Intel i40e Device Driver Version 2.7.26 Installation on RedHawk OS

Release Notes

November 2,2021

concurrent
REAL-TIME

1. Introduction:

本書は、RedHawk に移植済の Intel i40e version 2.7.26 デバイスドライバ・リリースノートである。.

2. Requirements:

このデバイスドライバをインストールする OS は、RedHawk 64 bits を想定している。
このドライバーは、以下に基づくデバイスと互換性があります。

* Intel(R) Ethernet Controller X710
* Intel(R) Ethernet Controller XL710
* Intel(R) Ethernet Network Connection X722
* Intel(R) Ethernet Controller XXV710
* Intel(R) Ethernet Controller V710

3. Installation:

本パッケージは、rpm バイナリで提供される。
以下の手順で、rpm ファイルをインストールする。
ドライバパッケージは RedHawk 各カーネルフレーバー用の、ドライバディレクトリ下にインストールされ、必要に応じて TRACE,DEBUG,STANDARD,KDUMP の initramfs カーネルモジュールが自動生成される。
このディレクトリは、RedHawk7.x では、
“/lib/modules`uname –r`/updates/drivers/net/ethernet/intel/i40e/” である。

以下にインストール例を示す。

```
# mount /dev/dvd /mnt
# cd /mnt

# rpm -ivh RPMS/x86_64/i40e-2.7.26-1.RedHawk-6.0.2.x86_64.rpm
準備中...                ########################################### [100%]
   1:i40e                ########################################### [100%]
original pci.ids saved in /usr/local/share/i40e
patching file Kconfig
Original driver saved in /usr/local/src/i40e-2.7.26
Installing i40e-2.7.26 drivers succeeded!
```

なお、README などの、ファイルは、/usr/share/doc/i40e-2.7.26 ディレクトリに、バックアップソースコードは/usr/local/src/i40e-2.7.26 ディレクトリにインストールされる。
以下に、rpm ファイルで、供給されるファイル例を示す。

```
# rpm -qpl i40e-2.7.26-1.RedHawk-6.0.2.x86_64.rpm
/lib/modules/2.6.36.4-RedHawk-6.0.2-debug/updates/drivers/net/ethernet/intel/i40e/i40e.ko.new
/lib/modules/2.6.36.4-RedHawk-6.0.2-kdump/updates/drivers/net/ethernet/intel/i40e/i40e.ko.new
/lib/modules/2.6.36.4-RedHawk-6.0.2-trace/updates/drivers/net/ethernet/intel/i40e/i40e.ko.new
/lib/modules/2.6.36.4-RedHawk-6.0.2/updates/drivers/net/ethernet/intel/i40e/i40e.ko.new
/usr/local/src/i40e-2.7.26/i40e-2.7.26.src.tar.gz
/usr/share/doc/i40e-2.7.26
/usr/share/doc/i40e-2.7.26/COPYING
/usr/share/doc/i40e-2.7.26/README
/usr/share/doc/i40e-2.7.26/file.list
/usr/share/doc/i40e-2.7.26/pci.updates
/usr/share/man/man7/i40e.7.gz
```

4. Building driver on a currently running RedHawk kernel

インストール後、カーネルモジュールおよび、ソースコードは、新しい版に入れ替えられているため、実際に動作しているデバイスドライバを、以下のコマンドで確認できる。

```
# modinfo i40e
filename:       /lib/modules/2.6.36.4-RedHawk-6.0.2-trace/updates/drivers/net/ethernet/intel/i40e/i40e.ko
version:        2.7.26
license:        GPL
description:    Intel(R) 40-10 Gigabit Ethernet Connection Network Driver
author:         Intel Corporation, <e1000-devel@lists.sourceforge.net>
srcversion:     E1C9A44F9765A0040BA5164
alias:          pci:v00008086d0000158Bsv*sd*bc*sc*i*
alias:          pci:v00008086d0000158Asv*sd*bc*sc*i*
alias:          pci:v00008086d000037D3sv*sd*bc*sc*i*
alias:          pci:v00008086d000037D2sv*sd*bc*sc*i*
alias:          pci:v00008086d000037D1sv*sd*bc*sc*i*
alias:          pci:v00008086d000037D0sv*sd*bc*sc*i*
alias:          pci:v00008086d000037CFsv*sd*bc*sc*i*
alias:          pci:v00008086d000037CEsv*sd*bc*sc*i*
alias:          pci:v00008086d00001588sv*sd*bc*sc*i*
alias:          pci:v00008086d00001587sv*sd*bc*sc*i*
alias:          pci:v00008086d000015FFsv*sd*bc*sc*i*
alias:          pci:v00008086d00001589sv*sd*bc*sc*i*
alias:          pci:v00008086d00001586sv*sd*bc*sc*i*
alias:          pci:v00008086d00001585sv*sd*bc*sc*i*
alias:          pci:v00008086d00001584sv*sd*bc*sc*i*
alias:          pci:v00008086d00001583sv*sd*bc*sc*i*
alias:          pci:v00008086d00001581sv*sd*bc*sc*i*
alias:          pci:v00008086d00001580sv*sd*bc*sc*i*
alias:          pci:v00008086d00001574sv*sd*bc*sc*i*
alias:          pci:v00008086d00001572sv*sd*bc*sc*i*
depends:
vermagic:       2.6.36.4-RedHawk-6.0.2-trace SMP preempt mod_unload modversions
parm:           max_vfs:Number of Virtual Functions: 0 = disable (default), 1-128 = enable this many VFs (array of int)
parm:           debug:Debug level (0=none,...,16=all) (int)
```

また、通常の下記マニュアル手順で、カーネルの再構築を行うことも可能である。

```
# cd  /lib/modules/`uname -r`/build
# ./ccur-config –c –n
# make –C `pwd` SUBDIRS=`pwd`/drivers/net/ethernet/intel/i40e REDHAWKFLAVOR=`cat /proc/ccur/flavor` modules
# make –C `pwd` SUBDIRS=`pwd`/drivers/net/ethernet/intel/i40e REDHAWKFLAVOR=`cat /proc/ccur/flavor` modules_install
```

但し、このマニュアル手順による方法では、オリジナルカーネルモジュールのファイルの上書きを行うため、本 rpm パッケージとモジュールをインストールする位置が異なる。このため、整合性に注意しなければならない。
なお、システムを再構築後、下記手順で initramfs カーネルモジュールを組み込むことができる。

```
# dracut --add-drivers " dca hwmon ixgbe" -f /boot/initramfs-`uname -r`.img  `uname -r`
```

正常に組み込むことが出来ると、下記のメッセージがコンソールに表示される。

```
i40e: Intel(R) 40-10 Gigabit Ethernet Connection Network Driver - version 2.7.26
i40e: Copyright(c) 2013 - 2018 Intel Corporation.
```

5. Remove driver

rpm パッケージを削除するためには、以下のコマンドを使用する。
インストール時に作成されたバックアップファイルを使用し、元の状態に戻される。

```
# rpm -e i40e
patching file Kconfig
```

6. README

i40e Linux* Base Driver for the Intel(R) Ethernet Controller 700 Series
==========

November 29, 2018

Contents
========

==========

Important Notes

TC0 must be enabled when setting up DCB on a switch

The kernel assumes that TC0 is available, and will disable Priority Flow
Control (PFC) on the device if TC0 is not available. To fix this, ensure TC0 is
enabled when setting up DCB on your switch.

Enabling a VF link if the port is disconnected

If the physical function (PF) link is down, you can force link up (from the
host PF) on any virtual functions (VF) bound to the PF. Note that this requires
kernel support (Redhat kernel 3.10.0-327 or newer, upstream kernel 3.11.0 or
newer, and associated iproute2 user space support). If the following command
does not work, it may not be supported by your system. The following command
forces link up on VF 0 bound to PF eth0:
  ip link set eth0 vf 0 state enable

Do not unload port driver if VF with active VM is bound to it
---
Do not unload a port's driver if a Virtual Function (VF) with an active Virtual
Machine (VM) is bound to it. Doing so will cause the port to appear to hang.
Once the VM shuts down, or otherwise releases the VF, the command will complete.

Configuring SR-IOV for improved network security

In a virtualized environment, on Intel(R) Ethernet Server Adapters that support
SR-IOV, the virtual function (VF) may be subject to malicious behavior.
Software-generated layer two frames, like IEEE 802.3x (link flow control), IEEE

802.1Qbb (priority based flow-control), and others of this type, are not expected and can throttle traffic between the host and the virtual switch, reducing performance. To resolve this issue, and to ensure isolation from unintended traffic streams, configure all SR-IOV enabled ports for VLAN tagging from the administrative interface on the PF. This configuration allows unexpected, and potentially malicious, frames to be dropped.

Overview

This driver supports kernel versions 2.6.32 and newer.

Driver information can be obtained using ethtool, lspci, and ifconfig. Instructions on updating ethtool can be found in the section Additional Configurations later in this document.

This driver is only supported as a loadable module at this time. Intel is not supplying patches against the kernel source to allow for static linking of the drivers.

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

Adapter teaming is implemented using the native Linux Channel bonding module. This is included in supported Linux kernels. Channel Bonding documentation can be found in the Linux kernel source:
/documentation/networking/bonding.txt

This driver supports XDP (Express Data Path) on kernel 4.14 and later. Note that XDP is blocked for frame sizes larger than 3KB.

The driver information previously displayed in the /proc file system is not supported in this release.

NOTE: 1 Gb devices based on the Intel(R) Ethernet Network Connection X722 do not support the following features:
 * Data Center Bridging (DCB)
 * QOS
 * VMQ
 * SR-IOV
 * Task Encapsulation offload (VXLAN, NVGRE)
 * Energy Efficient Ethernet (EEE)
 * Auto-media detect

Identifying Your Adapter

The driver in this release is compatible with devices based on the following:
 * Intel(R) Ethernet Controller X710
 * Intel(R) Ethernet Controller XL710
 * Intel(R) Ethernet Network Connection X722
 * Intel(R) Ethernet Controller XXV710

For the best performance, make sure the latest NVM/FW is installed on your device and that you are using the newest drivers.

For information on how to identify your adapter, and for the latest NVM/FW images and Intel network drivers, refer to the Intel Support website:
http://www.intel.com/support

SFP+ and QSFP+ Devices:

For information about supported media, refer to this document:
http://www.intel.com/content/dam/www/public/us/en/documents/release-notes/xl710-ethernet-controller-feature-matrix.pdf
NOTE: Some adapters based on the Intel(R) Ethernet Controller 700 Series only support Intel Ethernet Optics modules. On these adapters, other modules are not supported and will not function.

NOTE: For connections based on Intel(R) Ethernet Controller 700 Series, support is dependent on your system board. Please see your vendor for details.

NOTE:In all cases Intel recommends using Intel Ethernet Optics; other modules may function but are not validated by Intel. Contact Intel for supported media types.

NOTE: In systems that do not have adequate airflow to cool the adapter and optical modules, you must use high temperature optical modules.

==========

Building and Installation

To build a binary RPM* package of this driver, run 'rpmbuild -tb i40e-<x.x.x>.tar.gz', where <x.x.x> is the version number for the driver tar file.

Note: For the build to work properly, the currently running kernel MUST match the version and configuration of the installed kernel sources. If you have just recompiled the kernel reboot the system before building.

Note: RPM functionality has only been tested in Red Hat distributions.

1. Move the base driver tar file to the directory of your choice. For example, use '/home/username/i40e' or '/usr/local/src/i40e'.

2. Untar/unzip the archive, where <x.x.x> is the version number for the driver tar file:
   tar zxf i40e-<x.x.x>.tar.gz

3. Change to the driver src directory, where <x.x.x> is the version number for the driver tar:
   cd i40e-<x.x.x>/src/

4. Compile the driver module:
   # make install
   The binary will be installed as:
   /lib/modules/<KERNEL VERSION>/updates/drivers/net/ethernet/intel/i40e/i40e.ko

   The install location listed above is the default location. This may differ for various Linux distributions.
   NOTE: To compile the driver on some kernel/arch combinations, a package with the development version of libelf (e.g. libelf-dev, libelf-devel, elfutilsl-libelf-devel) may need to be installed.

NOTE: To gather and display additional statistics, use the
I40E_ADD_PROBES pre-processor macro:
#make CFLAGS_EXTRA=-DI40E_ADD_PROBES
Please note that this additional statistics gathering can impact
performance.

5. Load the module using the modprobe command:
   modprobe <i40e> [parameter=port1_value,port2_value]

   Make sure that any older i40e drivers are removed from the kernel before
   loading the new module:
   rmmod i40e; modprobe i40e

6. Assign an IP address to the interface by entering the following,
   where ethX is the interface name that was shown in dmesg after modprobe:

   ip address add <IP_address>/<netmask bits> dev ethX

7. Verify that the interface works. Enter the following, where IP_address
   is the IP address for another machine on the same subnet as the interface
   that is being tested:
   ping <IP_address>

Note: For certain distributions like (but not limited to) RedHat Enterprise
Linux 7 and Ubuntu, once the driver is installed the initrd/initramfs file may
need to be updated to prevent the OS loading old versions of the i40e driver.
The dracut utility may be used on RedHat distributions:
        # dracut --force
   For Ubuntu:
        # update-initramfs -u

==========

Command Line Parameters

In general, ethtool and other OS specific commands are used to configure user
changeable parameters after the driver is loaded. The i40e driver only supports
the max_vfs kernel parameter on older kernels that do not have the standard
sysfs interface. The only other module parameter supported is the debug
parameter that can control the default logging verbosity of the driver.

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 i40e [<option>=<VAL1>]

There needs to be a <VAL#> for each network port in the system supported by
this driver. The values will be applied to each instance, in function order.
For example:
modprobe i40e max_vfs=7

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

max_vfs

This parameter adds support for SR-IOV. It causes the driver to spawn up to

max_vfs worth of virtual functions.
Valid Range:
1-32 (Intel Ethernet Controller X710 based devices)
1-64 (Intel Ethernet Controller XXV710/XL710 based devices)

NOTE: This parameter is only used on kernel 3.7.x and below. On kernel 3.8.x and above, use sysfs to enable VFs. Also, for Red Hat distributions, this parameter is only used on version 6.6 and older. For version 6.7 and newer, use sysfs. For example:
#echo $num_vf_enabled > /sys/class/net/$dev/device/sriov_numvfs //enable VFs
#echo 0 > /sys/class/net/$dev/device/sriov_numvfs //disable VFs

The parameters for the driver are referenced by position. Thus, if you have a dual port adapter, or more than one adapter in your system, and want N virtual functions per port, you must specify a number for each port with each parameter separated by a comma. For example:

```
modprobe i40e max_vfs=4
```

This will spawn 4 VFs on the first port.

```
modprobe i40e max_vfs=2,4
```

This will spawn 2 VFs on the first port and 4 VFs on the second port.

NOTE: Caution must be used in loading the driver with these parameters. Depending on your system configuration, number of slots, etc., it is impossible to predict in all cases where the positions would be on the command line.

NOTE: Neither the device nor the driver control how VFs are mapped into config space. Bus layout will vary by operating system. On operating systems that support it, you can check sysfs to find the mapping.

Some hardware configurations support fewer SR-IOV instances, as the whole Intel Ethernet Controller XL710 (all functions) is limited to 128 SR-IOV interfaces in total.

NOTE: When SR-IOV mode is enabled, hardware VLAN filtering and VLAN tag stripping/insertion will remain enabled. Please remove the old VLAN filter before the new VLAN filter is added. For example,
ip link set eth0 vf 0 vlan 100 // set vlan 100 for VF 0
ip link set eth0 vf 0 vlan 0 // Delete vlan 100
ip link set eth0 vf 0 vlan 200 // set a new vlan 200 for VF 0

Configuring SR-IOV for improved network security

In a virtualized environment, on Intel(R) Ethernet Server Adapters that support SR-IOV, the virtual function (VF) may be subject to malicious behavior. Software-generated layer two frames, like IEEE 802.3x (link flow control), IEEE 802.1Qbb (priority based flow-control), and others of this type, are not expected and can throttle traffic between the host and the virtual switch, reducing performance. To resolve this issue, and to ensure isolation from unintended traffic streams, configure all SR-IOV enabled ports for VLAN tagging from the administrative interface on the PF. This configuration allows unexpected, and potentially malicious, frames to be dropped.

Configuring VLAN tagging on SR-IOV enabled adapter ports

To configure VLAN tagging for the ports on an SR-IOV enabled adapter, use the following command. The VLAN configuration should be done before the VF driver is loaded or the VM is booted.

$ ip link set dev <PF netdev id> vf <id> vlan <vlan id>

For example, the following instructions will configure PF eth0 and the first VF on VLAN 10.
$ ip link set dev eth0 vf 0 vlan 10

VLAN Tag Packet Steering

Allows you to send all packets with a specific VLAN tag to a particular SR-IOV virtual function (VF). Further, this feature allows you to designate a particular VF as trusted, and allows that trusted VF to request selective promiscuous mode on the Physical Function (PF).

To set a VF as trusted or untrusted, enter the following command in the Hypervisor:
  # ip link set dev eth0 vf 1 trust [on|off]

Once the VF is designated as trusted, use the following commands in the VM to set the VF to promiscuous mode.
  For promiscuous all:
  #ip link set eth2 promisc on
    Where eth2 is a VF interface in the VM
  For promiscuous Multicast:
  #ip link set eth2 allmulticast on
    Where eth2 is a VF interface in the VM

NOTE: By default, the ethtool priv-flag vf-true-promisc-support is set to "off",meaning that promiscuous mode for the VF will be limited. To set the promiscuous mode for the VF to true promiscuous and allow the VF to see all ingress traffic, use the following command.
  #ethtool -set-priv-flags p261p1 vf-true-promisc-support on
The vf-true-promisc-support priv-flag does not enable promiscuous mode; rather, it designates which type of promiscuous mode (limited or true) you will get when you enable promiscuous mode using the ip link commands above. Note that this is a global setting that affects the entire device. However,the vf-true-promisc-support priv-flag is only exposed to the first PF of the device. The PF remains in limited promiscuous mode (unless it is in MFP mode) regardless of the vf-true-promisc-support setting.

Now add a VLAN interface on the VF interface.
  #ip link add link eth2 name eth2.100 type vlan id 100

Note that the order in which you set the VF to promiscuous mode and add the VLAN interface does not matter (you can do either first). The end result in this example is that the VF will get all traffic that is tagged with VLAN 100.

Intel(R) Ethernet Flow Director

NOTE: Intel Ethernet Flow Director parameters are only supported on kernel versions 2.6.30 or newer.

The Intel Ethernet Flow Director performs the following tasks:

- Directs receive packets according to their flows to different queues.
- Enables tight control on routing a flow in the platform.
- Matches flows and CPU cores for flow affinity.
- Supports multiple parameters for flexible flow classification and load balancing (in SFP mode only).

NOTE: An included script (set_irq_affinity) automates setting the IRQ to CPU affinity.

NOTE: The Linux i40e driver supports the following flow types: IPv4, TCPv4, and UDPv4. For a given flow type, it supports valid combinations of IP addresses (source or destination) and UDP/TCP ports (source and destination). For example, you can supply only a source IP address, a source IP address and a destination port, or any combination of one or more of these four parameters.

NOTE: The Linux i40e driver allows you to filter traffic based on a user-defined flexible two-byte pattern and offset by using the ethtool user-def and mask fields. Only L3 and L4 flow types are supported for user-defined flexible filters. For a given flow type, you must clear all Intel Ethernet Flow Director filters before changing the input set (for that flow type).

ethtool commands:

To enable or disable the Intel Ethernet Flow Director:

```
# ethtool -K ethX ntuple <on|off>
```

When disabling ntuple filters, all the user programmed filters are flushed from the driver cache and hardware. All needed filters must be re-added when ntuple is re-enabled.

To add a filter that directs packet to queue 2, use -U or -N switch:

```
# ethtool -N ethX flow-type tcp4 src-ip 192.168.10.1 dst-ip \
192.168.10.2 src-port 2000 dst-port 2001 action 2 [loc 1]
```

To set a filter using only the source and destination IP address:

```
# ethtool -N ethX flow-type tcp4 src-ip 192.168.10.1 dst-ip \
192.168.10.2 action 2 [loc 1]
```

To set a filter based on a user defined pattern and offset:

```
# ethtool -N ethX flow-type tcp4 src-ip 192.168.10.1 dst-ip \
192.168.10.2 user-def 0xffffffff00000001 m 0x40 action 2 [loc 1]
```

where the value of the user-def field (0xffffffff00000001) is the pattern and m 0x40 is the offset.

Note that in this case the mask (m 0x40) parameter is used with the user-def field, whereas for cloud filter support the mask parameter is not used.

To see the list of filters currently present:

```
# ethtool <-u|-n> ethX
```

Application Targeted Routing (ATR) Perfect Filters

ATR is enabled by default when the kernel is in multiple transmit queue mode. An ATR Intel Ethernet Flow Director filter rule is added when a TCP-IP flow

starts and is deleted when the flow ends. When a TCP-IP Intel Ethernet Flow Director rule is added from ethtool (Sideband filter), ATR is turned off by the driver. To re-enable ATR, the sideband can be disabled with the ethtool -K option. For example:
ethtool –K [adapter] ntuple [off|on]

If sideband is re-enabled after ATR is re-enabled, ATR remains enabled until a TCP-IP flow is added. When all TCP-IP sideband rules are deleted, ATR is automatically re-enabled.

Packets that match the ATR rules are counted in fdir_atr_match stats in ethtool, which also can be used to verify whether ATR rules still exist.

Sideband Perfect Filters

Sideband Perfect Filters are used to direct traffic that matches specified characteristics. They are enabled through ethtool's ntuple interface. To add a new filter use the following command:
  ethtool -U <device> flow-type <type> src-ip <ip> dst-ip <ip> src-port <port> dst-port <port> action <queue>
Where:
  <device> - the ethernet device to program
  <type> - can be ip4, tcp4, udp4, or sctp4
  <ip> - the ip address to match on
  <port> - the port number to match on
  <queue> - the queue to direct traffic towards (-1 discards the matched traffic)
Use the following command to display all of the active filters:
  ethtool -u <device>
Use the following command to delete a filter:
  ethtool -U <device> delete <N>
Where <N> is the filter id displayed when printing all the active filters, and may also have been specified using "loc <N>" when adding the filter.

The following example matches TCP traffic sent from 192.168.0.1, port 5300, directed to 192.168.0.5, port 80, and sends it to queue 7:
  ethtool -U enp130s0 flow-type tcp4 src-ip 192.168.0.1 dst-ip 192.168.0.5 src-port 5300 dst-port 80 action 7

For each flow-type, the programmed filters must all have the same matching input set. For example, issuing the following two commands is acceptable:
  ethtool -U enp130s0 flow-type ip4 src-ip 192.168.0.1 src-port 5300 action 7
  ethtool -U enp130s0 flow-type ip4 src-ip 192.168.0.5 src-port 55 action 10
Issuing the next two commands, however, is not acceptable, since the first specifies src-ip and the second specifies dst-ip:
  ethtool -U enp130s0 flow-type ip4 src-ip 192.168.0.1 src-port 5300 action 7
  ethtool -U enp130s0 flow-type ip4 dst-ip 192.168.0.5 src-port 55 action 10
The second command will fail with an error. You may program multiple filters with the same fields, using different values, but, on one device, you may not program two tcp4 filters with different matching fields.

Matching on a sub-portion of a field is not supported by the i40e driver, thus partial mask fields are not supported.

The driver also supports matching user-defined data within the packet payload. This flexible data is specified using the "user-def" field of the ethtool command in the following way:
+--------------------------+-------------------------+
| 31 28 24 20 16 | 15 12 8 4 0 |

+----------------------------+-------------------------+
| offset into packet payload | 2 bytes of flexible data |
+----------------------------+-------------------------+

For example,
  ... user-def 0x4FFFF ...

tells the filter to look 4 bytes into the payload and match that value against 0xFFFF. The offset is based on the beginning of the payload, and not the beginning of the packet. Thus

  flow-type tcp4 ... user-def 0x8BEAF ...

would match TCP/IPv4 packets which have the value 0xBEAF 8 bytes into the TCP/IPv4 payload.

Note that ICMP headers are parsed as 4 bytes of header and 4 bytes of payload. Thus to match the first byte of the payload, you must actually add 4 bytes to the offset. Also note that ip4 filters match both ICMP frames as well as raw (unknown) ip4 frames, where the payload will be the L3 payload of the IP4 frame.

The maximum offset is 64. The hardware will only read up to 64 bytes of data from the payload. The offset must be even because the flexible data is 2 bytes long and must be aligned to byte 0 of the packet payload.

The user-defined flexible offset is also considered part of the input set and cannot be programmed separately for multiple filters of the same type. However, the flexible data is not part of the input set and multiple filters may use the same offset but match against different data.

To create filters that direct traffic to a specific Virtual Function, use the "action" parameter. Specify the action as a 64 bit value, where the lower 32 bits represents the queue number, while the next 8 bits represent which VF. Note that 0 is the PF, so the VF identifier is offset by 1. For example:

  ... action 0x800000002 ...

specifies to direct traffic to Virtual Function 7 (8 minus 1) into queue 2 of that VF.

Note that these filters will not break internal routing rules, and will not route traffic that otherwise would not have been sent to the specified Virtual Function.

Cloud Filter Support

On a complex network that supports multiple types of traffic (such as for storage as well as cloud), cloud filter support allows you to send one type of traffic (for example, the storage traffic) to the Physical Function (PF) and another type (say, the cloud traffic) to a Virtual Function (VF). Because cloud networks are typically VXLAN/GENEVE-based, you can define a cloud filter to identify VXLAN/GENEVE packets and send them to a queue in the VF to be processed by the virtual machine (VM). Similarly, other cloud filters can be designed for various other traffic tunneling.

NOTE: Cloud filters are only supported when the underlying device is in Single Function per Port mode.

NOTE: The "action -1" option, which drops matching packets in regular Intel

Ethernet Flow Director filters, is not available to drop packets when used with cloud filters.

NOTE: For IPv4 and ether flow-types, cloud filters cannot be used for TCP or UDP filters.

NOTE: Cloud filters can be used as a method for implementing queue splitting in the PF.

The following filters are supported:
  Cloud Filters
    Inner MAC, Inner VLAN (for NVGRE, VXLAN or GENEVE packets)
    Inner MAC, Inner VLAN, Tenant ID (for NVGRE, VXLAN or GENEVE packets)
    Inner MAC, Tenant ID (NVGRE packet or VXLAN/GENEVE packets)
    Outer MAC L2 filter
    Inner MAC filter
    Outer MAC, Tenant ID, Inner MAC
    Application Destination IP
    Application Source-IP, Inner MAC
    ToQueue: Use MAC, VLAN to point to a queue
  L3 filters
    Application Destination IP

Cloud filters are specified using ethtool's ntuple interface, but the driver uses user-def to determine whether to treat the filter as a cloud filter or a regular filter. To enable a cloud filter, set the highest bit of the user-def field, "user-def 0x8000000000000000" to enable the cloud features described below. This specifies to the driver to treat the filter specially and not treat it like the regular filters described above. Note that cloud filters also read the other bits in the user-def field separately so you cannot use the flexible data feature described above.

For regular Intel Ethernet Flow Director filters:
  - No user-def specified or highest bit (bit 63) is 0

  Example:

    ethtool -U enp130s0 flow-type ip4 src-ip 192.168.0.1 dst-ip 192.168.0.109
    action 6 loc

For L3 filters (non-tunneled packets):
  - "user-def 0x8000000000000000" (no Tenant ID/VNI specified in remaining
    bits of the user-def field)
  - Only L3 parameters (src-IP, dst-IP) are considered

  Example:

    ethtool -U enp130s0 flow-type ip4 src-ip 192.168.42.13 dst-ip 192.168.42.33
/
    src-port 12344 dst-port 12344 user-def 0x8000000000000000 action /
    0x200000000 loc 3
      Redirect traffic coming from 192.168.42.13 port 12344 with destination
      192.168.42.33 port 12344 into VF id 1, and call this "rule 3"

For cloud filters (tunneled packets):
  - All other filters, including where Tenant ID/VNI is specified.
  - The lower 32 bits of the user-def field can carry the tenant ID/VNI
    if required.
  - The VF can be specified using the "action" field, just as regular filters
    described in the Flow director Filter section above.

- Cloud filters can be defined with inner MAC, outer MAC, inner IP address, inner VLAN, and VNI as part of the cloud tuple. Cloud filters filter on destination (not source) MAC and IP. The destination and source MAC address fields in the ethtool command are overloaded as dst = outer, src = inner MAC address to facilitate tuple definition for a cloud filter.
- The 'loc' parameter specifies the rule number of the filter as being stored in the base driver

Example:

```
ethtool –U enp130s0 flow-type ether dst 8b:9d:ed:6a:ce:43 \
src 1d:44:9d:54:da:de user-def 0x8000000000000022 loc 38 \
action 0x200000000
```

Redirect traffic on VXLAN using tunnel id 34 (hex 0x22) coming from outer MAC address 8b:9d:ed:6a:ce:43 and inner MAC address 1d:44:9d:54:da:de into VF id 1 and call this “rule 38”.

==========

Additional Features and 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/modules.conf or /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 Base Driver is i40e.

Setting the link-down-on-close Private Flag

When the link-down-on-close private flag is set to "on", the port's link will go down when the interface is brought down using the ifconfig ethX down command.

Use ethtool to view and set link-down-on-close, as follows:

```
ethtool --show-priv-flags ethX
ethtool --set-priv-flags ethX link-down-on-close [on|off]
```

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

Jumbo Frames support is enabled by changing the Maximum Transmission Unit (MTU) to a value larger than the default value of 1500.

Use the ifconfig command to increase the MTU size. For example, enter the following where <x> is the interface number:

```
   ifconfig eth<x> mtu 9000 up
```

Alternatively, you can use the ip command as follows:

```
   ip link set mtu 9000 dev eth<x>
   ip link set up dev eth<x>
```

This setting is not saved across reboots. The setting change can be made permanent by adding 'MTU=9000' to the file:
/etc/sysconfig/network-scripts/ifcfg-eth<x> for RHEL or to the file /etc/sysconfig/network/<config_file> for SLES.

NOTE: The maximum MTU setting for Jumbo Frames is 9702. This value coincides with the maximum Jumbo Frames size of 9728 bytes.

NOTE: This driver will attempt to use multiple page sized buffers to receive each jumbo packet. This should help to avoid buffer starvation issues when allocating receive packets.

NOTE: Packet loss may have a greater impact on throughput when you use jumbo frames. If you observe a drop in performance after enabling jumbo frames, enabling flow control may mitigate the issue.

ethtool

The driver utilizes the ethtool interface for driver configuration and diagnostics, as well as displaying statistical information. The latest ethtool version is required for this functionality. Download it at:
http://ftp.kernel.org/pub/software/network/ethtool/

Supported ethtool Commands and Options for Filtering

-n --show-nfc
  Retrieves the receive network flow classification configurations.

rx-flow-hash tcp4|udp4|ah4|esp4|sctp4|tcp6|udp6|ah6|esp6|sctp6
  Retrieves the hash options for the specified network traffic type.

-N --config-nfc
  Configures the receive network flow classification.

rx-flow-hash tcp4|udp4|ah4|esp4|sctp4|tcp6|udp6|ah6|esp6|sctp6
m|v|t|s|d|f|n|r...
  Configures the hash options for the specified network traffic type.

  udp4 UDP over IPv4
  udp6 UDP over IPv6

  f Hash on bytes 0 and 1 of the Layer 4 header of the rx packet.
  n Hash on bytes 2 and 3 of the Layer 4 header of the rx packet.

Speed and Duplex Configuration

In addressing speed and duplex configuration issues, you need to distinguish between copper-based adapters and fiber-based adapters.

In the default mode, an Intel(R) Ethernet Network Adapter using copper connections will attempt to auto-negotiate with its link partner to determine the best setting. If the adapter cannot establish link with the link partner using auto-negotiation, you may need to manually configure the adapter and link partner to identical settings to establish link and pass packets. This should only be needed when attempting to link with an older switch that does not support auto-negotiation or one that has been forced to a specific speed or duplex mode. Your link partner must match the setting you choose. 1 Gbps speeds and higher cannot be forced. Use the autonegotiation advertising setting to manually set devices for 1 Gbps and higher.

NOTE: You cannot set the speed for devices based on the Intel(R) Ethernet Network Adapter XXV710 based devices.

Speed, duplex, and autonegotiation advertising are configured through the ethtool* utility. ethtool is included with all versions of Red Hat after Red Hat 7.2. For the latest version, download and install ethtool from the following website:

http://ftp.kernel.org/pub/software/network/ethtool/

Caution: Only experienced network administrators should force speed and duplex or change autonegotiation advertising manually. The settings at the switch must always match the adapter settings. Adapter performance may suffer or your adapter may not operate if you configure the adapter differently from your switch.

An Intel(R) Ethernet Network Adapter using fiber-based connections, however, will not attempt to auto-negotiate with its link partner since those adapters operate only in full duplex and only at their native speed.

NAPI

NAPI (Rx polling mode) is supported in the i40e driver.
For more information on NAPI, see
https://www.linuxfoundation.org/collaborate/workgroups/networking/napi

Flow Control

Ethernet Flow Control (IEEE 802.3x) can be configured with ethtool to enable receiving and transmitting pause frames for i40e. When transmit is enabled, pause frames are generated when the receive packet buffer crosses a predefined threshold. When receive is enabled, the transmit unit will halt for the time delay specified when a pause frame is received.

NOTE: You must have a flow control capable link partner.

Flow Control is disabled by default.

Use ethtool to change the flow control settings.

To enable or disable rx or tx Flow Control:
ethtool -A eth? rx <on|off> tx <on|off>
Note: This command only enables or disables Flow Control if auto-negotiation is disabled. If auto-negotiation is enabled, this command changes the parameters used for auto-negotiation with the link partner.

To enable or disable auto-negotiation:

ethtool -s eth? autoneg <on|off>
Note: Flow Control auto-negotiation is part of link auto-negotiation. Depending on your device, you may not be able to change the auto-negotiation setting.

RSS Hash Flow

Allows you to set the hash bytes per flow type and any combination of one or more options for Receive Side Scaling (RSS) hash byte configuration.

#ethtool -N <dev> rx-flow-hash <type> <option>

Where <type> is:
  tcp4   signifying TCP over IPv4
  udp4   signifying UDP over IPv4
  tcp6   signifying TCP over IPv6
  udp6   signifying UDP over IPv6
And <option> is one or more of:
  s   Hash on the IP source address of the rx packet.
  d   Hash on the IP destination address of the rx packet.
  f   Hash on bytes 0 and 1 of the Layer 4 header of the rx packet.
  n   Hash on bytes 2 and 3 of the Layer 4 header of the rx packet.

MAC and VLAN anti-spoofing feature

When a malicious driver attempts to send a spoofed packet, it is dropped by the hardware and not transmitted.
NOTE: This feature can be disabled for a specific Virtual Function (VF):
ip link set <pf dev> vf <vf id> spoofchk {off|on}

IEEE 1588 Precision Time Protocol (PTP) Hardware Clock (PHC)

Precision Time Protocol (PTP) is used to synchronize clocks in a computer network. PTP support varies among Intel devices that support this driver. Use "ethtool -T <netdev name>" to get a definitive list of PTP capabilities supported by the device.

IEEE 802.1ad (QinQ) Support

The IEEE 802.1ad standard, informally known as QinQ, allows for multiple VLAN IDs within a single Ethernet frame. VLAN IDs are sometimes referred to as "tags," and multiple VLAN IDs are thus referred to as a "tag stack." Tag stacks allow L2 tunneling and the ability to segregate traffic within a particular VLAN ID, among other uses.

The following are examples of how to configure 802.1ad (QinQ):
  ip link add link eth0 eth0.24 type vlan proto 802.1ad id 24
  ip link add link eth0.24 eth0.24.371 type vlan proto 802.1Q id 371
Where "24" and "371" are example VLAN IDs.

NOTES:
- 802.1ad (QinQ)is supported in 3.19 and later kernels.
- Receive checksum offloads, cloud filters, and VLAN acceleration are not supported for 802.1ad (QinQ) packets.

VXLAN and GENEVE Overlay HW Offloading

Virtual Extensible LAN (VXLAN) allows you to extend an L2 network over an L3 network, which may be useful in a virtualized or cloud environment. Some Intel(R) Ethernet Network devices perform VXLAN processing, offloading it from the operating system. This reduces CPU utilization.

VXLAN offloading is controlled by the tx and rx checksum offload options provided by ethtool. That is, if tx checksum offload is enabled, and the adapter has the capability, VXLAN offloading is also enabled.

Support for VXLAN and GENEVE HW offloading is dependent on kernel support of the HW offloading features.

Multiple Functions per Port

Some adapters based on the Intel Ethernet Controller X710/XL710 support multiple functions on a single physical port. Configure these functions through the System Setup/BIOS.

Minimum TX Bandwidth is the guaranteed minimum data transmission bandwidth, as a percentage of the full physical port link speed, that the partition will receive. The bandwidth the partition is awarded will never fall below the level you specify.

The range for the minimum bandwidth values is:
1 to ((100 minus # of partitions on the physical port) plus 1)
For example, if a physical port has 4 partitions, the range would be:
1 to ((100 - 4) + 1 = 97)

The Maximum Bandwidth percentage represents the maximum transmit bandwidth allocated to the partition as a percentage of the full physical port link speed. The accepted range of values is 1-100. The value is used as a limiter, should you chose that any one particular function not be able to consume 100% of a port's bandwidth (should it be available). The sum of all the values for Maximum Bandwidth is not restricted, because no more than 100% of a port's bandwidth can ever be used.

NOTE: X710/XXV710 devices fail to enable Max VFs (64) when Multiple Functions per Port (MFP) and SR-IOV are enabled. An error from i40e is logged that says "add vsi failed for VF N, aq_err 16". To workaround the issue, enable less than 64 virtual functions (VFs).

Data Center Bridging (DCB)

NOTE:
The kernel assumes that TC0 is available, and will disable Priority Flow Control (PFC) on the device if TC0 is not available. To fix this, ensure TC0 is enabled when setting up DCB on your switch.

DCB is a configuration Quality of Service implementation in hardware. It uses the VLAN priority tag (802.1p) to filter traffic. That means that there are 8 different priorities that traffic can be filtered into. It also enables priority flow control (802.1Qbb) which can limit or eliminate the number of

dropped packets during network stress. Bandwidth can be allocated to each of these priorities, which is enforced at the hardware level (802.1Qaz).

Adapter firmware implements LLDP and DCBX protocol agents as per 802.1AB and 802.1Qaz respectively. The firmware based DCBX agent runs in willing mode only and can accept settings from a DCBX capable peer. Software configuration of DCBX parameters via dcbtool/lldptool are not supported.

NOTE: Firmware LLDP can be disabled by setting the private flag disable-fw-lldp.

The i40e driver implements the DCB netlink interface layer to allow user-space to communicate with the driver and query DCB configuration for the port.

Interrupt Rate Limiting

The Intel(R) Ethernet Controller XL710 family supports an interrupt rate limiting mechanism. The user can control, via ethtool, the number of microseconds between interrupts.

Syntax:
ethtool -C ethX rx-usecs-high N

Valid Range: 0-235 (0=no limit)

The range of 0-235 microseconds provides an effective range of 4,310 to 250,000 interrupts per second. The value of rx-usecs-high can be set independently of rx-usecs and tx-usecs in the same ethtool command, and is also independent of the adaptive interrupt moderation algorithm. The underlying hardware supports granularity in 4-microsecond intervals, so adjacent values may result in the same interrupt rate.

One possible use case is the following:
ethtool -C ethX adaptive-rx off adaptive-tx off rx-usecs-high 20 rx-usecs 5 tx-usecs 5

The above command would disable adaptive interrupt moderation, and allow a maximum of 5 microseconds before indicating a receive or transmit was complete. However, instead of resulting in as many as 200,000 interrupts per second, it limits total interrupts per second to 50,000 via the rx-usecs-high parameter.

Application Device Queues (ADQ)

Application Device Queues (ADQ) allows you to dedicate one or more queues to a specific application. This can reduce latency for the specified application, and allow Tx traffic to be rate limited per application. Follow the steps below to set ADQ.

Requirements:
- Kernel version 4.15 or later
- The sch_mqprio, act_mirred and cls_flower modules must be loaded
- The latest version of iproute2
- NVM version 6.01 or later
- ADQ cannot be enabled when any the following features are enabled: Data Center Bridging (DCB), Multiple Functions per Port (MFP), or Sideband Filters.
- If another driver (for example, DPDK) has set cloud filters, you cannot enable ADQ.

1. Create traffic classes (TCs). You can create a maximum of 8 TCs per interface. The shaper bw_rlimit parameter is optional.

This example sets up two tcs, tc0 and tc1, with 16 queues each and max tx rate set to 1Gbit for tc0 and 3Gbit for tc1:

```
# tc qdisc add dev <interface> root mqprio num_tc 2 map 0 0 0 0 1 1 1 1
queues 16@0 16@16 hw 1 mode channel shaper bw_rlimit min_rate 1Gbit 2Gbit
max_rate 1Gbit 3Gbit
```

Where
map: priority mapping for up to 16 priorities to tcs (e.g. map 0 0 0 0 1 1 1 1 sets priorities 0-3 to use tc0 and 4-7 to use tc1)

queues: for each tc, <num queues>@<offset> (e.g. queues 16@0 16@16 assigns 16 queues to tc0 at offset 0 and 16 queues to tc1 at offset 16. Max total number of queues for all tcs is 64 or number of cores, whichever is lower.)

hw 1 mode channel: 'channel' with 'hw' set to 1 is a new new hardware offload mode in mqprio that makes full use of the mqprio options, the TCs, the queue configurations, and the QoS parameters.

shaper bw_rlimit: for each tc, sets minimum and maximum bandwidth rates. The totals must be equal or less than port speed. For example:

```
min_rate 1Gbit 3Gbit:
```

Verify bandwidth limit using network monitoring tools such as ifstat or sar –n DEV [interval] [number of samples]

NOTE: Setting up channels via ethtool (ethtool -L) is not supported when the TCs are configured using mqprio.

2. Enable HW TC offload on interface:

```
# ethtool -K <interface> hw-tc-offload on
```

3. Apply TCs to ingress (RX) flow of interface:

```
# tc qdisc add dev <interface> ingress
```

NOTES:
- Tunnel filters are not supported in ADQ. If encapsulated packets do arrive in non-tunnel mode, filtering will be done on the inner headers. For example, for VXLAN traffic in non-tunnel mode, if PCTYPE is identified as a VXLAN encapsulated packet, then the outer headers are ignored. Therefore, inner headers are matched.
- If a TC filter on a PF matches traffic over a VF (on the PF), that traffic will be routed to the appropriate queue of the PF, and will not be passed on the VF. Such traffic will end up getting dropped higher up in the TCP/IP stack as it does not match PF address data.
- If traffic matches multiple TC filters that point to different TCs, that traffic will be duplicated and sent to all matching TC queues. The hardware switch mirrors the packet to a VSI list when multiple filters are matched.

Forward Error Correction (FEC)

Allows you to set the Forward Error Correction (FEC) mode. FEC improves link stability, but increases latency. Many high quality optics, direct attach cables, and backplane channels provide a stable link without FEC.

NOTE: For devices to benefit from this feature, link partners must have FEC enabled.

On kernels older than 4.14, use the following private flags to disable
FEC modes:
  rs-fec (0 to disable, 1 to enable)
  base-r-fec (0 to disable, 1 to enable)
On kernel 4.14 or later, use ethtool to get/set the following FEC modes:
  No FEC
  Auto FEC
  BASE-R FEC
  RS FEC

SR-IOV Hypervisor Management Interface

The sysfs file structure below supports the SR-IOV hypervisor management
interface.

/sys/class/net/<interface-name>/device/sriov (see 1 below)
+-- [VF-id, 0 .. 127] (see 2 below)
| +-- trunk
| +-- vlan_mirror
| +-- engress_mirror
| +-- ingress_mirror
| +-- mac_anti_spoof
| +-- vlan_anti_spoof
| +-- loopback
| +-- mac
| +-- mac_list
| +-- promisc
| +-- vlan_strip
| +-- stats
| +-- link_state
| +-- max_tx_rate
| +-- min_tx_rate
| +-- spoofcheck
| +-- trust
| +-- vlan
*1: kobject started from “sriov” is not available from existing kernel
sysfs, and it requires device driver to implement this interface.
*2: assume maximum # of VF supported by a PF is 128. To support a device
that supports more than 128 SR-IOV instances, a “vfx” is added to 0..127.
With “vfx” kboject, users need to add vf index as the first parameter and
followed by “:”.

SR-IOV hypervisor functions:

- trunk
  Supports two operations: add and rem.
  - add: adds one or more VLAN id into VF VLAN filtering.
  - rem: removes VLAN ids from the VF VLAN filtering list.
  Example 1: add multiple VLAN tags, VLANs 2,4,5,10-20, by PF, p1p2, on
  a selected VF, 1, for filtering, with the sysfs support:
  #echo add 2,4,5,10-20 > /sys/class/net/p1p2/device/sriov/1/trunk
  Example 2: remove VLANs 5, 11-13 from PF p1p2 VF 1 with sysfs:
  #echo rem 5,11-13 > /sys/class/net/p1p2/device/sriov/1/trunk
  Note: for rem, if VLAN id is not on the VLAN filtering list, the
  VLAN id will be ignored.

- vlan_mirror
  Supports both ingress and egress traffic mirroring.
  Example 1: mirror traffic based upon VLANs 2,4,6,18-22 to VF 3 of PF p1p1.

echo add 2,4,6,18-22 > /sys/class/net/p1p1/device/sriov/3/vlan_mirror
Example 2: remove VLAN 4, 15-17 from traffic mirroring at destination VF 3.
echo rem 15-17 > /sys/class/net/p1p1/device/sriov/3/vlan_mirror
Example 3: remove all VLANs from mirroring at VF 3.
echo rem 0 - 4095> /sys/class/net/p1p1/device/sriov/3/vlan_mirror

- egress_mirror
Supports egress traffic mirroring.
Example 1: add egress traffic mirroring on PF p1p2 VF 1 to VF 7.
#echo add 7 > /sys/class/net/p1p2/device/sriov/1/egress_mirror
Example 2: remove egress traffic mirroring on PF p1p2 VF 1 to VF 7.
#echo rem 7 > /sys/class/net/p1p2/device/sriov/1/egress_mirror

- ingress_mirror
Supports ingress traffic mirroring.
Example 1: mirror ingress traffic on PF p1p2 VF 1 to VF 7.
#echo add 7 > /sys/class/net/p1p2/device/sriov/1/ingress_mirror
Example 2: show current ingress mirroring configuration.
#cat /sys/class/net/p1p2/device/sriov/1/ingress_mirror

- mac_anti_spoof
Supports Enable/Disable MAC anti-spoof. Allows VFs to transmit packets with any SRC MAC, which is needed for some L2 applications as well as vNIC bonding within VMs if set to OFF.
Example 1: enable MAC anti-spoof for PF p2p1 VF 1.
#echo ON /sys/class/net/p1p2/device/sriov/1/mac_anti_spoof
Example 2: disable MAC anti-spoof for PF p2p1 VF 1.
#echo OFF /sys/class/net/p1p2/device/sriov/1/mac_anti_spoof

- vlan_anti_spoof
Supports Enable/Disable VLAN anti-spoof. Allows VFs to transmit packets only with VLAN tag specified in “trunk” settings, also will not allow to transmit “untagged” packets if set to ON. Violation have to increment tx_spoof stats counter.
Example 1: enable VLAN anti-spoof for PF p2p1 VF 1.
#echo ON /sys/class/net/p1p2/device/sriov/1/vlan_anti_spoof
Example 2: disable VLAN anti-spoof for PF p2p1 VF 1.
#echo OFF /sys/class/net/p1p2/device/sriov/1/vlan_anti_spoof

- loopback
Supports Enable/Disable VEB/VEPA (Local loopback).
Example 1: allow traffic switching between VFs on the same PF.
#echo ON > /sys/class/net/p1p2/device/sriov/loopback
Example 2: send Hairpin traffic to the switch to which the PF is connected.
#echo OFF > /sys/class/net/p1p2/device/sriov/loopback
Example 3: show loopback configuration.
#cat /sys/class/net/p1p2/device/sriov/loopback

- mac
Supports setting default MAC address. If MAC address is set by this command, the PF will not allow VF to change it using an MBOX request.
Example 1: set default MAC address to VF 1.
#echo "00:11:22:33:44:55" > /sys/class/net/p1p2/device/sriov/1/mac
Example 2: show default MAC address.
#cat /sys/class/net/p1p2/device/sriov/1/mac

- mac_list
Supports adding additional MACs to the VF. The default MAC is taken from "ip link set p1p2 vf 1 mac 00:11:22:33:44:55" if configured. If not, a random address is assigned to the VF by the NIC. If the MAC is configured using

the IP LINK command, the VF cannot change it via MBOX/AdminQ requests.
  Example 1: add mac 00:11:22:33:44:55 and 00:66:55:44:33:22 to PF p1p2 VF 1.
  #echo add "00:11:22:33:44:55,00:66:55:44:33:22" >
  /sys/class/net/p1p2/device/sriov/1/mac_list
  Example 2: delete mac 00:11:22:33:44:55 from above VF device.
  #echo rem 00:11:22:33:44:55 > /sys/class/net/p1p2/device/sriov/1/mac_list
  Example 3: display a VF MAC address list.
  #cat /sys/class/net/p1p2/device/sriov/1/mac_lis

- promisc
  Supports setting/unsetting VF device unicast promiscuous mode and multicast
  promiscuous mode.
  Example 1: set MCAST promiscuous on PF p1p2 VF 1.
  #echo add mcast > /sys/class/net/p1p2/device/sriov/1/promisc
  Example 2: set UCAST promiscuous on PF p1p2 VF 1.
  #echo add ucast > /sys/class/net/p1p2/device/sriov/1/promisc
  Example 3: unset MCAST promiscuous on PF p1p2 VF 1.
  #echo rem mcast > /sys/class/net/p1p2/device/sriov/1/promisc
  Example 4: show current promiscuous mode configuration.
  #cat /sys/class/net/p1p2/device/sriov/1/promisc

- vlan_strip
  Supports enabling/disabling VF device outer VLAN stripping
  Example 1: enable VLAN strip on VF 3.
  # echo ON > /sys/class/net/p1p1/device/sriov/3/vlan_strip
  Example 2: disable VLAN striping VF 3.
  # echo OFF > /sys/class/net/p1p1/device/sriov/3/vlan_strip

- stats
  Supports getting VF statistics
  Example 1: display anti-spoofing violations counter for VF 1.
  #cat /sys/class/net/p1p2/device/sriov/1/stats/tx_spoofed

- link_state
  Sets/displays link status.
  Example 1: display link status on link speed.
  #cat /sys/class/net/p1p2/device/sriov/1/link_state
  Example 2 set VF 1 to track status of PF link.
  #echo auto > /sys/class/net/p1p2/device/sriov/1/link_state
  Example 3: disable VF 1.
  #echo disable > /sys/class/net/p1p2/device/sriov/1/link_state

Dynamic Device Personalization

Dynamic Device Personalization (DDP) allows you to change the packet processing
pipeline of a device by applying a profile package to the device at runtime.
Profiles can be used to, for example, add support for new protocols, change
existing protocols, or change default settings. DDP profiles can also be rolled
back without rebooting the system.
Requirements:
-Intel Ethernet X710/XXV710/XL710 adapter (X722 series devices are not
supported at this time)
-Firmware 6.0 or newer

To apply a profile, copy it first to intel/i40e/ddp directory relative to your
firmware root (usually /lib/firmware).
For example:
  /lib/firmware/intel/i40e/ddp

Then use the ethtool -f|--flash flag with region 100:
ethtool -f <interface name> <profile name> 100

For example:
  ethtool -f eth0 gtp.pkgo 100

You can roll back to a previously loaded profile using '-' instead of profile name:
ethtool -f <interface name> - 100

For example:
  ethtool -f eth0 - 100

For every rollback request one profile will be removed, from last to first (LIFO) order.

NOTES:
- DDP profiles are loaded only on the interface corresponding to first physical function of the device (PF0), but the configuration is applied to all ports of the adapter.
- DDP profiles are not persistent. A system reboot will reset the device to its default configuration.
- DDP profiles are NOT automatically unloaded when the driver in unbound/unloaded. Please note that subsequent driver reload may corrupt the profile configuration during its initialization and is NOT recommended.
- DDP profiles should be manually rolled-back before driver unload/unbind if the intention is to start with clean HW configuration.
- Exercise caution while loading DDP profiles, attempting to load files other than DDP profiles provided by Intel may cause system instability, system crashes, or system hangs.

More details about Dynamic Device Personalization can be found on the Intel Developer Zone site:
https://software.intel.com/en-us/articles/dynamic-device-personalization-for-intel-ethernet-700-series

Performance Optimization:

Driver defaults are meant to fit a wide variety of workloads, but if further optimization is required we recommend experimenting with the following settings.

NOTE: For better performance when processing small (64B) frame sizes, try enabling Hyper threading in the BIOS in order to increase the number of logical cores in the system and subsequently increase the number of queues available to the adapter.

Virtualized Environments:

1. Disable XPS on both ends by using the included virt_perf_default script
  or by running the following command as root:
  for file in `ls /sys/class/net/<ethX>/queues/tx-*/xps_cpus`;
  do echo 0 > $file; done

2. Using the appropriate mechanism (vcpupin) in the vm, pin the cpu's to
  individual lcpu's, making sure to use a set of cpu's included in the
  device's local_cpulist: /sys/class/net/<ethX>/device/local_cpulist.

3. Configure as many rx/tx queues in the VM as available. Do not rely on

the default setting of 1.

Non-virtualized Environments

Pin the adapter's IRQs to specific cores by disabling the irqbalance service
and using the included set_irq_affinity script. Please see the script's help
text for further options.

- The following settings will distribute the IRQs across all the cores
  evenly:

  # scripts/set_irq_affinity -x all <interface1> , [<interface2>, ...]

- The following settings will distribute the IRQs across all the cores that
  are local to the adapter (same NUMA node):

  # scripts/set_irq_affinity -x local <interface1> ,[<interface2>, ...]

For very CPU intensive workloads, we recommend pinning the IRQs to all cores.

For IP Forwarding: Disable Adaptive ITR and lower rx and tx interrupts per
queue using ethtool.

- Setting rx-usecs and tx-usecs to 125 will limit interrupts to about 8000
  interrupts per second per queue.

  # ethtool -C <interface> adaptive-rx off adaptive-tx off rx-usecs 125
  tx-usecs 125

For lower CPU utilization: Disable Adaptive ITR and lower rx and tx interrupts
per queue using ethtool.

- Setting rx-usecs and tx-usecs to 250 will limit interrupts to about 4000
  interrupts per second per queue.

  # ethtool -C <interface> adaptive-rx off adaptive-tx off rx-usecs 250
  tx-usecs 250

For lower latency: Disable Adaptive ITR and ITR by setting rx and tx to 0 using
ethtool.

  # ethtool -C <interface> adaptive-rx off adaptive-tx off rx-usecs 0
  tx-usecs 0

==========

Known Issues/Troubleshooting
============================

LACP PDUs not being sent

Configuring bonding interface of mode 4 (802.3ad) on Ubuntu 14.04.5 will
require manually recompiling the kernel with additional patch applied:
 https://lore.kernel.org/patchwork/patch/651248/

The reason for that is because Ubuntu's kernel 4.4 backports new
ethtool GKLINKSETTINGS API from kernel 4.6 without tuning bonding driver

to actually use it.

Failure to enable MAX VFs when MFP and SR-IOV enabled

X710/XXV710 devices fail to enable Max VFs (64) when Multiple Functions per Port (MFP) and SR-IOV are enabled. An error from i40e is logged that says "add vsi failed for VF N, aq_err 16". To workaround the issue, enable less than 64 virtual functions (VFs).

ip link show command shows incorrect VF MAC if VF MAC was set from VF side
---
Executing the command "ip link show" only shows MAC addresses if they are set by the PF. Otherwise, it shows all zeros.

This is expected behavior. The PF driver is passing zeroes to the VF driver that the VF driver can generate its own random MAC address and report it to the guest OS. Without this feature, some guest operating systems will incorrectly assign the VF a new interface name each time they reboot.

SSL error (No such file) when installing driver on Ubuntu 14.04

When installing the driver on Ubuntu 14.04, you may get an SSL error stating "no such file or directory." This issue does not affect driver installation or performance and can be ignored.

IPv6/UDP checksum offload does not work on some older kernels
---
Some distributions with older kernels do not properly enable IPv6/UDP checksum offload. To use IPv6 checksum offload, it may be necessary to upgrade to a newer kernel.

Poor performance when using VXLAN encapsulation

When using VXLAN encapsulation on Red Hat Enterprise Linux 7.2 and 7.3, you may experience poor performance due to limitations in the kernel on these OS releases. To resolve this issue, upgrade your kernel.

Driver Buffer Overflow Fix

The fix to resolve CVE-2016-8105, referenced in Intel SA-00069 <https://security-center.intel.com/advisory.aspx?intelid=INTEL-SA-00069&languageid=en-fr>, is included in this and future versions of the driver.

depmod warning messages about unknown symbol during installation
---
During driver installation, you may see depmod warning messages referring to unknown symbols i40e_register_client and i40e_unregister_client. These messages are informational only; no user action is required. The installation should complete successfully.

Error: <ifname> selects TX queue XX but real number of TX queues is YY

When configuring the number of queues under heavy traffic load, you may see an error message stating "<ifname> selects TX queue XX, but real number of TX queues is YY". This message is informational only and does not affect functionality.

Windows Server 2016 Does Not Work as a Guest OS on Older RHEL and SLES KVMs

Microsoft* Windows Server* 2016 does not work as a guest operating system on the KVM hypervisor version included with Red Hat* Enterprise Linux* (RHEL) version 6.8 and Suse* Linux Enterprise Server (SLES) version 11.4. Windows Server 2016 does work as a guest OS on RHEL 7.2 and SLES 12.1.

Fixing Performance Issues When Using IOMMU in Virtualized Environments

The IOMMU feature of the processor prevents I/O devices from accessing memory outside the boundaries set by the OS. It also allows devices to be directly assigned to a Virtual Machine. However, IOMMU may affect performance, both in latency (each DMA access by the device must be translated by the IOMMU) and in CPU utilization (each buffer assigned to every device must be mapped in the IOMMU).

If you experience significant performance issues with IOMMU, try using it in "passthrough" mode by adding the following to the kernel boot command line:

```
intel_iommu=on iommu=pt
```

NOTE: This mode enables remapping for assigning devices to VMs, providing near-native I/O performance, but does not provide the additional memory protection.

Transmit hangs leading to no traffic

Disabling flow control while the device is under stress may cause tx hangs and eventually lead to the device no longer passing traffic. You must reboot the system to resolve this issue.

Incomplete messages in the system log

The NVMUpdate utility may write several incomplete messages in the system log. These messages take the form:

```
in the driver Pci Ex config function byte index 114
in the driver Pci Ex config function byte index 115
```

These messages can be ignored.

Bad checksum counter incorrectly increments when using VXLAN

When passing non-UDP traffic over a VXLAN interface, the port.rx_csum_bad counter increments for the packets.

Statistic counters reset when promiscuous mode is changed

Changing promiscuous mode triggers a reset of the physical function driver. This will reset the statistic counters.

Virtual machine does not get link

If the virtual machine has more than one virtual port assigned to it, and those virtual ports are bound to different physical ports, you may not get link on all of the virtual ports. The following command may work around the issue:
ethtool -r <PF>
Where <PF> is the PF interface in the host, for example: p5p1. You may need to run the command more than once to get link on all virtual ports.

MAC address of Virtual Function changes unexpectedly

If a Virtual Function's MAC address is not assigned in the host, then the VF (virtual function) driver will use a random MAC address. This random MAC address may change each time the VF driver is reloaded. You can assign a static MAC address in the host machine. This static MAC address will survive a VF driver reload.

Changing the number of Rx or Tx queues with ethtool -L may cause a kernel panic
---
Changing the number of Rx or Tx queues with ethtool -L while traffic is flowing and the interface is up may cause a kernel panic. Bring the interface down first to avoid the issue. For example:
  ip link set ethx down
  ethtool -L ethx combined 4

Adding an Intel Ethernet Flow Director Sideband rule fails incorrectly
---
If you try to add an Intel Ethernet Flow Director rule when no more sideband rule space is available, i40e logs an error that the rule could not be added, but ethtool returns success. You can remove rules to free up space. In addition, remove the rule that failed. This will evict it from the driver's cache.

Intel Ethernet Flow Director Sideband Logic adds duplicate filter
---
The Intel Ethernet Flow Director Sideband Logic adds a duplicate filter in the software filter list if the location is not specified or the specified location differs from the previous location but has the same filter criteria. In this case, the second of the two filters that appear is the valid one in hardware and it decides the filter action.

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, either turn on ARP filtering by entering:
echo 1 > /proc/sys/net/ipv4/conf/all/arp_filter

This only works if your kernel's version is higher than 2.4.5.

NOTE: This setting is not saved across reboots. The configuration change can be made permanent by adding the following line to the file /etc/sysctl.conf:
net.ipv4.conf.all.arp_filter = 1

Another alternative is to 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 packet UDP stress with the i40e driver, the system may drop UDP packets due to socket buffers being full. Setting the driver Intel Ethernet Flow Control variables to the minimum may resolve the issue. You may also try increasing the kernel's default buffer sizes by changing the values in

/proc/sys/net/core/rmem_default and rmem_max

Unplugging Network Cable While ethtool -p is Running

In kernel versions 2.6.32 and newer, unplugging the network cable while ethtool -p is running will cause the system to become unresponsive to keyboard commands, except for control-alt-delete. Restarting the system should resolve the issue.

Rx Page Allocation Errors

'Page allocation failure. order:0' errors may occur under stress with kernels 2.6.25 and newer.
This is caused by the way the Linux kernel reports this stressed condition.

Lower than expected performance

Some PCIe x8 slots are actually configured as x4 slots. These slots have insufficient bandwidth for full line rate with dual port and quad port devices. In addition, if you put a PCIe v3.0-capable adapter into a PCIe v2.x slot, you cannot get full bandwidth. The driver detects this situation and writes the following message in the system log:

"PCI-Express bandwidth available for this card is not sufficient for optimal performance. For optimal performance a x8 PCI-Express slot is required."

If this error occurs, moving your adapter to a true PCIe v3.0 x8 slot will resolve the issue.

ethtool may incorrectly display SFP+ fiber module as direct attached cable

Due to kernel limitations, port type can only be correctly displayed on kernel 2.6.33 or greater.

Running ethtool -t ethX command causes break between PF and test client

When there are active VFs, "ethtool -t" performs a full diagnostic. In the process, it resets itself and all attached VFs. The VF drivers encounter a

disruption, but are able to recover.

Enabling SR-IOV in a 64-bit Microsoft* Windows Server* 2012/R2 guest OS under Linux KVM

KVM Hypervisor/VMM supports direct assignment of a PCIe device to a VM. This includes traditional PCIe devices, as well as SR-IOV-capable devices based on the Intel Ethernet Controller XL710.

Unable to obtain DHCP lease on boot with RedHat

In configurations where the auto-negotiation process takes more than 5 seconds, the boot script may fail with the following message:
"ethX: failed. No link present. Check cable?"

This error may occur even though the presence of link can be confirmed using ethtool ethx. In this case, try setting "LINKDELAY=30" in /etc/sysconfig/network-scripts/ifdfg-ethx.

The same issue can occur during a network boot (via PXE) on RedHat distributions that use the dracut script:
"Warning: No carrier detected on interface <interface_name>"

In this case add "rd.net.timeout.carrier=30" at the kernel command line.

NOTE: Link time can vary. Adjust LINKDELAY value accordingly.

Alternatively, NetworkManager can be used to configure the interfaces, which avoids the set timeout. For configuration instructions of NetworkManager refer to the documentation provided by your distribution.

Loading i40e driver in 3.2.x and newer kernels displays kernel tainted message
---
Due to recent kernel changes, loading an out of tree driver causes the kernel to be tainted.

Unexpected Issues when the device driver and DPDK share a device
---
Unexpected issues may result when an i40e device is in multi driver mode and the kernel driver and DPDK driver are sharing the device. This is because access to the global NIC resources is not synchronized between multiple drivers. Any change to the global NIC configuration (writing to a global register, setting global configuration by AQ, or changing switch modes) will affect all ports and drivers on the device. Loading DPDK with the "multi-driver" module parameter may mitigate some of the issues.

==========

Support
=======
For general information, go to the Intel support website at:
http://www.intel.com/support/

or the Intel Wired Networking project hosted by Sourceforge at:

http://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.

==========

License

==========

Trademarks
