Urika®-GX Aries Network Resiliency Guide

(2.2.UP00)

S-3018

Contents

1 About the Urika®-GX Network Resiliency Guide

This publication contains administrative information about performing network resiliency related tasks on the Cray Urika®-GX system.

Typographic Conventions

`Monospace`	Indicates program code, reserved words, library functions, command-line prompts, screen output, file/path names, key strokes (e.g., `Enter` and `Alt-Ctrl-F`), and other software constructs.
`Monospaced Bold`	Indicates commands that must be entered on a command line or in response to an interactive prompt.
`Oblique` or *`Italics`*	Indicates user-supplied values in commands or syntax definitions.
Proportional Bold	Indicates a graphical user interface window or element.
\ (backslash)	At the end of a command line, indicates the Linux® shell line continuation character (lines joined by a backslash are parsed as a single line). Do not type anything after the backslash or the continuation feature will not work correctly.

Record of Revision

Release Date	Release
September, 2018	`2.2UP00`
May, 2018	`2.1UP00`
December, 2017	`2.0UP00`
April, 2017	`1.2UP00`
December, 2016	`1.1UP00`
August, 2016	`1.0UP00`
March, 2016	`0.5UP00`

Scope and Audience

The intended audience of this guide is system administrators and end users of the Urika®-GX system.

Record of Revision

Editorial changes.

Trademarks

Feedback

Visit the Cray Publications Portal at *http://pubs.cray.com* and make comments online using the *Contact Us* button in the upper-right corner or Email *pubs@cray.com*. Your comments are important to us and we will respond within 24 hours.

2 Network Congestion Causes

Congestion on the Aries™ network may result from a variety of causes. Congestion occurs most often when application programmers use one-sided programming models such as PGAS (Partitioned Global Address Space) and Cray SHMEM with applications that perform many- or all-to-one communication. Missing or compromised lanes, routes, links and channels on the HSN can also cause congestion. It is possible that the Lustre Network Driver (`kgnilnd`) or any of the higher level software that relies on it (e.g., Lustre and Data Virtualization Services) can cause congestion.

3 Return a Dual Aries Network Card (dANC) into Service

About this task

After a dANC has been repaired or when a replacement dANC is available, use the following procedure to return the dANC into service.

Procedure

1. Physically insert the dANC into the slot. To complete this step, see the hardware maintenance and replacement procedures documentation for the Cray system, or contact a Cray Service representative. In the following instructions, *dANC_ID* is a comma separated list of all the dANC IDs. in the rack.

2. On the SMW, execute the `xtcli power up` *dANC_ID* command.

   ```
   smw:~> xtcli power up dANC_ID
   ```

3. Ensure that the dANC is ready by entering the following command, and wait until the command returns the correct response:

   ```
   smw:~> xtalive dANC_ID
   The expected response was received.
   ```

4. Verify the status of the Aries Network Card Controller (ANCC) to ensure that its "`Comp state`" is "`ready`" and that there are no flags set.

   ```
   smw:~> xtcli status -t bc dANC_ID
   ```

5. Bounce the dANC.

   ```
   smw:~> xtbounce dANC_ID
   ```

6. Execute the `rackfw` command to update the ANCC BIOS, node BIOS, microcontroller, and FPGAs as required.

   ```
   smw:~> rackfw -t danc dANC_ID
   ```

7. Execute the `xtbounce --linkdown` *dANC_ID* command to prepare the dANC for the warm swap (takes down all HSN links on the dANC).

   ```
   smw:~> xtbounce --linkdown dANC_ID
   ```

8. Add the dANC(s) to the HSN by executing the `xtwarmswap --add` *dANC_ID*`,...` command. This command activates routing on the newly installed dANC and automatically executes a mini-`xtdiscover`

command once the warm swap steps have completed successfully. No additional manual invocation of `xtdiscover`, which gets the new hardware attributes from the added dANCs, is necessary.

```
smw:~> xtwarmswap --add dANC_ID
```

Because the `xtwarmswap --add` command initializes the added dANCs, the time to return the dANCs back to service is about 10 minutes, including the time to initialize the dANCs and initialize the links to the dANCs.

9. Boot the nodes.

4 About Link Resiliency

The Aries™ high-speed network (HSN) consists of lanes and channels. A lane provides bi-directional communication between two ports and a channel is three lanes. A Link Control Block (LCB) facilitates the transport of data across asynchronous chip boundaries by providing buffering and handshake support logic between higher-level network core logic and lowest-level channel protocol.

Cray XC series systems, which use the Aries™ interconnect technology, have hardware and software support that allows the system to handle certain types of hardware failures without requiring a system reboot. In addition, the same technology allows for the removal and replacement of compute blades without a system reboot. These features contribute to a reduction in both planned and unplanned system downtime.

Cray® Urika®-GX systems, which use the Aries™ interconnect technology, have hardware and software support that allows the system to handle certain types of hardware failures without requiring a system reboot. In addition, the same technology allows for the removal and replacement of network cards without a system reboot. These features contribute to a reduction in both planned and unplanned system downtime.

In the case of loss of power to an Aries™ Network Card (ANC), blade, set of blades, or the warm swap out of a compute blade, applications running on the affected blades will either be killed, or in the case of a warm swap out, be allowed to complete.

In the case of loss of power to a Dual Aries Network Card (dANC), or the warm swap out of a network card, applications running on nodes served by the affected dANCs will either be killed.

Warm swap of service blades is not supported.

Recovery from link failure is handled identically for compute blades and for service blades.

There are several components to Aries™ link resiliency:

- Hardware design that permits failed link detection and corrective action
- Software (`bcnwd`) on each blade controller (BC) that detects failed links and power loss to ANCs.
- Software (`anccnwd`) on each Aries Network Card Controller (ANCC) that detects failed links and power loss to dANCs.
- A daemon on the System Management Workstation (SMW) (`xtnlrd`) that coordinates the system response to failures.
- Another daemon on the SMW (`xthwerrlogd`) that logs hardware errors.
- An administrative command on the SMW (`xtwarmswap`) that facilitates warm swap of blades.
- An administrative command on the SMW (`xtwarmswap`) that facilitates warm swap of dANCs.

For more information, see the `xthwerrlogd`, `xtnlrd`, and `xtwarmswap` man pages.

When the Cray system is booted using the `xtbootsys` command, the `xtnlrd` and `xthwerrlogd` daemons are started on the SMW. Link monitoring on each BC is also enabled at this time, and link failures are logged by `xthwerrlogd` and responded to by `xtnlrd` rerouting the HSN around the failures.

When the SMW is booted, the `xtnlrd` and `xthwerrlogd` daemons are started. Link monitoring on each ANCC is started when a full system `xtbounce` is done, and link failures are logged by `xthwerrlogd` and responded to by `xtnlrd` rerouting the HSN around the failures.

4.1 Automatic Recovery of a Failed Single HSN Channel

When a single Aries™ high-speed network (HSN) channel fails, two link control blocks (LCBs) are reported as failed by `bcnwd` on the blade controllers (BCs) at each end of the channel. The failures are reported in the `xtnlrd` log file; for example:

```
2012-11-18 23:45:51 smw 8609 cb_hw_error: failed_component c0-0c0s4a0l30, type 37,
error_code 0x6218, error_category 0x0080
2012-11-18 23:45:51 smw 8609 cb_hw_error: failed_component c0-0c0s5a0l30, type 37,
error_code 0x620a, error_category 0x0080
2012-11-18 23:45:51 smw 8609 cb_hw_error: failed_component c0-0c0s9a0l45, type 37,
error_code 0x6218, error_category 0x0080
2012-11-18 23:45:51 smw 8609 cb_hw_error: failed_component c0-0c0s9a0l45, type 37,
error_code 0x620a, error_category 0x0080
```

When a single Aries™ high-speed network (HSN) channel fails, two link control blocks (LCBs) are reported as failed by `anccnwd` on the Aries Network Card Controllers (ANCCs) at each end of the channel. The failures are reported in the `xtnlrd` log file; for example:

```
2012-11-18 23:45:51 smw 8609 cb_hw_error: failed_component r0s0c0a0l30, type 37,
error_code 0x6218, error_category 0x0080
2012-11-18 23:45:51 smw 8609 cb_hw_error: failed_component r0s0c0a0l30, type 37,
error_code 0x620a, error_category 0x0080
2012-11-18 23:45:51 smw 8609 cb_hw_error: failed_component r0s0c0a0l45, type 37,
error_code 0x6218, error_category 0x0080
2012-11-18 23:45:51 smw 8609 cb_hw_error: failed_component r0s0c0a0l45, type 37,
error_code 0x620a, error_category 0x0080
```

These failures are followed by a series of steps (shown below) as the automatic recovery actions are performed. The automatic recovery steps typically complete in about 30 seconds. Each link endpoint with a fatal error has an alert flag set to indicate that the link is not available and should be bypassed.

The automatic recovery steps are visible in the `xtnlrd` log file. In summary, these steps are:

- `initial`: Waits for failures
- `aggregate_failures`: Waits 10 seconds by default for any more links to fail
- `link_failed`: Begins to process the failed links
- `alive`: Determines which blades are alive

  When power loss to a blade occurs, no response to the `alive` request is received; instead a 30-second timeout occurs for the blade that lost power.

  When power loss to a cabinet occurs, no response to the `alive` request is received; instead cabinet power loss results in a maximum of 960 LCB failures being reported to `xtnlrd`. The exact number depends on the topology class.
- `alive`: Determines which Aries Network Card Controllers (ANCCs) are alive

  When power loss to a dANC occurs, no response to the `alive` request is received; instead a 30-second timeout occurs for the dANC that lost power.
- `route_compute`: Computes and stages new routes to the BCs
- `route_compute`: Computes and stages new routes to the Aries Network Card Controllers (ANCCs)
- `quiesce`: Stops all high-speed network traffic temporarily
- `switch_netwatch`: Causes the `netwatch` daemon (`bcnwd`) on the blades to use the new routes file

- `switch_netwatch`: Causes the `netwatch` daemon (`anccnwd`) on the dANCs to use the new routes file
- `down_unused_links`: Causes links that are up but unused to be taken down
- `route_install`: Asserts new routes in the Aries™ chips
- `unquiesce`: Resumes all high-speed network traffic
- `finish`: Performs final cleanup
- `initial`: Waits for failures (process restarts)

4.2 Automatic Recovery of a Failed High-speed Network Cable

Aries™ network connections are divided into three groups: green links (15 per Aries ASIC), black links (15 per Aries ASIC), and blue links (10 per Aries ASIC). Green links are electrical connections over the backplane to other Aries in the same chassis. Black links are electrical connections over cables to other chassis within the same cabinet group. Blue links are optical connections over fiber to other cabinet groups.

Aries™ network connections are divided into two groups: green links (15 per Aries ASIC) and black links (15 per Aries ASIC). Green links are electrical connections over the backplane to other Aries in the same chassis. Black links are electrical connections over cables to other chassis within the same cabinet group.

Loss of a black network cable results in the failure of at most 6 LCBs, whereas loss of a blue network cable results in the failure of at most 8 LCBs. These appear in the `xtnlrd` log file as two entries per failed LCB, and each of the failed LCBs has an alert flag set. These failures are followed by a series of steps (shown below) as the automatic recovery actions are performed. The automatic recovery steps typically complete in about 30 seconds.

Loss of a black network cable results in the failure of at most 6 LCBs. These appear in the `xtnlrd` log file as two entries per failed LCB, and each of the failed LCBs has an alert flag set. These failures are followed by a series of steps (as following) as the automatic recovery actions are performed. The automatic recovery steps typically complete in about 30 seconds.

The automatic recovery steps are visible in the `xtnlrd` log file. In summary, these steps are:

- `initial`: Waits for failures
- `aggregate_failures`: Waits 10 seconds by default for any more links to fail
- `link_failed`: Begins to process the failed links
- `alive`: Determines which blades are alive

  When power loss to a blade occurs, no response to the `alive` request is received; instead a 30-second timeout occurs for the blade that lost power.

  When power loss to a cabinet occurs, no response to the `alive` request is received; instead cabinet power loss results in a maximum of 960 LCB failures being reported to `xtnlrd`. The exact number depends on the topology class.
- `alive`: Determines which Aries Network Card Controllers (ANCCs) are alive

  When power loss to a dANC occurs, no response to the `alive` request is received; instead a 30-second timeout occurs for the dANC that lost power.
- `route_compute`: Computes and stages new routes to the BCs
- `route_compute`: Computes and stages new routes to the Aries Network Card Controllers (ANCCs)
- `quiesce`: Stops all high-speed network traffic temporarily

- `switch_netwatch`: Causes the `netwatch` daemon (`bcnwd`) on the blades to use the new routes file
- `switch_netwatch`: Causes the `netwatch` daemon (`anccnwd`) on the dANCs to use the new routes file
- `down_unused_links`: Causes links that are up but unused to be taken down
- `route_install`: Asserts new routes in the Aries™ chips
- `unquiesce`: Resumes all high-speed network traffic
- `finish`: Performs final cleanup
- `initial`: Waits for failures (process restarts)

4.3 Automatic Recovery of Power Loss to a Dual Aries™ Network Card (dANC)

dANC power loss appears very similar to an Aries™ Network Card (ANC) power loss, except that no response to the `alive` request is received; instead a 30-second time out occurs for the dANC that lost power. The `alive` stage of the recovery process shows a time out for the dANC that lost power, such as:

```
2012-10-26 15:48:10 smw 52404 generic_rsp_timeout: ERROR: Did not receive responses
from
the following L0s: r0s1c0
2012-10-26 15:48:10 smw 52404 ***** dispatch: current_state check_alive *****
2012-10-26 15:48:10 smw 52404 add_blade_to_failed_list: adding blade r0s1c0 to
bladefailed list
```

The automatic recovery steps that are performed are provided below. Due to the time out, which is 30 seconds by default, recovery from a failed dANC typically takes about 60 seconds.

The automatic recovery steps are visible in the `xtnlrd` log file. In summary, these steps are:

- `initial`: Waits for failures
- `aggregate_failures`: Waits 10 seconds by default for any more links to fail
- `link_failed`: Begins to process the failed links
- `alive`: Determines which blades are alive

  When power loss to a blade occurs, no response to the `alive` request is received; instead a 30-second timeout occurs for the blade that lost power.

  When power loss to a cabinet occurs, no response to the `alive` request is received; instead cabinet power loss results in a maximum of 960 LCB failures being reported to `xtnlrd`. The exact number depends on the topology class.
- `alive`: Determines which Aries Network Card Controllers (ANCCs) are alive

  When power loss to a dANC occurs, no response to the `alive` request is received; instead a 30-second timeout occurs for the dANC that lost power.
- `route_compute`: Computes and stages new routes to the BCs
- `route_compute`: Computes and stages new routes to the Aries Network Card Controllers (ANCCs)
- `quiesce`: Stops all high-speed network traffic temporarily
- `switch_netwatch`: Causes the `netwatch` daemon (`bcnwd`) on the blades to use the new routes file
- `switch_netwatch`: Causes the `netwatch` daemon (`anccnwd`) on the dANCs to use the new routes file

- `down_unused_links`: Causes links that are up but unused to be taken down
- `route_install`: Asserts new routes in the Aries™ chips
- `unquiesce`: Resumes all high-speed network traffic
- `finish`: Performs final cleanup
- `initial`: Waits for failures (process restarts)

4.4 Automatic Recovery of Power Loss to an Aries™ Network Card on a Dual Aries Network Card (dANC)

Loss of power to an Aries™ Network Card (ANC) results in at most 25 LCB failures being reported by `xtnlrd`. Urika-GX systems use a class 0 topology. Although the endpoints on the dANC whose ANC lost power are not reported as failed, the ANC on that dANC is reported and alert flags are set on both the ANC and all LCBs that were reported as failed. This results in the entire dANC being routed around.

The following output appears in the `xtnlrd` log file, including a report of the dANC that lost ANC power.

```
2012-10-26 16:20:21 smw 52404 ***** dispatch: current_state alive *****
2012-10-26 16:20:21 smw 52404 INFO: 47 out of 48 L0s are alive
2012-10-26 16:20:21 smw 52404 Beginning to wait for response(s)
2012-10-26 16:20:21 smw 52404 svid 0:r0s1c0 (131282: Mezzanine Voltage Fault)
2012-10-26 16:20:21 smw 52404 Received 47 of 47 responses
2012-10-26 16:20:21 smw 52404 ***** dispatch: current_state check_alive *****
2012-10-26 16:20:21 smw 52404 MODULE ERROR: r0s1c0 - 210 - Mezzanine Voltage Fault
2012-10-26 16:20:21 smw 52404 add_blade_to_list: adding blade r0s1c0 to bladefailed
list
```

The steps that are automatically performed to reroute the HSN are provided below. The automatic recovery steps typically complete in about 30 seconds.

The automatic recovery steps are visible in the `xtnlrd` log file. In summary, these steps are:

- `initial`: Waits for failures
- `aggregate_failures`: Waits 10 seconds by default for any more links to fail
- `link_failed`: Begins to process the failed links
- `alive`: Determines which blades are alive

  When power loss to a blade occurs, no response to the `alive` request is received; instead a 30-second timeout occurs for the blade that lost power.

  When power loss to a cabinet occurs, no response to the `alive` request is received; instead cabinet power loss results in a maximum of 960 LCB failures being reported to `xtnlrd`. The exact number depends on the topology class.
- `alive`: Determines which Aries Network Card Controllers (ANCCs) are alive

  When power loss to a dANC occurs, no response to the `alive` request is received; instead a 30-second timeout occurs for the dANC that lost power.
- `route_compute`: Computes and stages new routes to the BCs
- `route_compute`: Computes and stages new routes to the Aries Network Card Controllers (ANCCs)
- `quiesce`: Stops all high-speed network traffic temporarily

- `switch_netwatch`: Causes the `netwatch` daemon (`bcnwd`) on the blades to use the new routes file
- `switch_netwatch`: Causes the `netwatch` daemon (`anccnwd`) on the dANCs to use the new routes file
- `down_unused_links`: Causes links that are up but unused to be taken down
- `route_install`: Asserts new routes in the Aries™ chips
- `unquiesce`: Resumes all high-speed network traffic
- `finish`: Performs final cleanup
- `initial`: Waits for failures (process restarts)

4.5 Automatic xtnlrd Actions to Respond to Critical Aries™ ASIC Errors

In addition to link failure due to a bad Link Control Block (LCB), cable, Aries™ Network Card (ANC), blade, or cabinet, critical errors on the Aries ASIC are also monitored by `bcnwd` and responded to by `xtnlrd`.In addition to link failure due to a bad Link Control Block (LCB), cable, Aries™ Network Card (ANC), dANC, critical errors on the Aries ASIC are also monitored by `anccnwd` and responded to by `xtnlrd`. Depending on which logic block in the Aries reported the critical error, different actions are taken by `xtnlrd`. The following table explains the various actions:

Table 1. Automatic xtnlrd Actions to Respond to Critical Aries™ ASIC Errors

Block	Component	Action	Note
NIC	`rt_node`	Alert on node	
PI	`rt_node`	Alert on node	
LB_NIC	`rt_node`	Alert on node	Some LB errors related to a specific NIC or PI, so for these, an alert is set on the node.
LB	`rt_node`	Route around Aries	Other LB errors are Aries-wide
NL	`rt_node`	Route around Aries	Netlink errors can affect the whole Aries, but significant error analysis needs to be done to determine the scope of each NL error.
NT	`rt_aries_lcb`	Route around failed link	Do not consult the critical error table.
PT	`rt_aries_lcb`	Route around Aries	Ptile errors are not necessarily internal to that group of Ptiles. Need to distinguish between a critical Ntile error (row 0-4) where we just route around a bad link, and a critical Ptile error (row 5) where we route around the bad Aries.
RTR	`rt_aries`	Route around Aries	Router errors affect the whole Aries.
PCIe	`rt_node` `rt_aries_nic`	Alert on node	

4.6 Identify Unrouteable Configurations

Cray XC series systems have a dragonfly network topology that provides for less restrictive connectivity than a torus based network topology. Because of this, it is expected that there will also be fewer unrouteable configurations. The main requirement for any Cray XC series system configuration to be routeable is that it must be fully connected. Another characteristic of the configuration that is not strictly required, but is highly desirable, is that every group is connected to every other group. If this is not the case, performance between some pairs of groups can be greatly impacted.

Cray® Urika®-GX systems have a dragonfly network topology that provides for less restrictive connectivity than a torus based network topology. Because of this, it is expected that there will also be fewer unrouteable configurations. The main requirement for any Urika®-GX system configuration to be routeable is that it must be fully connected. Another characteristic of the configuration that is not strictly required, but is highly desirable, is that every group is connected to every other group. If this is not the case, performance between some pairs of groups can be greatly impacted.

In other cases however, the configuration per se is fine, but one blade is unable to complete the request due to some unexpected failure specific to the blade (or rarely, to multiple blades). Such blades are typically operating fine otherwise, and may be actively running jobs, but cannot complete the new routing operation. This can happen with any routing attempt; that is, it is not limited to the removal of components.

In other cases however, the configuration per se is fine, but one dANC is unable to complete the request due to some unexpected failure specific to the dANC (or rarely, to multiple dANCs). Such dANCs are typically operating fine otherwise, and may be actively running jobs, but cannot complete the new routing operation. This can happen with any routing attempt; that is, it is not limited to the removal of components.

If a warmswap operation fails during the test reroute stage, then something is preventing the operation from completing due to a failure to route the proposed new (that is, post-warmswap) configuration, and the following output is displayed, as in this example of a warmswap remove operation. Note that if a large number of blades encounter problems, the warmswap output might display only an overall error count.Note that if a large number of dANCs encounter problems, the warmswap output might display only an overall error count.

```
smw:~# xtwarmswap --remove c0-0c2s15
Adding 0 blades and removing 1 blade in partition p0
Removing blades:
   c0-0c2s15

Sending command to xtnlrd
17:43:35(T+00:00) Warm swap beginning
17:43:35(T+00:00) Setting alert flag on blades being removed
17:43:35(T+00:00) Turning link monitoring off for blades being removed
17:43:35(T+00:00)     Link monitoring turned off for blades being removed
17:43:35(T+00:00) Testing for routeability
17:43:36(T+00:01)     Test reroute proceeding...
17:43:46(T+00:11)     Test reroute proceeding...
17:43:56(T+00:21)     Test reroute proceeding...
17:44:06(T+00:31)     Test reroute proceeding...
17:44:16(T+00:41)     Test reroute proceeding...
17:44:26(T+00:51)     Test reroute proceeding...
17:44:35(T+01:00)     ERROR: test route exited with an error status
17:44:35(T+01:00)     Timeouts were reported on the following blade(s):
17:44:35(T+01:00)         c0-0c2s13
17:44:35(T+01:00) Clearing alert flags for blades that could not be removed
17:44:35(T+01:00)     Clearing alert flags on blades that failed to be removed
17:44:35(T+01:00) Turning link monitoring on for blades that could not be removed
```

```
17:44:35(T+01:00) Cleaning up

Daemon reported errno 11 (Resource temporarily unavailable)
Error text:
Error during route computation
1 of 45 blades reported timeouts during routing.
You may need to reboot these blade controllers, or warmswap the blades out,
before trying again.

FAILURE: Warm swap command failed.
```

```
smw:~# xtwarmswap --remove r0s1c0
Adding 0 blades and removing 1 blade in partition p0
Removing blades:
   r0s1c0

Sending command to xtnlrd
17:43:35(T+00:00) Warm swap beginning
17:43:35(T+00:00) Setting alert flag on blades being removed
17:43:35(T+00:00) Turning link monitoring off for blades being removed
17:43:35(T+00:00)     Link monitoring turned off for blades being removed
17:43:35(T+00:00) Testing for routeability
17:43:36(T+00:01)     Test reroute proceeding...
17:43:46(T+00:11)     Test reroute proceeding...
17:43:56(T+00:21)     Test reroute proceeding...
17:44:06(T+00:31)     Test reroute proceeding...
17:44:16(T+00:41)     Test reroute proceeding...
17:44:26(T+00:51)     Test reroute proceeding...
17:44:35(T+01:00)     ERROR: test route exited with an error status
17:44:35(T+01:00)     Timeouts were reported on the following blade(s):
17:44:35(T+01:00)         r0s1c0
17:44:35(T+01:00) Clearing alert flags for blades that could not be removed
17:44:35(T+01:00)     Clearing alert flags on blades that failed to be removed
17:44:35(T+01:00) Turning link monitoring on for blades that could not be removed
17:44:35(T+01:00) Cleaning up

Daemon reported errno 11 (Resource temporarily unavailable)
Error text:
Error during route computation
1 of 45 blades reported timeouts during routing.
You may need to reboot these blade controllers, or warmswap the blades out,
before trying again.

FAILURE: Warm swap command failed.
```

A failure could also occur during a warmswap add operation; for example:

```
smw:~# xtwarmswap --add c0-0c2s15
Adding 1 blade and removing 0 blades in partition p0
Adding blades:
    c0-0c2s15

Sending command to xtnlrd
17:48:59(T+00:00) Warm swap beginning
17:48:59(T+00:00) Clearing alert flags on blades being added
17:49:00(T+00:01)     Finished clearing link alerts
17:49:00(T+00:01) Testing for routeability
17:49:01(T+00:01)     ERROR: test route exited with an error status
17:49:01(T+00:01)     Failures were reported on the following blade(s):
17:49:01(T+00:01)         c0-0c2s13
```

```
17:49:01(T+00:01) Marking HSN links down on blades that could not be added
17:49:01(T+00:01) Correcting alert flags for blades or LCBs that could not be added
or removed
17:49:01(T+00:01)     Restoring alert flags for blades that failed to be added
17:49:01(T+00:01) Cleaning up

Daemon reported errno 11 (Resource temporarily unavailable)
Error text:
Error during route computation
1 of 45 blades reported errors during routing.
You may need to reboot these blade controllers, or warmswap the blades out,
before trying again.

FAILURE: Warm swap command failed.
```

```
smw:~# xtwarmswap --add r0s1c0
Adding 1 blade and removing 0 blades in partition p0
Adding blades:
    r0s1c0

Sending command to xtnlrd
17:48:59(T+00:00) Warm swap beginning
17:48:59(T+00:00) Clearing alert flags on blades being added
17:49:00(T+00:01)     Finished clearing link alerts
17:49:00(T+00:01) Testing for routeability
17:49:01(T+00:01)     ERROR: test route exited with an error status
17:49:01(T+00:01)     Failures were reported on the following blade(s):
17:49:01(T+00:01)         r0s1c0
17:49:01(T+00:01) Marking HSN links down on blades that could not be added
17:49:01(T+00:01) Correcting alert flags for blades or LCBs that could not be added
or removed
17:49:01(T+00:01)     Restoring alert flags for blades that failed to be added
17:49:01(T+00:01) Cleaning up

Daemon reported errno 11 (Resource temporarily unavailable)
Error text:
Error during route computation
1 of 45 blades reported errors during routing.
You may need to reboot these blade controllers, or warmswap the blades out,
before trying again.

FAILURE: Warm swap command failed.
```

Usually, this type of failure will result in the blades being automatically removed from the system if a critical resiliency operation is triggered to address a failed component. However, this is not the case for warmswap operations, as manual recovery may be possible without the blade's removal, allowing running jobs to be preserved and the blade to be left in the system. For the blade or blades that were reported as having failed or timed out, see *Recover a Blade That Failed to Respond to the Reroute Request*.

Usually, this type of failure will result in the dANCs being automatically removed from the system if a critical resiliency operation is triggered to address a failed component. However, this is not the case for warmswap operations, as manual recovery may be possible without the dANC's removal, allowing running jobs to be preserved and the dANC to be left in the system. For the dANC or dANCs that were reported as having failed or timed out, see *Recover a Dual Aries Network Card (dANC) that Failed to Respond to the Reroute Request* on page 17.

Note that these types of failures are normally expected for one or a very small number of blades. If large numbers of blades are indicated to have failed, then a more general routing issue may be the ultimate cause instead. This may be an indirect indication that the new configuration is not routable or may indicate a more general system issue.

Note that these types of failures are normally expected for one or a very small number of dANCs. If large numbers of dANCs are indicated to have failed, then a more general routing issue may be the ultimate cause instead. This may be an indirect indication that the new configuration is not routable or may indicate a more general system issue.

If a routing issue appears to have occurred, additional information may be found in the output of the `rtr` command corresponding to the failed test reroute. This output is recorded in the `xtnlrd` (`nlrd-`*`YYYYMMDD`*) log. Use the following command to display the relevant log entries:

```
smw:~# grep "CMD: rtr:" nlrd-YYYYMMDD
CMD: rtr: WARNING: c0-0c2s15: ERROR: <error-string>; procedure aborted
CMD: rtr: ERROR: 1 errors, rtr failed
CMD: rtr: WARNING: c0-0c0s4: bcrtr process died due to signal 6"
CMD: rtr: ERROR: 1 errors, rtr failed
CMD: rtr: WARNING: c0-0c0s1: bcrtr process exited abnormally with status 14"
CMD: rtr: ERROR: 1 errors, rtr failed
CMD: rtr: ERROR: Timeout while executing bcrtr -f /var/tmp/rtr-cfg.test -S --id
test --instance=0x82e5, didn't hear from 1 BC: c0-0c2s6
```

```
smw:~# grep "CMD: rtr:" nlrd-YYYYMMDD
CMD: rtr: WARNING: r0s1c0: ERROR: <error-string>; procedure aborted
CMD: rtr: ERROR: 1 errors, rtr failed
CMD: rtr: WARNING: r0s1c0: anccrtr process died due to signal 6"
CMD: rtr: ERROR: 1 errors, rtr failed
CMD: rtr: WARNING: r0s1c0: anccrtr process exited abnormally with status 14"
CMD: rtr: ERROR: 1 errors, rtr failed
CMD: rtr: ERROR: Timeout while executing anccrtr -f /var/tmp/rtr-cfg.test -S --id
test --instance=0x82e5, didn't hear from 1 BC: r0s1c0
```

4.6.1 Recover a Dual Aries Network Card (dANC) that Failed to Respond to the Reroute Request

Prerequisites

- The `xtwarmswap` command was executed and failed during the test reroute stage, and an otherwise routable individual dANC (or dANCs) fails to respond to the reroute request.
- The failed dANC was identified in the output of the failing `xtwarmswap` command.

About this task

For warmswap operations, a manual recovery of a dANC may be possible without the dANC's removal, allowing running jobs to be preserved and the dANC to be left in the system.

Procedure

1. Reboot the dANC controllers for the dANC or dANCs reporting problems.

   ```
   ssh root@dANC_ID reboot comma-separated list of non-responsive ANCC names
   ```

2. Rerun the previously-failing `xtwarmswap` command.

   ```
   smw:~# xtwarmswap --remove r0s1c0
   ```

3. If the command fails a second time due to a routing failure, note the still-failing dANC or dANCs reported in the `xtwarmswap` output. Then remove these dANCs with `xtwarmswap --remove` before reattempting the original warmswap operation.

4.7 Link Resiliency Parameters

Parameters in the `xtnlrd.ini` file control the system response to failures of link resiliency on the system.

`aggregate_timeout`

Seconds to wait after a link failure occurs before taking action to route around it, to allow any other link failures to arrive (default is `10` seconds)

`alive_timeout`

Seconds to wait for alive responses from blades (default is `30` seconds)

Seconds to wait for alive responses from dual Aries Network Cards (dANCs) (default is `30` seconds)

`init_new_blades_timeout`

Seconds to wait for blade initialization to finish (default is `1200` seconds)

Seconds to wait for dANC initialization to finish (default is `1200` seconds)

`init_new_links_timeout`

Seconds to wait for link initialization to finish (default is `600` seconds)

`quiesce_timeout`

Seconds to wait for quiesce to finish (default is `120` seconds)

`reroute_l0_timeout`

Seconds to wait for route requests to BCs to time out (default is `60` seconds)

`reroute_retries`

Number of route command retries on error (default is `1`)

`reroute_retry_wait`

Seconds between route command retries on failure (default is `5` seconds)

`reroute_smw_timeout`

Seconds to wait for route requests to the SMW to time out (default is `15` seconds)

`route_compute_timeout`

Seconds to wait for route computation to finish (default is `300` seconds)

`route_install_timeout`

Seconds to wait for route installation to finish (default is `60` seconds)

`unquiesce_timeout`

Seconds to wait for unquiesce to finish (default is `120` seconds)

`userexit_on_failure`

Calls user exit script `/opt/cray/hss/default/etc/xtnlrd.userexit` if link recovery fails (default is y)

4.8 Update the System Configuration while the Nodes are Booted

To change the system configuration physically while the nodes are booted, use the `xtwarmswap` command to remove or add one or more Dual Aries Network Cards (dANCs) or to remove or add a high-speed network (HSN) cable.

CAUTION: When warm swapping out a dANC, note that the 8 nodes served by that dANC will be unusable since their Aries network connections will go away. Before performing the warm swap remove, be sure to shut down the nodes.

The `xtwarmswap` command runs on the SMW and coordinates with the `xtnlrd` daemon to take the necessary steps to perform warm swap operations.

4.8.1 Reuse One or More Previously-failed HSN Links

About this task

Before failed links can be reintegrated into the HSN configuration, the alert flags on the LCBs that are to be reused must be cleared. The `xtnlrd` daemon will automatically do this when told to perform a warm swap `sync` operation.

To integrate failed links back into the HSN configuration, the `xtwarmswap` command may be invoked with one of the following:

- `-s` *`LCB, ...`*, specifying the list of LCBs to bring back up
- `-s all`, to bring in all available LCBs
- `-s none`, to cause a reroute without changing the LCBs that are in use

Procedure

1. Execute an `xtwarmswap -s` *`LCB_names`* `-p p0` to clear the alert flags on the specified LCBs and to reroute the HSN using those links.

   Note that by specifying `all` on the `xtwarmswap -s` option, all LCBs with alerts will be reused and, hence, their alert flags will be cleared. Alternatively, by specifying `none` on the `xtwarmswap -s` option, no LCB alerts will be cleared and the system will be rerouted without any changes to the configuration.

2. Execute an `xtwarmswap -s all -p p0` command to tell the system to reroute the HSN using all available links.

3. Execute an `xtwarmswap -s` *`LCB_names`* `-p` *`partition_name`* to clear the alert flags on the specified LCBs and to reroute the HSN using those links.

   Note that by specifying `all` on the `xtwarmswap -s` option, all LCBs with alerts will be reused and, hence, their alert flags will be cleared. Alternatively, by specifying `none` on the `xtwarmswap -s` option, no LCB alerts will be cleared and the system will be rerouted without any changes to the configuration.

4. Execute an `xtwarmswap -s LCB_names -p partition_name` to tell the system to reroute the HSN using the specified set of LCBs in addition to those that are currently in use.

   Doing so will clear the alert flags on the specified LCBs automatically. If the warm swap fails, the alert flag will be restored to the specified LCBs.

5. Execute an `xtwarmswap -s all -p partition_name` command to tell the system to reroute the HSN using all available links.

The `xtwarmswap` command results in `xtnlrd` performing the same link recovery steps as for a failed link, but with two differences: no alert flags are set, and an `init_new_links` and a `reset_new_links` step are performed to initialize both ends of any links to be used, before new routes are asserted into the Aries™ routing tables.

The elapsed time for the warm swap synchronization operation is typically about `30` seconds.

4.8.2 Reuse One or More Previously-failed Dual Aries Network Cards (dANCs)

About this task

Failed dANCs have alert flags set on the ASICs and the LCBs. These alert flags must be cleared before the dANCs can be reused.

Before previously-failed dANCs can be reused, the alert flags on their Aries ASICs and on the link endpoints (LCBs) must be cleared. This is done automatically by `xtnlrd` when told to perform a warm swap `add` operation.

Execute the `xtwarmswap --add` command to bring the dANCs back into the HSN configuration. Doing so clears any alert flags on the added dANCs and LCBs relating to those dANCs, initializes the dANCs, and initializes the links to the dANCs.

Procedure

1. Ensure that dANCs have power.

2. Ensure that an `xtalive` command to all required dANCs succeeds.

3. Add the dANC(s) to the HSN by executing the `xtwarmswap --add dANC_ID`,... command. Note that this command automatically executes a mini-`xtdiscover` command after the warm swap steps have completed successfully. No manual invocation of `xtdiscover`, which gets the new hardware attributes from the added dANCs, is necessary.

   Because the `xtwarmswap --add` command initializes the added dANCs, the time to return the dANCs back to service is about 10 minutes, including the time to initialize the dANCs, and initialize the links to the dANCs.

4. Boot the nodes.

4.8.3 Remove a Dual Aries Network Card (dANC) from Service While the System is Running

About this task

A dANC can be physically removed for maintenance or replacement while the system is running; however, the applications using the nodes served by the dANC to be removed must be allowed to drain, or be killed beforehand.

Verify that the proposed system configuration is routable prior to starting this warm swap procedure. Doing this in advance of idling the nodes on the dANCs to be removed provides assurance that a valid set of nodes is being taken out of service before affecting the system. Log on to the SMW as `crayadm` and execute the following command, where `p0` is the partition from which the dANCs are being removed:

```
smw:~> rtr -S --id test --remove=dANC_ID,dANC_ID,dANC_ID,... p0
```

Procedure

1. Shut down the affected nodes using instructions documented in *Cray® Urika®-GX System Administration Guide* ".

2. Execute the `xtwarmswap --remove` *dANC_ID* command to remove the dANC from service. The routing of the Cray HSN will be updated to route around the removed dANC.

   The `--remove` stage of the `xtwarmswap` process uses the Aries™ resiliency infrastructure and takes about 30 seconds to complete.

   ```
   smw:~> xtwarmswap --remove dANC_ID
   ```

3. Execute the `xtcli power down` *dANC_ID* command, which helps to identify which dANC to pull (all lights are off on the dANC).

   ```
   smw:~> xtcli power down dANC_ID
   ```

4. Physically remove the dANC, if desired. To complete this step, see the hardware maintenance and replacement procedures documentation for the Cray system, or contact a Cray Service representative.

4.9 Identify Lane Degrades

Each High Speed Network (HSN) link is comprised of three SerDes lanes in each direction. Under normal circumstances, all three lanes are operational and running cleanly. The Aries™ SerDes and LCB hardware do a very good job of keeping the lanes and links alive, even in the presence of soft errors. There can, however, be cases where lane hardware is faulty, such as a bad cable. In those cases, one or more lanes can give up and shut down due to excessive errors. This is due to bad signal paths in the HSN hardware. The link itself will continue to run at reduced bandwidth as long as there is at least one lane running. If a lane degrades, the hardware involved needs to be serviced.

Use the `xtnetwatch` log to identify degraded lanes. In this example, `lanemask=6`, which means that lane 0 is down.

```
130625 09:41:35  ##############  ##############  ###############  ################
130625 09:41:35  LCB ID          Peer LCB        Category         Description
130625 09:41:35  ##############  ##############  ###############  ##############
130625 13:30:18  c0-0c2s10a0l23  c1-0c1s10a0l22  Info             RX Lane
Degrade,
lanemask: 6
130625 13:30:18  c0-0c2s10a0l23  c1-0c1s10a0l22  Info             TX Lane
Degrade,
lanemask: 6
```

```
130625 09:41:35  ##############  ##############  ###############  ################
130625 09:41:35  LCB ID          Peer LCB        Category         Description
130625 09:41:35  ##############  ##############  ###############  ##############
130625 13:30:18  r0s0c1a1l44     unknown         Info             RX Lane
Degrade,
lanemask: 6
130625 13:30:18  r0s0c1a1l45     unknown         Info             TX Lane
Degrade,
lanemask: 6
```

5 About Network Congestion

Network congestion is a condition that occurs when the volume of the traffic on the high-speed network (HSN) exceeds the network's capacity to handle it. This causes traffic in the network to stall and make extremely slow forward progress. In and of itself, network congestion is not a problem and is not unique to any particular network. However, in the rare case of extreme congestion, problems can arise that may affect system performance.

On the Aries™ network, packets must reach their destinations quickly enough to avoid various higher level timeouts and to enable reasonable transit latencies. The Outstanding Request Buffer (ORB) is a logic block in the Aries ASIC that tracks sent packets. The ORB waits for response packets to match the extant entries. If the sending ORB does not receive a response packet within some number of seconds then a timeout occurs. Under normal conditions, this implies a lost packet that is then reported to higher-level clients for handling. If a timeout occurs, the Hardware Supervisory System (HSS) on the blade detects the condition and executes an algorithm to safely scrub the timed-out ORB entries.If a timeout occurs, the Hardware Supervisory System (HSS) on the dual Aries Network Card (dANC) detects the condition and executes an algorithm to safely scrub the timed out ORB entries. The scrub algorithm includes a 10-second quiesce of the Network Interface Card (NIC) whose ORB indicated a timeout condition. When the HSN experiences congestion, however, a timeout may not actually reflect a lost packet. Instead, in a congested network, the response packet's latency will increase, which is the motivation for quiescing the NIC to ensure only truly lost packets get reported.

5.1 Congestion Management Software Components

To manage congestion on the Aries™ network, HSS software monitors and throttles traffic injected into the network when necessary. Congestion protection is implemented by two daemons: one on the SMW, called `xtnlrd`, and one on the BC controllers, called `bcbwtd`. It limits the aggregate injection bandwidth across all compute nodes to less than the ejection bandwidth of a single node. This alleviates congestion in the system, which in turn has the effect of reducing network latency. This trade-off is acceptable because Congestion Protection is active only in cases of extreme congestion.

To manage congestion on the Aries™ network, HSS software monitors and throttles traffic injected into the network when necessary. This congestion protection is implemented by two daemons: one on the SMW, called `xtnlrd`, and one on the Aries Network Card Controllers (ANCCs), called `anccbwtd`. It limits the aggregate injection bandwidth across all compute nodes to less than the ejection bandwidth of a single node. This alleviates congestion in the system, which in turn has the effect of reducing network latency. This trade-off is acceptable because Congestion Protection is active only in cases of extreme congestion.

5.2 How Congestion Management Software Detects Network Congestion

The `bcbwtd` daemon running on the BC monitors the percentage of time that traffic trying to enter the network from the nodes attached to the BC is stalled and the percentage of time network tiles are stalled. When these percentages cross a high water mark threshold, `bcbwtd` communicates that information to the `xtnlrd` daemon running on the SMW. After the congestion subsides, `bcbwtd` again relays this information to the SMW. When the SMW receives a number of notifications of congestion sufficient to cross another high water mark threshold, Congestion Protection is enabled.

The `anccbwtd` daemon running on the Aries Network Card Controller (ANCC) monitors the percentage of time that traffic trying to enter the network from the nodes attached to the BC is stalled and the percentage of time network tiles are stalled. When these percentages cross a high water mark threshold, `anccbwtd` communicates that information to the `xtnlrd` daemon running on the System Management Workstation (SMW). After the congestion subsides, `anccbwtd` again relays this information to the SMW. When the SMW receives a number of notifications of congestion sufficient to cross another high water mark threshold, congestion protection is enabled.

Congestion protection remains active until congestion subsides and the number of congested tiles and nodes drops below their respective low water mark thresholds. If an application (or combination of applications) persistently congests the network, Congestion Protection is reapplied and released periodically until the application terminates.

5.3 Congestion Management Log File and Messages

The daemon handling congestion management on the SMW, called `xtnlrd`, produces a log file that includes:

- Following an increase in overall system congestion, a chronological, component-based history of each congestion event is generated, ranging from the first detection of congestion to the time total system congestion again returns to zero (non-increases are omitted to help readability; that is, only up-ticks are logged).
- Because once congestion begins, it can spread to its neighbors quickly, this history is displayed in the order in which `xtnlrd` receives the congestion events from the individual blades.Because once congestion begins, it can spread to its neighbors quickly, this history is displayed in the order in which `xtnlrd` receives the congestion events from the individual dual Aries Network Cards (dANCs). The congestion history is logged for all incidents of detected congestion, even those that do not reach the threshold for throttling the system. This history includes the network ASIC reporting the congestion, how much was reported, and which NICs were congested, as well as the resulting effect on the overall system tile and NIC congestion. The congestion history is displayed following the system's return to zero overall reported congestion.
- Events received from the Application Level Placement Scheduler (ALPS) when an application starts and ends.
- A periodic update showing all applications that are running, repeated every five minutes, or whenever a congestion protection event occurs.
- A list of the top 10 applications by aggregate ejection bandwidth (e.g., the per-node ejection bandwidth summed across all nodes in the application) whenever a congestion protection event occurs.
- A list of the top 10 nodes by ejection flit counts whenever a congestion protection event occurs, which is useful for identifying certain congesting many-to-few communications patterns.
- The applications running on those nodes, including APID, number of nodes, the user ID, and the application name.

The `xtnlrd` log file is located in `/var/opt/cray/log/`*`session-id`*`/nlrd.`*`session-id`*.

The `xtnlrd` log file is located in `/var/opt/cray/log/p0-default/nlrd.p0-`*`datetime`*.

The following log messages are produced in the `xtnlrd` log file when a throttle event is triggered and additional information is gathered about the event:

```
2012-02-09 19:51:01 smw check_for_throttle_action: tile_weight 32 nic_weight 1
2012-02-09 19:51:01 smw do_throttle: Telling all blades to throttle network
bandwidth
2012-02-09 19:51:01 smw _do_throttle: sending throttle mask 0xf0xff to 16326 blades
2012-02-09 19:51:06 smw Executing: Bandwidth data gathering
2012-02-09 19:51:11 smw Command succeeded.
2012-02-09 19:51:11 smw Executing: Unthrottle service blades only
2012-02-09 19:51:13 smw Command succeeded.
```

In addition, any time a congestion protection event occurs, a message like the following is inserted by `aprun` into `/var/opt/cray/log/p`*`n`*`-current/messages-`*`YYYYMMDD`*. A similar message is also written to the user's standard out:

```
Feb 9 11:32:40 nid00002 aprun[14718]: apid=6204, Network throttled,
user=10320, batch_id=unknown, nodes_throttled=20, node_seconds=00:01:50
```

5.4 Possible Critical Errors Resulting from Network Congestion

In the rare event that the software is unable to successfully protect the system, a delayed response for a timed-out packet can cause an ORB critical error event. An ORB critical error event will abort the application and provide messages for the user and administrator. The following message will appear on the application's `stderr` during ORB error events:

```
Cray HSN detected critical error 0xNNNptag N.
```

where the particular HSN critical errors (0x*NNN*) that may result from excessive network congestion are:

- `0x4801: ORB Request with No Entry` - This indicates that there was no matching entry for the response packet within the ORB.
- `0x4802: ORB Command Mismatch` - This indicates that the command stored in the ORB entry is different than the command indicated by the response packet.
- `0x480a: ORB DWORD Flit Mismatch` - This indicates that the response packet size is different than the corresponding entry in the ORB.

CAUTION: If administrators or application programmers receive reports of the above errors, contact a Cray service representative for more information.

To learn how application programmers can modify their programs to prevent these types of errors on Cray XC series systems, see *Aries™ Network Tech Note - Application Changes to Avoid Network Congestion (S-0048)*.

5.5 SMW Network Congestion Parameters

Parameters in the `xtnlrd.ini` file control the system response to failures of network congestion on the system.

`bw_congest_kill_top_app`

Controls whether `xtnlrd` should tell ALPS to kill the top congesting application after `bw_congest_max_consecutive_hits` times at the top of the list (default is n)

`bw_congest_max_consecutive_hits`

Number of times an APID has to be at the top of the congestion list to be eligible to be killed (default is `2`)

`bw_congest_throttle_delay`

Seconds to wait before the parent `xtnlrd` gathers ejection bandwidth data (default is `5`)

`bw_congest_top_nodes`

Number of non-service nodes to log as having high ejection bandwidth (default is `10`)

`bw_throttle_nic_enable`

Throttle based on NIC congestion (default is `y`)

`bw_throttle_nic_hi_num`

Total number of congested NICs to initiate throttling (default is `10`)

`bw_throttle_nic_lo_num`

Total number of congested NICs before unthrottling is done (default is `6`)

`bw_throttle_tile_enable`

Throttle based on tile congestion (default is `y`)

`bw_throttle_tile_hi_num`

Total number of congested tiles to initiate throttling (default is `20`)

`bw_throttle_tile_lo_num`

Total number of congested tiles before unthrottling is done (default is `12`)

`bw_unthrottle_interval`

Minimum seconds for a throttle to persist (default is `20` seconds)

5.6 Aries™ Network Card Controller (ANCC) Network Congestion Parameters

Parameters in the `xtnlrd.ini` file control the system response to failures of network congestion on the system.

`bw_em_sample_interval_sec`

Seconds between ejection bandwidth samples on a ANCC; the default is `4`

`bw_qos_rate_b1=511999`

Do not change; this parameter is related to the unthrottled and throttled Aries™ traffic shaping control settings

`bw_qos_rate_b2=5119`

Do not change; this parameter is related to the unthrottled and throttled Aries™ traffic shaping control settings

`bw_qos_size_b1=17`

Do not change; this parameter is related to the unthrottled and throttled Aries™ traffic shaping control settings

`bw_qos_size_b2=1`

Do not change; this parameter is related to the unthrottled and throttled Aries™ traffic shaping control settings

`bw_throttle_heartbeat_misses`

Number of missed `xtnlrd` heartbeats before `gmbwtd` locally throttles that Dual Aries Network Card (dANC). Receipt of a heartbeat while locally throttled causes an unthrottle; the default is `3`

`bw_throttle_heartbeat_sec`

Time in seconds between heartbeats from `xtnlrd` to `gmbwtd` to indicate the SMW and HSS network is still alive. These heartbeats are intended to protect the system from HSN congestion if the HSS system is dead (`0 == off`); the default is `3`

`bw_throttle_nic_history`

Number of sample periods in a row before sending a congestion notification (`ec_l0_bw_throttle`); the default is `3`

`bw_throttle_node_stall_thresh`

Ration of stalls to forwarded flits in the NIC; the default is `200000`

`bw_throttle_sample_usec`

Sample rate, in microseconds, to sample MMRs for congestion detection; the default is `500000`

`bw_throttle_tile_vc0_stall_thresh`

Ratio of stalls to forwarded flits on the request channel; the default is `200000`

`bw_throttle_tile_history`

Number of sample periods in a row before sending a congestion notification (`ec_l0_bw_throttle`); the default is `3`