Linux Base Driver for Intel(R) Ethernet Adaptive Virtual Function

Contents

Overview

This file describes the iavf Linux Base Driver. This driver was formerly called i40evf.

The iavf driver supports the below mentioned virtual function devices and can only be activated on kernels running the i40e or newer Physical Function (PF) driver compiled with CONFIG_PCI_IOV. The iavf driver requires CONFIG_PCI_MSI to be enabled.

The guest OS loading the iavf driver must support MSI-X interrupts.

Identifying Your Adapter

The driver in this kernel is compatible with devices based on the following:

- Intel(R) XL710 X710 Virtual Function
- Intel(R) X722 Virtual Function
- Intel(R) XXV710 Virtual Function
- Intel(R) Ethernet Adaptive Virtual Function

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

Additional Features and Configurations

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.

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/

Setting VLAN Tag Stripping

If you have applications that require Virtual Functions (VFs) to receive packets with VLAN tags, you can disable VLAN tag stripping for the VF. The Physical Function (PF) processes requests issued from the VF to enable or disable VLAN tag stripping. Note that if the PF has assigned a VLAN to a VF, then requests from that VF to set VLAN tag stripping will be ignored.

To enable/disable VLAN tag stripping for a VF, issue the following command from inside the VM in which you are running the VF:

```
# ethtool -K <if_name> rxvlan on/off
```

or alternatively:

```
# ethtool --offload <if_name> rxvlan on/off
```

Adaptive Virtual Function

Adaptive Virtual Function (AVF) allows the virtual function driver, or VF, to adapt to changing feature sets of the physical function driver (PF) with which it is associated. This allows system administrators to update a PF without having to update all the VFs associated with it. All AVFs have a single common device ID and branding string.

AVFs have a minimum set of features known as "base mode," but may provide additional features depending on what features are available in the PF with which the AVF is associated. The following are base mode features:

- 4 Queue Pairs (QP) and associated Configuration Status Registers (CSRs) for Tx/Rx
- i40e descriptors and ring format
- Descriptor write-back completion
- 1 control queue, with i40e descriptors, CSRs and ring format
- 5 MSI-X interrupt vectors and corresponding i40e CSRs
- 1 Interrupt Throttle Rate (ITR) index
- 1 Virtual Station Interface (VSI) per VF
- 1 Traffic Class (TC), TC0
- Receive Side Scaling (RSS) with 64 entry indirection table and key, configured through the PF
- 1 unicast MAC address reserved per VF
- 16 MAC address filters for each VF
- Stateless offloads - non-tunneled checksums
- AVF device ID
- HW mailbox is used for VF to PF communications (including on Windows)

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.

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:

- The sch_mqprio, act_mirred and cls_flower modules must be loaded
- The latest version of iproute2
- If another driver (for example, DPDK) has set cloud filters, you cannot enable ADQ
- Depending on the underlying PF device, ADQ cannot be enabled when the following features are enabled:
  - Data Center Bridging (DCB)
  - Multiple Functions per Port (MFP)
  - Sideband Filters

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]`

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:

- 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

Bonding fails with VFs bound to an Intel(R) Ethernet Controller 700 series device

If you bind Virtual Functions (VFs) to an Intel(R) Ethernet Controller 700 series based device, the VF slaves may fail when they become the active slave. If the MAC address of the VF is set by the PF (Physical Function) of the device, when you add a slave, or change the active-backup slave, Linux bonding tries to sync the backup slave's MAC address to the same MAC address as the active slave. Linux bonding will fail at this point. This issue will not occur if the VF's MAC address is not set by the PF.

Traffic Is Not Being Passed Between VM and Client

You may not be able to pass traffic between a client system and a Virtual Machine (VM) running on a separate host if the Virtual Function (VF, or Virtual NIC) is not in trusted mode and spoof checking is enabled on the VF. Note that this situation can occur in any combination of client, host, and guest operating system. For information on how to set the VF to trusted mode, refer to the section "VLAN Tag Packet Steering" in this readme document. For information on setting spoof checking, refer to the section "MAC and VLAN anti-spoofing feature" in this readme document.

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.

Using four traffic classes fails

Do not try to reserve more than three traffic classes in the iavf driver. Doing so will fail to set any traffic classes and will cause the driver to write errors to stdout. Use a maximum of three queues to avoid this issue.

Multiple log error messages on iavf driver removal

If you have several VFs and you remove the iavf driver, several instances of the following log errors are written to the log:

```
Unable to send opcode 2 to PF, err I40E_ERR_QUEUE_EMPTY, aq_err ok
Unable to send the message to VF 2 aq_err 12
ARQ Overflow Error detected
```

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 addresses of Virtual Functions not working as expected

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.

Driver Buffer Overflow Fix

The fix to resolve CVE-2016-8105, referenced in Intel SA-00069 https://www.intel.com/content/www/us/en/security-center/advisory/intel-sa-00069.html is included in this and future versions of the driver.

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
```

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).

Rx Page Allocation Errors

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

Support

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

https://support.intel.com

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 the supported kernel with a supported adapter, email the specific information related to the issue to e1000-devel@lists.sf.net