripts are added to the boot process, these changes will only last only until the next system reboot.

Resolving Slow UDP Traffic

If your server does not seem to be able to receive UDP traffic as fast as it can receive TCP traffic, it could be because Linux, by default, does not set the network stack buffers as large as they need to be to support high UDP transfer rates. One way to alleviate this problem is to allow more memory to be used by the IP stack to store incoming data.

For instance, use the commands:

```
sysctl -w net.core.rmem_max=262143
```

and:

```
sysctl -w net.core.rmem_default=262143
```

to increase the read buffer memory max and default to 262143 (256k - 1) from defaults of max=131071 (128k - 1) and default=65535 (64k - 1). These variables will increase the amount of memory used by the network stack for receives, and can be increased significantly more if necessary for your application.

Additional Configurations

Configuring the Driver on Different Distributions

Configuring a network driver to load properly when the system is started is distribution dependent. Typically, the configuration process involves adding an alias line to /etc/modprobe.conf as well as editing other system startup scripts and/or configuration files. Many popular Linux distributions ship with tools to make these changes for you. To learn the proper way to configure a network device for your system, refer to your distribution documentation. If during this process you are asked for the driver or module name, the name for the Linux Base Driver for the Intel 10GbE Family of Adapters is ixgb.

Viewing Link Messages

Link messages will not be displayed to the console if the distribution is restricting system messages. In order to see network driver link messages on your console, set dmesg to eight by entering the following:

```
dmesg -n 8
```

NOTE: This setting is not saved across reboots.

Jumbo Frames

The driver supports Jumbo Frames for all adapters. Jumbo Frames support is enabled by changing the MTU to a value larger than the default of 1500. The maximum value for the MTU is 16114. Use the ip command to increase the MTU size. For example:

```
ip li set dev ethx mtu 9000
```

The maximum MTU setting for Jumbo Frames is 16114. This value coincides with the maximum Jumbo Frames size of 16128.

Ethtool

The driver utilizes the ethtool interface for driver configuration and diagnostics, as well as displaying statistical information. The ethtool version 1.6 or later is required for this functionality.

The latest release of ethtool can be found from https://www.kernel.org/pub/software/network/ethtool/

NOTE:

> The ethtool version 1.6 only supports a limited set of ethtool options. Support for a more complete ethtool feature set can be enabled by upgrading to the latest version.

NAPI

NAPI (Rx polling mode) is supported in the ixgb driver.

See https://wiki.linuxfoundation.org/networking/napi for more information on NAPI.

Known Issues/Troubleshooting

NOTE:

> After installing the driver, if your Intel Network Connection is not working, verify in the "In This Release" section of the readme that you have installed the correct driver.

Cable Interoperability Issue with Fujitsu XENPAK Module in SmartBits Chassis

Excessive CRC errors may be observed if the Intel(R) PRO/10GbE CX4 Server adapter is connected to a Fujitsu XENPAK CX4 module in a SmartBits chassis using 15 m/24AWG cable assemblies manufactured by Fujitsu or Leoni. The CRC errors may be received either by the Intel(R) PRO/10GbE CX4 Server adapter or the SmartBits. If this situation occurs using a different cable assembly may resolve the issue.

Cable Interoperability Issues with HP Procurve 3400cl Switch Port

Excessive CRC errors may be observed if the Intel(R) PRO/10GbE CX4 Server adapter is connected to an HP Procurve 3400cl switch port using short cables (1 m or shorter). If this situation occurs, using a longer cable may resolve the issue.

Excessive CRC errors may be observed using Fujitsu 24AWG cable assemblies that Are 10 m or longer or where using a Leoni 15 m/24AWG cable assembly. The CRC errors may be received either by the CX4 Server adapter or at the switch. If this situation occurs, using a different cable assembly may resolve the issue.

Jumbo Frames System Requirement

Memory allocation failures have been observed on Linux systems with 64 MB of RAM or less that are running Jumbo Frames. If you are using Jumbo Frames, your system may require more than the advertised minimum requirement of 64 MB of system memory.

Performance Degradation with Jumbo Frames

Degradation in throughput performance may be observed in some Jumbo frames environments. If this is observed, increasing the application's socket buffer size and/or increasing the /proc/sys/net/ipv4/tcp_*mem entry values may help. See the specific application manual and /usr/src/linux*/Documentation/ networking/ip-sysctl.txt for more details.

Allocating Rx Buffers when Using Jumbo Frames

Allocating Rx buffers when using Jumbo Frames on 2.6.x kernels may fail if the available memory is heavily fragmented. This issue may be seen with PCI-X adapters or with packet split disabled. This can be reduced or eliminated by changing the amount of available memory for receive buffer allocation, by increasing /proc/sys/vm/min_free_kbytes.

Multiple Interfaces on Same Ethernet Broadcast Network

Due to the default ARP behavior on Linux, it is not possible to have one system on two IP networks in the same Ethernet broadcast domain (non-partitioned switch) behave as expected. All Ethernet interfaces will respond to IP traffic for any IP address assigned to the system. This results in unbalanced receive traffic.

If you have multiple interfaces in a server, do either of the following:

- Turn on ARP filtering by entering:

  ```
  echo 1 > /proc/sys/net/ipv4/conf/all/arp_filter
  ```

- Install the interfaces in separate broadcast domains - either in different switches or in a switch partitioned to VLANs.

UDP Stress Test Dropped Packet Issue

Under small packets UDP stress test with 10GbE driver, the Linux system may drop UDP packets due to the fullness of socket buffers. You may want to change the driver's Flow Control variables to the minimum value for controlling packet reception.

Tx Hangs Possible Under Stress

Under stress conditions, if TX hangs occur, turning off TSO "ethtool -K eth0 tso off" may resolve the problem.

Support

For general information, go to the Intel support website at:

https://www.intel.com/support/

or the Intel Wired Networking project hosted by Sourceforge at:

https://sourceforge.net/projects/e1000

If an issue is identified with the released source code on a supported kernel with a supported adapter, email the specific information related to the issue to e1000-devel@lists.sf.net