175 Cabot Street – Suite 210
Lowell, MA 01854 USA
www.cspi.com
NASDAQ: CSPI

What RSS hashing is available in Sniffer10G?

Model:

ARC Series E Adapters

Software:

Sniffer10G

Operating System:

Supports both Windows and Linux Operating Systems.

Information:

The Receive-Side Scaling (RSS) hashing flags are set via the environment variable **SNF_RSS_FLAGS.**

SNF_RSS_FLAGS can be specified to let the implementation know which IP/TCP/UDP fields are significant when generating the hash. By default, RSS is computed on IPv4/IPv6 addresses and source/destination ports when the protocol is TCP or UDP.

Sniffer10G V3 Receive-Side Scaling (RSS) Hashing

By default, SNF_RSS_FLAGS = (SNF_RSS_IP | SNF_RSS_SRC_PORT | SNF_RSS_DST_PORT | SNF_RSS_GTP | SNF_RSS_GRE).

In **snf.h** there is this definition:

```
enum snf_rss_mode_flags {
  SNF_RSS_IP        =  0x01, /**< Include IP (v4 or v6) SRC/DST addr in hash */
  SNF_RSS_SRC_PORT  =  0x10, /**< Include TCP/UDP/SCTP SRC port in hash */
  SNF_RSS_DST_PORT  =  0x20, /**< Include TCP/UDP/SCTP DST port in hash */
  SNF_RSS_GTP       =  0x40, /**< Include GTP TEID in hash */
  SNF_RSS_GRE       =  0x80, /**< Include GRE contents in hash */};
```

If the packet is an IP packet and SNF_RSS_IP is desired, the IPs will be considered in the hash. If the IP packet is UDP or TCP or SCTP, then the ports are considered in the hashing. If all the traffic is tunneled through GTP or GRE protocols, the SNF_RSS_GTP or SNF_RSS_GRE flags can be used to consider elements of those headers in the hash. When SNF_RSS_GTP is set, the 'TEID' is used as part of the hash if the first payload bye indicated that the packet is a GTP version 1 or GTP version 2 packet. If SNF_RSS_GRE is used, the headers of the packet encapsulated in the GRE payload will be used in hashing.

We parse VLAN but then any RSS is applied only if what follows is an IPv4/IPv6 header that is followed by TCP/UDP header. This means that any form of TCP/UDP tunneling considers only the outermost headers.

For Ethernet frames bearing “non-hashable” traffic (e.g., MPLS or other non-Myricom supported packets), they are distributed to the lowest core id or ring 0. No packets get dropped based on the type of packet it is. The only drop cases are bad CRC32 or when there is an overflow (either the adapter and/or application cannot sustain the packet rate).

There is also a user-defined custom hash function capability if you would like to define your own filters. There is an example of how to write a user-defined custom hash function in the **Sniffer10G v3 API Documentation.**

For QinQ support, the customer must write a custom hash function.

Sniffer10G v2 Receive-Side Scaling Hashing

By default, SNF_RSS_FLAGS=(SNF_RSS_IP | SNF_RSS_SRC_PORT | SNF_RSS_DST_PORT).
In **snf.h** there is this definition

```
enum snf_rss_mode_flags {
SNF_RSS_IP = 0x01, /**< Include IP (v4 or v6) SRC/DST addr in hash */
SNF_RSS_SRC_PORT = 0x10, /**< Include TCP/UDP SRC port in hash */
SNF_RSS_DST_PORT = 0x20 /**< Include TCP/UDP DST port in hash */ };
```

Sniffer10G’s RSS hashing options only operate on the outermost headers in the presence of tunneling. If the packet is an IP packet and SNF_RSS_IP is desired, the IPs will be considered in the hash. If the IP packet is UDP or TCP, the the ports are considered in the hashing. If all the traffic is tunneled through only a few distinct UDP/TCP flows, then the hashing will only distribute through those outer UDP headers.

We parse VLAN but then any RSS is applied only if what follows is an IPv4/Ipv6 header that is followed by a TCP/UDP header. This means that any form of TCP/UDP tunneling considers only the outermost headers.

For Ethernet frames bearing “non-hashable” traffic (e.g., MPLS or other non-Myricom supported packets), they are distributed to the lowest core id or ring 0. No packets get dropped based on the type of packet it is. The only drop cases are bad CRC32 or when there is an overflow (either the adapter and/or application cannot sustain the packet rate).

There is also a user-defined custom hash function capability if you would like to define your own filters.
For QinQ support, the customer must write a custom hash function.

Revision	**Date**	**Change**
1	7/22/2016	Initial Draft
2	8/19/2016	Feedback Edits