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

Contents

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.

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

Identifying Your Adapter

The driver 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.

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

SFP+ and QSFP+ Devices

For information about supported media, refer to this document:
https://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. 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: 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 systems that do not have adequate airflow to cool the adapter and optical modules, you must use high temperature optical modules.

Virtual Functions (VFs)

Use sysfs to enable VFs. 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
```

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

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: 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).

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 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 matching 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.

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
/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.

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: https://www.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.

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://wiki.linuxfoundation.org/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 on 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:

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)

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.

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.

Interrupt Rate Limiting

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

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
```

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.

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
```

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.

1. Create traffic classes (TCs). Maximum of 8 TCs can be created per interface. The shaper bw_rlimit parameter is optional.

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
```

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: â€˜channelâ€™ with â€˜hwâ€™ 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. 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]*

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:

- Run all tc commands from the iproute2 <pathtoiproute2>/tc/ directory.
- ADq is not compatible with cloud filters.
- Setting up channels via ethtool (ethtool -L) is not supported when the TCs are configured using mqprio.
- You must have iproute2 latest version
- NVM version 6.01 or later is required.
- 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.
- 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, PCTYPE is identified as a VXLAN encapsulated packet, 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.

Known Issues/Troubleshooting

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

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.

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.

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.