NAME

ena - FreeBSD kernel driver for Elastic Network Adapter (ENA) family

SYNOPSIS

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

device ena

Alternatively, to load the driver as a module at boot time, place the following line in loader.conf(5):

if_ena_load="YES"

DESCRIPTION

The ENA is a networking interface designed to make good use of modern CPU features and system architectures.

The ENA device exposes a lightweight management interface with a minimal set of memory mapped registers and extendable command set through an Admin Queue.

The driver supports a range of ENA devices, is link-speed independent (i.e., the same driver is used for 10GbE, 25GbE, 40GbE, etc.), and has a negotiated and extendable feature set.

Some ENA devices support SR-IOV. This driver is used for both the SR-IOV Physical Function (PF) and Virtual Function (VF) devices.

The ENA devices enable high speed and low overhead network traffic processing by providing multiple Tx/Rx queue pairs (the maximum number is advertised by the device via the Admin Queue), a dedicated MSI-X interrupt vector per Tx/Rx queue pair, and CPU cacheline optimized data placement.

When RSS is enabled, each Tx/Rx queue pair is bound to a corresponding CPU core and its NUMA domain. The order of those bindings is based on the RSS bucket mapping. For builds with RSS support disabled, the CPU and NUMA management is left to the kernel. Receive-side scaling (RSS) is supported for multi-core scaling.

The **ena** driver and its corresponding devices implement health monitoring mechanisms such as watchdog, enabling the device and driver to recover in a manner transparent to the application, as well as debug logs.

Some of the ENA devices support a working mode called Low-latency Queue (LLQ), which saves several more microseconds.

Support for the netmap(4) framework is provided by the **ena** driver. Kernel must be built with the DEV_NETMAP option to be able to use this feature.

HARDWARE

Supported PCI vendor ID/device IDs:

- 1d0f:0ec2 - ENA PF
- 1d0f:1ec2 - ENA PF with LLQ support
- 1d0f:ec20 - ENA VF
- 1d0f:ec21 - ENA VF with LLQ support

LOADER TUNABLES

The **ena** driver's behavior can be changed using run-time or boot-time sysctl arguments. The boot-time arguments can be set at the loader(8) prompt before booting the kernel, or stored in the loader.conf(5). The run-time arguments can be set using the sysctl(8) command.

Boot-time tunables:

hw.ena.enable_9k_mbufs

Use 9k mbufs for the Rx descriptors. The default is 0. If the node value is set to 1, 9k mbufs will be used for the Rx buffers. If set to 0, page size mbufs will be used instead.

Using 9k buffers for Rx can improve Rx throughput, but in low memory conditions it might increase allocation time, as the system has to look for 3 contiguous pages. This can further lead to OS instability, together with ENA driver reset and NVMe timeouts. If network performance is critical and memory capacity is sufficient, the 9k mbufs can be used.

hw.ena.force_large_llq_headers

Force the driver to use large LLQ headers (224 bytes). The default is 0. If the node value is set to 0, the regular size LLQ header will be used, which is 96B. In some cases, the packet header can be bigger than this (for example - IPv6 with multiple extensions). In such a situation, the large LLQ headers should be used by setting this node value to 1. This will take effect only if the device supports both LLQ and large LLQ headers. Otherwise, it will fallback to the no LLQ mode or regular header size.

Increasing LLQ header size reduces the size of the Tx queue by half, so it may affect the number of dropped Tx packets.

Run-time tunables:

hw.ena.log_level

Controls extra logging verbosity of the driver. The default is 2. The higher the logging level, the more logs will be printed out. 0 means all extra logs are disabled and only error logs will be printed out. Default value (2) reports errors, warnings and is verbose about driver operation.

The possible flags are:

- 0 - ENA_ERR - Enable driver error messages and ena_com error logs.
- 1 - ENA_WARN - Enable logs for non-critical errors.
- 2 - ENA_INFO - Make the driver more verbose about its actions.
- 3 - ENA_DBG - Enable debug logs.

NOTE: In order to enable logging on the Tx/Rx data path, driver must be compiled with ENA_LOG_IO_ENABLE compilation flag.

Example: To enable logs for errors and warnings, the following command should be used:

```
sysctl hw.ena.log_level=1
```

dev.ena.X.io_queues_nb

Number of the currently allocated and used IO queues. The default is max_num_io_queues. Controls the number of IO queue pairs (Tx/Rx). As this call has to reallocate the queues, it will reset the interface and restart all the queues - this means that everything, which was currently held in the queue, will be lost, leading to potential packet drops.

This call can fail if the system isn't able to provide the driver with enough resources. In that situation, the driver will try to revert the previous number of the IO queues. If this also fails, the device reset will be triggered.

Example: To use only 2 Tx and Rx queues for the device ena1, the following command should be used:

```
sysctl dev.ena.1.io_queues_nb=2
```

dev.ena.X.rx_queue_size

Size of the Rx queue. The default is 1024. Controls the number of IO descriptors for each Rx queue. The user may want to increase the Rx queue size if they observe a high number of Rx drops in the driver's statistics. For performance reasons, the Rx queue size must be a power of 2.

This call can fail if the system isn't able to provide the driver with enough resources. In that

situation, the driver will try to revert to the previous number of the descriptors. If this also fails, the device reset will be triggered.

Example: To increase Rx ring size to 8K descriptors for the device ena0, the following command should be used:

```
sysctl dev.ena.0.rx_queue_size=8192
```

dev.ena.X.buf_ring_size

Size of the Tx buffer ring (drbr). The default is 4096. Input must be a power of 2. Controls the number of mbufs that can be held in the Tx buffer ring. The drbr is used as a multiple-producer, single-consumer lockless ring for buffering extra mbufs coming from the stack in case the Tx procedure is busy sending the packets, or the Tx ring is full. Increasing the size of the buffer ring may reduce the number of Tx packets being dropped in case of a big Tx burst, which cannot be handled by the IO queue immediately. Each Tx queue has its own drbr.

It is recommended to keep the drbr with at least the default value, but in case the system lacks the resources, it can be reduced. This call can fail if the system is not able to provide the driver with enough resources. In that situation, the driver will try to revert to the previous number of the drbr and trigger the device reset.

Example: To set drbr size for interface ena0 to 2048, the following command should be used:

```
sysctl dev.ena.0.buf_ring_size=2048
```

dev.ena.X.eni_metrics.sample_interval

Interval in seconds for updating ENI metrics. The default is 0. Determines how often (if ever) the ENI metrics should be updated. The ENI metrics are being updated asynchronously in a timer service in order to avoid admin queue overload by sysctl node reading. The value in this node controls the interval between issuing admin commands to the device, which will update the ENI metrics values.

If some application is periodically monitoring the eni_metrics, then the ENI metrics interval can be adjusted accordingly. Value 0 turns off the update completely. Value 1 is the minimum interval and is equal to 1 second. The maximum allowed update interval is 1 hour.

Example: To update ENI metrics for the device ena1 every 10 seconds, the following command should be used:

```
sysctl dev.ena.1.eni_metrics.sample_interval=10
```

dev.ena.X.rss.indir_table_size

RSS indirection table size. The default is 128. Returns the number of entries in the RSS indirection table.

Example: To read the RSS indirection table size, the following command should be used:

```
sysctl dev.ena.0.rss.indir_table_size
```

dev.ena.X.rss.indir_table

RSS indirection table mapping. The default is x:y key-pairs of indir_table_size length. Updates selected indices of the RSS indirection table.

The entry string consists of one or more x:y keypairs, where x stands for the table index and y for its new value. Table indices that don’t need to be updated can be omitted from the string and will retain their existing values.

If an index is entered more than once, the last value is used.

Example: To update two selected indices in the RSS indirection table, e.g. setting index 0 to queue 5 and then index 5 to queue 0, the following command should be used:

```
sysctl dev.ena.0.rss.indir_table="0:5 5:0"
```

dev.ena.X.rss.key

RSS hash key. The default is 40 bytes long randomly generated hash key. Controls the RSS Toeplitz hash algorithm key value.

Only available when driver compiled without the kernel side RSS support.

Example: To change the RSS hash key value to

0x6d, 0x5a, 0x56, 0xda, 0x25, 0x5b, 0x0e, 0xc2,
0x41, 0x67, 0x25, 0x3d, 0x43, 0xa3, 0x8f, 0xb0,
0xd0, 0xca, 0x2b, 0xcb, 0xae, 0x7b, 0x30, 0xb4,
0x77, 0xcb, 0x2d, 0xa3, 0x80, 0x30, 0xf2, 0x0c,
0x6a, 0x42, 0xb7, 0x3b, 0xbe, 0xac, 0x01, 0xfa

the following command should be used:

```
sysctl dev.ena.0.rss.key=6d5a56da255b0ec24167253d43a38fb0d0ca2bcbae7b30b477cb2da38030f20c6a42b7
```

DIAGNOSTICS

Device initialization phase

ena%d: failed to init mmio read less

Error occurred during initialization of the mmio register read request.

ena%d: Can not reset device

Device could not be reset.
Device may not be responding or is already during reset.

ena%d: device version is too low

Version of the controller is too old and it is not supported by the driver.

ena%d: Invalid dma width value %d

The controller is unable to request dma transaction width.
Device stopped responding or it demanded invalid value.

ena%d: Can not initialize ena admin queue with device

Initialization of the Admin Queue failed.
Device may not be responding or there was a problem with initialization of the resources.

ena%d: Cannot get attribute for ena device rc: %d

Failed to get attributes of the device from the controller.

ena%d: Cannot configure aenq groups rc: %d

Errors occurred when trying to configure AENQ groups.

Driver initialization/shutdown phase

ena%d: PCI resource allocation failed!
ena%d: failed to pmap registers bar
ena%d: can not allocate ifnet structure
ena%d: Error with network interface setup
ena%d: Failed to enable and set the admin interrupts
ena%d: Error, MSI-X is already enabled

ena%d: Failed to enable MSIX, vectors %d rc %d
ena%d: Not enough number of MSI-X allocated: %d
ena%d: Error with MSI-X enablement
ena%d: could not allocate irq vector: %d
ena%d: unable to allocate bus resource: registers!
ena%d: unable to allocate bus resource: msix!

Resource allocation failed when initializing the device.
Driver will not be attached.

ena%d: ENA device init failed (err: %d)
ena%d: Cannot initialize device

Device initialization failed.
Driver will not be attached.

ena%d: failed to register interrupt handler for irq %ju: %d

Error occurred when trying to register Admin Queue interrupt handler.

ena%d: Cannot setup mgmnt queue intr

Error occurred during configuration of the Admin Queue interrupts.

ena%d: Enable MSI-X failed

Configuration of the MSI-X for Admin Queue failed.
There could be lack of resources or interrupts could not have been configured.
Driver will not be attached.

ena%d: VLAN is in use, detach first

VLANs are being used when trying to detach the driver.
VLANs must be detached first and then detach routine have to be called again.

ena%d: Unmapped RX DMA tag associations
ena%d: Unmapped TX DMA tag associations

Error occurred when trying to destroy RX/TX DMA tag.

ena%d: Cannot init indirect table
ena%d: Cannot fill indirect table
ena%d: Cannot fill hash function
ena%d: Cannot fill hash control
ena%d: WARNING: RSS was not properly initialized, it will affect bandwidth

Error occurred during initialization of one of RSS resources.
The device will work with reduced performance because all RX packets will be passed to queue 0 and there will be no hash information.

ena%d: LLQ is not supported. Fallback to host mode policy.
ena%d: Failed to configure the device mode. Fallback to host mode policy.
ena%d: unable to allocate LLQ bar resource. Fallback to host mode policy.

Error occurred during Low-latency Queue mode setup.
The device will work, but without the LLQ performance gain.

ena%d: failed to enable write combining.

Error occurred while setting the Write Combining mode, required for the LLQ.

ena%d: failed to tear down irq: %d
ena%d: dev has no parent while releasing res for irq: %d Release of the interrupts failed.

Additional diagnostic
ena%d: Invalid MTU setting. new_mtu: %d max_mtu: %d min mtu: %d

Requested MTU value is not supported and will not be set.

ena%d: Failed to set MTU to %d

This message appears when either MTU change feature is not supported, or device communication error has occurred.

ena%d: Keep alive watchdog timeout.

Device stopped responding and will be reset.

ena%d: Found a Tx that wasn’t completed on time, qid %d, index %d.

Packet was pushed to the NIC but not sent within given time limit.
It may be caused by hang of the IO queue.

ena%d: The number of lost tx completion is above the threshold (%d > %d). Reset the device

If too many Tx weren't completed on time the device is going to be reset.
It may be caused by hanged queue or device.

ena%d: Trigger reset is on

Device will be reset.
Reset is triggered either by watchdog or if too many TX packets were not completed on time.

ena%d: device reset scheduled but trigger_reset is off

Reset task has been triggered, but the driver did not request it.
Device reset will not be performed.

ena%d: Device reset failed

Error occurred while trying to reset the device.

ena%d: Cannot initialize device
ena%d: Error, mac address are different
ena%d: Error, device max mtu is smaller than ifp MTU
ena%d: Validation of device parameters failed
ena%d: Enable MSI-X failed
ena%d: Failed to create I/O queues
ena%d: Reset attempt failed. Can not reset the device

Error occurred while trying to restore the device after reset.

ena%d: Device reset completed successfully, Driver info: %s

Device has been correctly restored after reset and is ready to use.

ena%d: Allocation for Tx Queue %u failed
ena%d: Allocation for Rx Queue %u failed
ena%d: Unable to create Rx DMA map for buffer %d
ena%d: Failed to create io TX queue #%d rc: %d

ena%d: Failed to get TX queue handlers. TX queue num %d rc: %d
ena%d: Failed to create io RX queue[%d] rc: %d
ena%d: Failed to get RX queue handlers. RX queue num %d rc: %d
ena%d: could not allocate irq vector: %d
ena%d: failed to register interrupt handler for irq %ju: %d

IO resources initialization failed.
Interface will not be brought up.

ena%d: LRO[%d] Initialization failed!

Initialization of the LRO for the RX ring failed.

ena%d: failed to alloc buffer for rx queue
ena%d: failed to add buffer for rx queue %d
ena%d: refilled rx qid %d with only %d mbufs (from %d)

Allocation of resources used on RX path failed.
If happened during initialization of the IO queue, the interface will not be brought up.

ena%d: NULL mbuf in rx_info

Error occurred while assembling mbuf from descriptors.

ena%d: tx_info doesn’t have valid mbuf
ena%d: Invalid req_id: %hu
ena%d: failed to prepare tx bufs

Error occurred while preparing a packet for transmission.

ena%d: ioctl promisc/allmulti

IOCTL request for the device to work in promiscuous/allmulti mode.
See ifconfig(8) for more details.

SUPPORT

If an issue is identified with the released source code with a supported adapter, please email the specific information related to the issue to <*akiyano@amazon.com*>, <*osamaabb@amazon.com*> and <*darinzon@amazon.com*>.

SEE ALSO

netmap(4), vlan(4), ifconfig(8)

HISTORY

The **ena** driver first appeared in FreeBSD 11.1.

AUTHORS

The **ena** driver was developed by Amazon and originally written by Semihalf.