# Revision History

<table>
<thead>
<tr>
<th>Revision Number</th>
<th>Description</th>
<th>Revision Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Install-guide for vCMTS reference dataplane release v21.10.0</td>
<td>November 2021</td>
</tr>
</tbody>
</table>
Related Information

This install guide relates to v21.10.0 of the Intel vCMTS reference dataplane package which may be downloaded from the Intel 01.org site at the following link: https://01.org/access-network-dataplanes.

Legal Information

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL’S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request. Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order. Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or by visiting Intel’s Web site at www.intel.com.

© 2021 Intel Corporation. All rights reserved. Intel, the Intel logo, and Xeon are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.
# Contents

1  Introduction....................................................................................................................... 7  
1.1  System overview............................................................................................................... 7  
  1.1.1  vCMTS Reference Dataplane Pipeline..................................................................... 9  
  1.1.2  Cloud-native Software Stack ............................................................................... 10  
  1.1.3  CPU Core Management ....................................................................................... 12  
  1.1.4  Power Management Features ............................................................................. 13  
1.2  Network Interface Configuration ................................................................................... 15  
1.3  Memory Module Configuration ....................................................................................... 16  
1.4  System Configuration ................................................................................................... 18  
1.5  Release Package Contents ............................................................................................ 20  
2  Installation Guide ............................................................................................................... 23  
2.1  vCMTS dataplane server preparation............................................................................... 23  
  2.1.1  Configure system BIOS settings .......................................................................... 23  
  2.1.2  Check NIC Firmware and Driver versions ........................................................... 24  
  2.1.3  Check disk space settings ................................................................................... 25  
  2.1.4  Check OS and Linux kernel versions .................................................................... 26  
  2.1.5  Disable automatic Linux package updates ........................................................... 27  
  2.1.6  Change file size limit settings ............................................................................. 27  
  2.1.7  Configure Linux GRUB settings ......................................................................... 28  
  2.1.8  Install public key for remote access to traffic-generator server and Ansible host . 29  
  2.1.9  Load the vCMTS reference dataplane package .................................................... 29  
  2.1.10 Configure proxy servers ...................................................................................... 30  
  2.1.11 Install Linux packages ....................................................................................... 31  
2.2  vCMTS traffic-generator server preparation ................................................................... 32  
  2.2.1  Configure system BIOS settings .......................................................................... 32  
  2.2.2  Check NIC Firmware and Driver versions ........................................................... 33  
  2.2.3  Check disk space settings ................................................................................... 34  
  2.2.4  Check OS and Linux kernel versions .................................................................... 35  
  2.2.5  Disable automatic Linux package updates ........................................................... 36  
  2.2.6  Change file size limit settings ............................................................................. 37  
  2.2.7  Configure Linux GRUB settings ......................................................................... 37  
  2.2.8  Install public key for remote access to dataplane server and Ansible host ........ 38  
  2.2.9  Load the vCMTS reference dataplane package .................................................... 38  
  2.2.10 Configure proxy servers ...................................................................................... 39  
  2.2.11 Install Linux packages ....................................................................................... 40  
  2.2.12 Install the vCMTS platform management tool .................................................... 40  
2.3  vCMTS Ansible Host preparation ................................................................................... 41  
  2.3.1  Install public key for remote access to dataplane and traffic-generator servers.. 41  
  2.3.2  Load the vCMTS reference dataplane package .................................................... 42
Introduction

2.3.3 Configure proxy servers ................................................................. 42
2.3.4 Install Linux packages .................................................................. 43
2.3.5 Install the vCMTS platform management tool .............................. 43
2.4 Configure vCMTS platform settings ............................................... 45
2.4.1 vCMTS dataplane platform configuration .................................... 45
2.4.2 vCMTS traffic-generator platform configuration ....................... 46
2.5 Connect NIC interfaces .................................................................... 47
2.6 Install BMRA Components .............................................................. 48
2.6.1 Download BMRA Components .................................................... 48
2.6.2 Apply vCMTS Reference dataplane settings to BMRA playbooks ... 48
2.6.3 Install Intel® IPSEC MB Library ................................................... 49
2.6.4 Deploy BMRA ansible playbooks ............................................... 49
2.6.5 Verify Kubernetes installation and label nodes .......................... 49
2.7 Install vCMTS dataplane node software ........................................ 51
2.7.1 Generate openssl certificates .................................................... 51
2.7.2 Build vCMTS power manager .................................................... 51
2.7.3 Install Intel pcm-memory tool .................................................... 51
2.7.4 Build vCMTS dataplane application container .......................... 52
2.7.5 Build vCMTS Upstream Scheduler application container ......... 52
2.8 Install vCMTS traffic-generator node software .............................. 53
2.8.1 Build Pktgen application container .......................................... 53
2.9 Configure vCMTS dataplane ........................................................... 54
2.9.1 Configure vCMTS dataplane service-group options .................. 57
2.10 Run vCMTS dataplane and traffic-generator software .................. 59
2.10.1 Start infrastructure and application instances ......................... 59
2.10.2 How to stop infrastructure and application instances ............. 60
2.11 System verification ........................................................................ 61
2.11.1 Start dataplane traffic ............................................................. 61
2.11.2 Check the Grafana dashboard .................................................. 62
2.11.3 vCMTS CLI Tool ..................................................................... 65
2.11.4 Attach to application instances .............................................. 66
2.12 System reboot .............................................................................. 67
3 Appendix ......................................................................................... 68
3.1 Tuning P-states for optimum performance .................................... 68
3.1.1 Install power management utilities tool .................................... 68
3.1.2 Determine current P-state settings .......................................... 68
3.1.3 Configure and Tune P-state settings ........................................ 70
3.1.4 Verify P-state settings .............................................................. 71
3.2 DOCSIS DDP Forwarding Sample Application ............................. 72
3.2.1 Description .............................................................................. 72
3.2.2 User Guide ............................................................................ 73
List of Figures

Figure 1  Intel vCMTS Reference Dataplane - Deployment View ................................................. 8
Figure 2  Overview of vCMTS Dataplane Packet-processing.......................................................... 9
Figure 3  Intel vCMTS Reference Dataplane - Packet-processing Pipeline................................... 10
Figure 4  Intel vCMTS Reference Dataplane – Cloud-native Software Stack ............................. 11
Figure 5  Intel vCMTS Dataplane Reference Platform – Layout for 32 Core Dual Processor ........ 12
Figure 6  Intel vCMTS Dataplane Reference Platform – Power Management Architecture ........... 13
Figure 7  Intel Speed Select Technology - Base Frequency Feature ............................................. 14
Figure 8  Grafana Dashboard Login Screen ................................................................................... 63
Figure 9  Intel vCMTS Dataplane Reference Platform – System Summary Dashboard.................. 64

List of Tables

Table 1  System BIOS settings – vCMTS Dataplane server .......................................................... 23
Table 2  Linux GRUB settings – vCMTS Dataplane server .......................................................... 28
Table 3  System BIOS settings – vCMTS Traffic-generator server .............................................. 32
Table 4  Linux GRUB settings – vCMTS Traffic-generator server ............................................... 37
Table 5  vCMTS dataplane configurable service-group options .................................................... 54
Table 6  vCMTS dataplane fixed service-group options ................................................................. 56
Table 7  Service-group bandwidth guide ......................................................................................... 57

Reference Documents

[1] Intel® Container Bare Metal Reference Architecture Guide,  

[2] Intel® Speed Select Technology Article,  
1 Introduction

This document describes, step by step, how to install and run the Intel® vCMTS Reference Dataplane system with the Intel® Container Bare Metal Reference Architecture (BMRA). This includes a DPDK-Pktgen-based vCMTS traffic generation system for upstream and downstream traffic simulation.

1.1 System overview

The Intel® vCMTS reference dataplane environment consists of a vCMTS dataplane node and a traffic generation node. The reference platform, as presented in this guide, for both of these nodes is a 2U server with Intel® Xeon® 6338N scalable processors and Intel® 810 100G or 200G network interface cards (NICs). The vCMTS dataplane node may also optionally include Intel® QuickAssist Technology (QAT) 8970 100Gbps or 8960 50Gbps PCIe cards.

Note that the system described in this document should serve as a sample system configuration. Servers based on other Intel® Xeon® scalable processors or Xeon® D processors with different core-counts are also supported. Intel® 710 10G quad-port or 25G dual-port NICs may be used instead of 810 NICs, and if sufficient PCI slots are available on the platform, multiple NICs per CPU may be used.

The entire system is deployed under the Intel® Container Bare Metal Reference Architecture (BMRA). This is a Kubernetes-based reference software stack which is designed to enable early adoption of Intel platforms and open-source platform software capabilities for cloud-native network function deployments. This complete, flexible, and scalable solution template allows deployment of Kubernetes clusters that can be based on multiple worker nodes managed by one or more Kubernetes control nodes. All servers are connected with one or two switches that provide connectivity within the cluster and to the cloud. Ansible playbooks are provided for auto-provisioning and auto-configuration of BMRA and can be installed on any connected host server. The Intel® vCMTS reference dataplane has been adapted to be deployed by BMRA ansible playbooks and run on the BMRA cloud-native software stack. See reference [1] for more detail on BMRA.

Under the BMRA deployment, multiple containerd* containers host DPDK-based DOCSIS MAC upstream and downstream dataplane processing for individual cable service-groups (SGs) on the vCMTS dataplane node. On the vCMTS traffic-generation node containerd* containers host DPDK Pktgen-based traffic generation instances which simulate DOCSIS traffic into corresponding vCMTS dataplane instances.

Each vCMTS dataplane instance is deployed as a Pod under Kubernetes*, and represents a cable service-group. DOCSIS control-plane is simulated through a JSON configuration file containing subscriber cable-modem information. DOCSIS upstream Scheduler instances may also be optionally configured to simulate the granting of upstream traffic bandwidth by a vCMTS.

Telemetry functions also run in containerd* containers under Kubernetes. A comprehensive set of vCMTS dataplane statistics and platform KPI’s are gathered by the open-source collectd* daemon and stored in a Prometheus* time-series database. A Grafana* dashboard is provided for visualization of these metrics based on Prometheus* queries.
The sample platform configuration shown in Figure 1 below supports vCMTS dataplane processing for a configurable number of service-groups on a BMRA Kubernetes orchestrated platform. Various mappings of cores and VF’s to service-group pods are supported as described in section 2.9 “Configure vCMTS dataplane”.

**NOTE:** if Upstream Scheduling is enabled, additional pods are deployed for this function. Each Upstream Scheduling Pod allocates upstream bandwidth for multiple vCMTS Pod’s.
1.1.1 vCMTS Reference Dataplane Pipeline

The core component of the Intel® vCMTS reference dataplane release package is a reference implementation of a DOCSIS MAC dataplane, also known as the vCMTS dataplane.

Intel has developed a DOCSIS MAC dataplane compliant with DOCSIS 3.1 specifications (notably MULPI, DEPI, UEPI and SEC specifications) and based on the DPDK packet-processing framework. The key purpose of this development is to provide a tool for characterization of vCMTS dataplane packet-processing performance and power-consumption on Intel® Xeon® platforms.

Below is a high level view of DOCSIS MAC dataplane processing performed by a vCMTS Dataplane network function.

**Figure 2  Overview of vCMTS Dataplane Packet-processing**

The vCMTS upstream and downstream packet processing pipelines implemented by Intel® are shown in Figure 3 below. Note that the downstream dataplane is implemented as a two-stage pipeline of upper-MAC and lower-MAC processing. The DPDK API used for each significant DOCSIS MAC dataplane function is also shown below.
Each vCMTS pod as shown in the Kubernetes orchestrated platform in Figure 1 contains an instantiation of the above upstream and downstream dataplane pipelines. Each vCMTS pod handles all subscriber traffic for a specific cable service-group which covers a group of cable subscribers in a particular geographical area.

1.1.2 Cloud-native Software Stack

The Intel® vCMTS reference dataplane runs on the Intel® Container Bare Metal Reference Architecture (BMRA) cloud-native software stack as shown in Figure 4 below.

vCMTS upstream and downstream dataplane processing for individual cable service-groups run in Containerd* containers on Ubuntu OS, allowing them to be instantiated and scaled independently.

The entire system including applications, telemetry, power-management and infrastructure management is orchestrated by Kubernetes*, with Intel-developed plugins being used for resource management functions such as assignment of SR-IOV interfaces for Intel® NIC’s and QAT devices.

The python-based vCMTS platform management tool, vcmts-pm provides commands to abstract and simplify the configuration of vCMTS components before handing over management to Kubernetes*.
Figure 4  Intel vCMTS Reference Dataplane – Cloud-native Software Stack

Intel Xeon® 5P and D (2nd & 3rd Gen), 700 and 800 series NIC’s, Quick Assist
1.1.3 CPU Core Management

CPU core allocation is managed by the Kubernetes Native CPU Manager, which manages a shared pool of CPU cores from which cores may be allocated for exclusive use by upstream and downstream dataplane containers.

Initially the CPU core pool contains all of the available CPU cores in the compute node.

vCMTS dataplane pods are assigned QoS class of “Guaranteed” so that when containers are created for these pods by the kubelet process, CPU cores are removed from the shared pool and allocated exclusively for the lifetime of the container. Any existing containers using these cores are migrated to another core.

In this way each vCMTS dataplane Pod can request exclusive cores for its upstream and downstream dataplane threads, as well as for Upstream Scheduling if enabled. The native CPU manager provides CPU core isolation which is critical for data-plane performance.

Figure 5 below shows an example of a CPU core layout on a 32-core dual-processor CPU. In this example CPU cores are allocated for upstream and downstream processing per vCMTS Dataplane Pod (each of which represents a Cable service-group) and for Upstream Scheduling. Two sibling hyper-threads of a core for are utilized for downstream dataplane processing per pod while upstream processing utilizes one sibling hyper-thread of a core for upstream dataplane with the other sibling hyper-thread reserved for management processing such as statistics collection. A single hyper-thread sibling handles the upstream scheduling for 2 Cable service-groups.

**Figure 5  Intel vCMTS Dataplane Reference Platform – Layout for 32 Core Dual Processor**

NOTE: Control-Plane processing is not currently supported for the Intel Reference vCMTS platform. CPU resources are reserved for this function not running on the Intel Reference vCMTS platform.
1.1.4 Power Management Features

The Intel® vCMTS reference dataplane offers the capability to make use of the latest Intel® Power Management technology. These features enable both power saving and improved performance through the use of Intel® Speed Select Technology and DPDK Power Management features. Three power management features are supported: Time-of-day power management, Branch-monitoring and Intel® Speed Select Technology Base Frequency. These power management features may be enabled using the `vcmts-pm` tool menu and are described in more detail in the following sections.

1.1.4.1 Time of Day Power Management

By using the time-of-day power management feature, power-profiles may be configured for automated power management of a Kubernetes-orchestrated vCMTS system. A power profile may be configured for each vCMTS dataplane Pod, and these are sent to the Power-Manager at startup, as shown in the architecture diagram below.

These power-profiles determine how dataplane CPU core state is managed by the DPDK Power-Manager. For quiet hours during the day when network traffic is low, dataplane CPU cores associated with a service-group may be set to a power-state with a lower frequency as determined from its power-profile. The quiet and busy hours can be configured using the `vcmts-pm` tool provided in the Intel® vCMTS reference dataplane release package.
1.1.4.2  Branch Monitor Power Management

This power management method uses branch ratio counters to determine CPU core busy-ness. When this method is selected, a list of vCMTS dataplane cores will be monitored.

The algorithm pays close attention to the ratio between branch hits and branch misses. A tightly polling PMD thread will have a very low branch ratio, which indicates that a low rate of packets is being processed and the core frequency may be scaled down to the minimum allowed value.

When the packet receive rate increases, the code path will alter, causing the branch ratio to increase. When the ratio goes above the ratio threshold, the core frequency will be scaled up to the maximum allowed value.

Through this method idle and busy periods can be identified and power management applied to enable savings without negatively impacting dataplane performance.

1.1.4.3  Speed Select Technology Base Frequency (SST-BF)

Intel® Speed Select Technology is an umbrella term for a collection of features that provide more granular control over CPU performance. The Intel® Speed Select Technology Base Frequency (SST-BF) feature allows specific cores to run at a higher base frequency (i.e. P1) by reducing the base frequencies of other cores, as shown below.

![Figure 7  Intel Speed Select Technology - Base Frequency Feature](image)

The number of low/high priority cores and core-frequencies depends on the CPU SKU, e.g. for a 32-core Intel® Xeon® 3rd generation N SKU there may be 18 high-priority cores which could be allocated for downstream traffic processing.

The vCMTS reference system supports the prioritization of certain service-groups to use a higher base frequency than others, thus improving their performance.

This feature is only available on compatible platforms. For more details on Intel® Speed Select Technology, see reference [2].
1.2 Network Interface Configuration

The network interface ports of the vCMTS dataplane and traffic-generation nodes shown in Figure 1 should be interconnected by optical fiber cables via an Ethernet switch.

Traffic should be routed between correlating Pktgen and vCMTS dataplane instances by MAC learning in the switch as MAC addresses of vCMTS dataplane and Pktgen ports are based on service-group ID's. Switch configuration is not covered in this document.

If an Ethernet Switch is not available, the NIC ports of the vCMTS dataplane and traffic-generation nodes may be connected directly by optical fiber cables. However, in this case care must be taken to connect the physical NIC ports of correlating Pktgen and vCMTS dataplane application instances. A helper script is provided in the release package to assist with this and instructions are included in the later sections of this install guide as environment settings must be configured before doing this.

When vCMTS dataplane traffic is started as described in section 2.11.11 an ARP request is sent from each traffic-generator instance to establish a link with its correlating vCMTS dataplane instance.

⚠️ Please note that NICs must be installed in appropriate CPU-affinitized PCI slots for balanced I/O (which for the sample configuration in Figure 1 means two NICs per CPU socket). Note also that all NICs used for vCMTS dataplane traffic must be of the same type.

The NIC layout for the system can be checked by running the command shown below.

Note that for the example below only the 100G E810 NICs are used for vCMTS dataplane traffic. Device ports with their most significant address bit unset below (4b:00.0, 4e:00.0) are affinitized to CPU socket 0, while those with their most significant address bit set (b1:00.0, b4:00.0) are affinitized to CPU socket 1. This system configuration is an example of balanced I/O.

```
lspci | grep Ethernet

17:00.0 Ethernet controller: Intel Corporation Ethernet Controller X710 for 10GBASE-T (rev 02)
17:00.1 Ethernet controller: Intel Corporation Ethernet Controller X710 for 10GBASE-T (rev 02)
4b:00.0 Ethernet controller: Intel Corporation Ethernet Controller E810-C for QSFP (rev 02)
4e:00.0 Ethernet controller: Intel Corporation Ethernet Controller E810-C for QSFP (rev 02)
b1:00.0 Ethernet controller: Intel Corporation Ethernet Controller E810-C for QSFP (rev 02)
b4:00.0 Ethernet controller: Intel Corporation Ethernet Controller E810-C for QSFP (rev 02)
```
1.3 Memory Module Configuration

It is very important to ensure that DRAM modules are installed correctly in the server so that all memory channels are utilized.

For example, for the Intel® Xeon® 6338N scalable processor there are 8 memory channels per CPU socket. In this case a minimum of 16 DRAM modules are required to utilize all memory channels. Furthermore, modules should be installed in the correct color-coded slots for optimum memory channel utilization. In general, it is recommended to use high-speed memory modules if possible.

DRAM module layout on the system can be checked by running the command below.

```bash
lshw -class memory
```

Correct memory-channel utilization can also be verified using the pcm-memory tool which is provided by Intel. Below is an example of a correctly configured system which is using all 16 memory channels.
```
<table>
<thead>
<tr>
<th>Socket 0</th>
<th>Socket 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Channel Monitoring</td>
<td>Memory Channel Monitoring</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mem Ch 0: Reads (MB/s):</th>
<th>Mem Ch 0: Reads (MB/s):</th>
<th>Mem Ch 0: Reads (MB/s):</th>
<th>Mem Ch 0: Reads (MB/s):</th>
</tr>
</thead>
<tbody>
<tr>
<td>21.98</td>
<td>4.29</td>
<td>21.98</td>
<td>4.29</td>
</tr>
<tr>
<td>Writes (MB/s):</td>
<td>Writes (MB/s):</td>
<td>Writes (MB/s):</td>
<td>Writes (MB/s):</td>
</tr>
<tr>
<td>17.34</td>
<td>1.32</td>
<td>17.34</td>
<td>1.32</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------</td>
<td>-------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>21.93</td>
<td>4.30</td>
<td>21.93</td>
<td>4.30</td>
</tr>
<tr>
<td>Writes (MB/s):</td>
<td>Writes (MB/s):</td>
<td>Writes (MB/s):</td>
<td>Writes (MB/s):</td>
</tr>
<tr>
<td>17.25</td>
<td>1.34</td>
<td>17.25</td>
<td>1.34</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------</td>
<td>-------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>22.07</td>
<td>4.31</td>
<td>22.07</td>
<td>4.31</td>
</tr>
<tr>
<td>Writes (MB/s):</td>
<td>Writes (MB/s):</td>
<td>Writes (MB/s):</td>
<td>Writes (MB/s):</td>
</tr>
<tr>
<td>17.52</td>
<td>1.35</td>
<td>17.52</td>
<td>1.35</td>
</tr>
<tr>
<td>~ ~ ~</td>
<td>~ ~ ~</td>
<td>~ ~ ~</td>
<td>~ ~ ~</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Writes (MB/s):</td>
<td>Writes (MB/s):</td>
<td>Writes (MB/s):</td>
<td>Writes (MB/s):</td>
</tr>
<tr>
<td>17.21</td>
<td>1.51</td>
<td>17.21</td>
<td>1.51</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------</td>
<td>-------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>21.83</td>
<td>4.33</td>
<td>21.83</td>
<td>4.33</td>
</tr>
<tr>
<td>Writes (MB/s):</td>
<td>Writes (MB/s):</td>
<td>Writes (MB/s):</td>
<td>Writes (MB/s):</td>
</tr>
<tr>
<td>17.15</td>
<td>1.44</td>
<td>17.15</td>
<td>1.44</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>NODE 0 Mem Read (MB/s):</td>
<td>NODE 0 Mem Read (MB/s):</td>
<td>NODE 0 Mem Read (MB/s):</td>
<td>NODE 0 Mem Read (MB/s):</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------</td>
<td>-------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>175.58</td>
<td>34.55</td>
<td>175.58</td>
<td>34.55</td>
</tr>
<tr>
<td>NODE 0 Mem Write (MB/s):</td>
<td>NODE 0 Mem Write (MB/s):</td>
<td>NODE 0 Mem Write (MB/s):</td>
<td>NODE 0 Mem Write (MB/s):</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------</td>
<td>-------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>138.71</td>
<td>11.11</td>
<td>138.71</td>
<td>11.11</td>
</tr>
<tr>
<td>NODE 0 PMM Read (MB/s):</td>
<td>NODE 0 PMM Read (MB/s):</td>
<td>NODE 0 PMM Read (MB/s):</td>
<td>NODE 0 PMM Read (MB/s):</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------</td>
<td>-------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>NODE 0 PMM Write (MB/s):</td>
<td>NODE 0 PMM Write (MB/s):</td>
<td>NODE 0 PMM Write (MB/s):</td>
<td>NODE 0 PMM Write (MB/s):</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------</td>
<td>-------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>NODE 0 Memory (MB/s):</td>
<td>NODE 0 Memory (MB/s):</td>
<td>NODE 0 Memory (MB/s):</td>
<td>NODE 0 Memory (MB/s):</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------</td>
<td>-------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>314.29</td>
<td>45.66</td>
<td>314.29</td>
<td>45.66</td>
</tr>
</tbody>
</table>

| System DRAM Read Throughput (MB/s): | 210.13 |
| System DRAM Write Throughput (MB/s): | 149.82 |
| System PMM Read Throughput (MB/s): | 0.00 |
| System PMM Write Throughput (MB/s): | 0.00 |
| System Read Throughput (MB/s): | 210.13 |
| System Write Throughput (MB/s): | 149.82 |
| System Memory Throughput (MB/s): | 359.94 |
```

1.4 System Configuration

The following is a sample system configuration for Intel vCMTS reference dataplane v21.10.0.

<table>
<thead>
<tr>
<th>vCMTS Dataplane Node</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hardware</strong></td>
</tr>
<tr>
<td>CPU</td>
</tr>
<tr>
<td>Memory</td>
</tr>
<tr>
<td>Hard Drive</td>
</tr>
</tbody>
</table>
| Network Interface Card        | 2 x Intel® Ethernet Network Adaptor 810 100GbE (x16 PCI)  
or 4 x Intel® Ethernet Network Adapter XXV710-DA2 25GbE  
or 4 x Intel® Ethernet Network Adapter X710-DA4 10GbE  
NOTES: Additional NICs may be used if PCIe slots are available |
| Crypto Acceleration Card      | Intel® QuickAssist Adapter 8970 Card (100Gbps, x16 PCI) or 8960 (50Gbps) |
| **Software**                  |
| Host OS                       | Ubuntu* 20.04, Linux* Kernel v5.4.0+ |
| BMRA                          | 21.09 |
| DPDK                          | DPDK v21.08, intel-ipsec-mb v1.0 |
| vCMTS                         | Intel vCMTS Reference Dataplane v21.10.0 |
| Linux Container               | Containerd 1.4.6 |
| Container Orchestrator        | Kubernetes* v1.21.1 |
| Statistics                    | Collectd* v5.12.0, Prometheus 2.29.2, Grafana 7.2.0 |

<table>
<thead>
<tr>
<th>vCMTS Traffic Generator Node</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hardware</strong></td>
</tr>
<tr>
<td>CPU</td>
</tr>
<tr>
<td>Memory</td>
</tr>
<tr>
<td>Hard Drive</td>
</tr>
</tbody>
</table>
| Network Interface Card        | 2 x Intel® Ethernet Network Adaptor 810 100GbE (x16 PCI)  
or 4 x Intel® Ethernet Network Adapter XXV710-DA2 25GbE  
or 4 x Intel® Ethernet Network Adapter X710-DA4 10GbE  
NOTES: Additional NICs may be used if PCIe slots are available |
| **Software**                  |
| Host OS                       | Ubuntu* 20.04, Linux* Kernel v5.4.0+ |
| BMRA                          | BMRA Ansible Paybook v21.09 |
| DPDK                          | DPDK v20.08 |
## Introduction

<table>
<thead>
<tr>
<th>Linux Container</th>
<th>Containerd 1.4.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Container Orchestrator</td>
<td>Kubernetes* v1.21.1</td>
</tr>
<tr>
<td>Traffic Generator</td>
<td>DPDK Pktgen v19.11.0</td>
</tr>
</tbody>
</table>

### vCMTS Ansible Host

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Any server or PC with a minimum of 8GB RAM and 128GB of disk space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software</td>
<td>Intel® BMRA v21.09 running on Ubuntu* 20.04 with Linux* Kernel v5.4.0+</td>
</tr>
</tbody>
</table>
1.5 Release Package Contents

The directory tree of the Intel vCMTS Reference Dataplane Release package is shown below.
The following is a description of the main directories of the release package, and their contents.

"vcmts" (top-level) directory:
Contains the following release documentation:
- README: release package details
- LICENSE: license file relevant to the package
- RELEASE_NOTES: release note relevant to the package

"bmra" subdirectory:
Contains files related to BMRA and Kubernetes, which are required for the Intel(R) vCMTS reference dataplane performance analysis environment, in the following sub-directories:
- config: contains configuration template files for BMRA
- helm: contains helm files used for configuration of vCMTS dataplane, vCMTS upstream scheduler, Pktgen and Power-Mgmt Kubernetes Pod's
- patches: contains files to patch BMRA with features needed for the vCMTS reference dataplane performance analysis environment

"pktgen" subdirectory:
Contains files to build the traffic simulation components of the vCMTS reference dataplane platform, in the following sub-directories:
- config: contains PKT files with commands performed at Pktgen start-up such as setting of src/dest IP addresses and ARP handling
- container: contains files to build the container image for the vCMTS traffic generator (based on DPDK Pktgen application)

"power-mgr" subdirectory:
Contains files related to the vCMTS Power Manager, in the following sub-directory:
- container: contains a file to build the docker image for the vCMTS power manager (based on the DPDK Power manager)

"src" sub-directory:
Contains top-level Makefile to build the vCMTS dataplane application, and directory structure for source-code modules.

The "src" directory contains the following sub-directories:
- app: contains C source-code modules, container image file and sample configuration files for the vCMTS dataplane application, us-sched application and docsis-ddp-forward sample application
lib: contains C source-code modules for libraries used by the vCMTS dataplane application
mk: contains makefiles to build the vCMTS dataplane application
patches: contains files to patch DPDK and Pktgen with changes needed for the vCMTS reference dataplane performance analysis environment

"telemetry" sub-directory:
Contains files used to configure the telemetry components of the vCMTS dataplane application environment, in the following sub-directories:
collectd: contains files for the collectd service for vCMTS dataplane telemetry
prometheus: contains files for the prometheus service for vCMTS dataplane telemetry
grafana: contains files used to deploy vCMTS dataplane telemetry dashboards to the grafana service

"tools" sub-directory:
Contains tools which may be used to configure and manage the vCMTS dataplane reference system, in the following sub-directories:

cvmts-env: contains shell scripts to configure the vCMTS dataplane and traffic generation environments and deploy them on BMRA

cvmts-pm: contains python source modules for a platform management tool which performs the following functions:

- BMRA configuration based on platform hardware detection
- configuration of vCMTS dataplane application instances
- start/stop/re-start vCMTS dataplane application instances
- start/stop Pktgen application instances
- start/stop/rate control of Pktgen-generated traffic
- vCMTS dataplane throughput and latency measurement

vcmts-cli: contains python source modules for a command-line tool to display status and statistics for vCMTS dataplane application instances

"traffic-profiles" subdirectory:
Contains a zip package with control-plane configuration files for a number of vCMTS service-group scenarios and correlating PCAP traffic files for upstream and downstream traffic simulation. A zip-file is included in the release package with pcap's based on an iMix packet-size distribution. A separate zip-file is provided containing pcap's specific for the docsis-ddp-forward sample application
2 Installation Guide

2.1 vCMTS dataplane server preparation
A number of steps are required to prepare the vCMTS dataplane server for software installation.

⚠ Please note that you must log in as root user to the vCMTS dataplane server to perform these steps.

2.1.1 Configure system BIOS settings
The following System BIOS settings are required on the vCMTS dataplane server.

<table>
<thead>
<tr>
<th>BIOS Setup Menu</th>
<th>BIOS Option</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced-&gt;Processor Configuration</td>
<td>Intel(R) Hyper-Threading Tech</td>
<td>Enabled</td>
</tr>
<tr>
<td>Advanced-&gt;Integrated IO Configuration</td>
<td>Intel(R) VT for Directed I/O</td>
<td>Enabled</td>
</tr>
<tr>
<td>Advanced-&gt;Power &amp; Performance</td>
<td>CPU Power and Performance profile</td>
<td>Balanced Performance</td>
</tr>
<tr>
<td>Advanced-&gt;Power &amp; Performance-&gt;CPU P State Control</td>
<td>Intel(R) Turbo Boost Technology</td>
<td>Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOTE: enable if SST-BF required</td>
</tr>
<tr>
<td>Advanced-&gt;Power &amp; Performance-&gt;CPU P State Control</td>
<td>Energy Efficient Turbo</td>
<td>Disabled</td>
</tr>
<tr>
<td>Advanced-&gt;Power &amp; Performance-&gt;CPU P State Control</td>
<td>Enhanced Intel Speed Select Technology</td>
<td>Enabled</td>
</tr>
<tr>
<td>Advanced-&gt;Power &amp; Performance-&gt;CPU C State Control</td>
<td>C1E</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

⚠ Please note: Bi-furcation configuration in BIOS is required when using Intel 200G (E810-2CQDA2) NIC's. Please contact Intel for details.
2.1.2 Check NIC Firmware and Driver versions

2.1.2.1 Intel 710 25G or 10G NIC's
If using Intel 710 25G or 10G NIC's, check the i40e base driver and firmware versions by running `ethtool` as per the example below for a dataplane network interface.

E.g. for network interface `ens785f1`

```
ethtool -i ens785f1
driver: i40e
version: 2.10.19.30
firmware-version: 8.10 0x80009424 1.1747.0
expansion-rom-version:
bus-info: 0000:18:00.1
supports-statistics: yes
supports-test: yes
supports-eeprom-access: yes
supports-register-dump: yes
supports-priv-flags: yes
```

The driver version should be 2.10.19 or later.
The firmware version should be 8.10 or later.

If need to update, the required 710 NIC driver and firmware can be downloaded at the links below.

710 NIC Driver

710 NIC Firmware

2.1.2.2 Intel 810 100G NIC's
If using Intel 810 100G NIC's, check the ICE base driver and firmware versions by running `ethtool` as per the example below for a dataplane network interface.

E.g. for network interface `ens801f0`
The latest driver version available at the time of this vCMTS reference dataplane release is: 1.6.7
The compatible firmware version for this driver version is: 3.00

If need to update, the required 810 NIC driver and firmware can be downloaded at the links below.

810 NIC Driver

810 NIC Firmware

2.1.3 Check disk space settings
At least 400GB of disk-space should be available on the vCMTS dataplane server for the installation of the vCMTS reference dataplane software and infrastructure components.

See typical disk info below for a fully installed vCMTS dataplane server.

```
df -h
Filesystem       Size  Used  Avail %Mounted on
udev             11G   0   11G  0% /dev
tmpfs            9.4G  875M   8.5G  10% /run
/dev/sdb1        440G  188G  229G  46% /
tmpfs            47G   0   47G  0% /dev/shm
tmpfs            5.0M   0   5.0M  0% /run/lock
tmpfs            47G   0   47G  0% /sys/fs/cgroup
```

It is recommended to disable swap space as described below.
First check if swap space is enabled, by running the following command.

```
blkid
```

```
/dev/sda1: UUID="08070332-90b8-45a5-891d-e76167ee876d" TYPE="ext4" PARTUUID="ee39ea61-01"
/dev/sda5: UUID="34c0a658-225b-4b28-abcb-8ac33c30b819" TYPE="swap" PARTUUID="ee39ea61-05"
```

If there is an entry such as TYPE="swap" in the above, this needs to be disabled by running the following command.

```
swapoff -a
```

Furthermore, swap space should be permanently disabled by commenting out any swap entry in the /etc/fstab file.

```
sed -i.bk '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab
```

On reboot, the disabling of swap space can be verified by running the following command, which should display the output below.

```
free -h
```

```
Mem:   62G    57G    239M    2.4M    4.7G    4.4G
Swap:    0B     0B     0B
```

### 2.1.4 Check OS and Linux kernel versions

The recommended OS distribution is Ubuntu 20.04 LTS, and recommended Linux kernel version is 5.4.x or later.

Run the following command to check the OS distribution version

```
lsb_release -a
```

The following output is expected.

```
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 20.04.1 LTS
Release:        20.04
Codename:       bionic
```

Run the following command to check the kernel version of the system:
The following type of output is expected:

```
uname -r
```

```
5.4.0-47-generic
```

If a kernel version older than 5.4.x is displayed, the required kernel can be installed by running the following command:

```
apt-get update
```

### 2.1.5 Disable automatic Linux package updates

First, create the automatic upgrades system file by running the following command.

```
dpkg-reconfigure --priority-low unattended-upgrades
```

Select "Yes" to create the file.

Next, disable automatic package upgrades by editing the system file `/etc/apt/apt.conf.d/20auto-upgrades`, which was created above, as follows:

```
APT::Periodic::Update-Package-Lists "0";
APT::Periodic::Unattended-Upgrade "0";
```

Automatic package updates must also be disabled by editing the system file, `/etc/apt/apt.conf.d/10periodic`, as follows:

```
APT::Periodic::Update-Package-Lists "0";
```

### 2.1.6 Change file size limit settings

Add the following lines to the system file `/etc/security/limits.conf`

```
root soft    fsize unlimited
root hard    fsize unlimited
root soft    nofile unlimited
root hard    nofile unlimited
```
Changes will be applied upon next login to a bash shell.

This change is required as it has been known to cause deployment issues on some systems.

## 2.1.7 Configure Linux GRUB settings

The following Linux GRUB settings are required on the vCMTS dataplane server.

### Table 2  Linux GRUB settings – vCMTS Dataplane server

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
</table>
| isolcpus=2-31,34-63,66-95,98-127 | Isolate vCMTS dataplane cores from the Linux kernel task scheduler  
**NOTE:** based on Dual 32-core CPU |
| nr_cpus=128 | Total number of logical cores on the system (aka hyper-threads) |

The sample isolcpus and nr_cpus settings shown here are for a 32-core dual-processor CPU package. If not using a 32-core dual-processor CPU, the nr_cpus and isolcpus settings need to be adapted to the core-layout of the CPU package being used, as shown in the instructions that follow below. Note also that for the isolcpus setting, all except the first 2 cores on each CPU in the package must be isolated from the Linux kernel task scheduler.

Based on the above table for a 32-core dual-processor CPU package, the Linux kernel GRUB file `/etc/default/grub` should be edited as follows to set the appropriate `GRUB_CMDLINE_LINUX` options.

```bash
GRUB_CMDLINE_LINUX="isolcpus=2-31,34-63,66-95,98-127 nr_cpus=128"
```

⚠️ It is very important to verify the core layout of the CPU package being used on the target system and to adapt `isolcpus` and `nr_cpus` GRUB settings based on this.

The core layout of the CPU package can be verified by running the `lscpu` command as shown below.

The output shown here is for a 32-core dual-processor CPU package.

```
lscpu | grep "CPU(s):"
CPU(s): 128
NUMA node0 CPU(s): 0-31,64-95
NUMA node1 CPU(s): 32-63,96-127
```

Once editing of the Linux kernel GRUB file is complete, run the commands below to compile the GRUB configuration and reboot the server for updated settings to take effect:
2.1.8 Install public key for remote access to traffic-generator server and Ansible host

A public key must be installed to allow remote access from vCMTS dataplane server to the traffic generator server and ansible host. Replace the entry below with the actual vCMTS traffic-generator server hostname and ansible hostname.

```
ssh-keygen -b 4096 -t rsa
go where id_rsa.pub root@trafficgen-hostname
ssh-copy-id -i ~/.ssh/id_rsa.pub root@ansible-hostname
ssh-copy-id -i ~/.ssh/id_rsa.pub root@localhost
```

Select default options at each prompt.

2.1.9 Load the vCMTS reference dataplane package

Next, copy the vCMTS reference dataplane release package to the vCMTS dataplane server and extract into the root directory of the installation, which is assumed to be /opt below.

```
cp intel-vcmtsd-v21-10-0.tar.gz /opt
cd /opt
tar -zxvf intel-vcmtsd-v21-10-0.tar.gz
cd vcmts
ls -lR
```

All files from the vCMTS reference dataplane package files as described in section 1.5, "Release Package Contents" should have been extracted to the /opt/vcmts directory.

Set the MYHOME and VCMTSD_HOST environment variables as follows (assuming the vCMTS reference dataplane package will be installed in the /opt directory)

```
export VCMTSD_HOST=y
export MYHOME=/opt
```
These should also be added as environment settings to the root bashrc file ~/.bashrc (again assuming that the vCMTS reference dataplane release package has been installed into /opt).

```
export VCMSTD_HOST=y
export MYHOME=/opt
```

### 2.1.10 Configure proxy servers

⚠ Please note that proxy configuration is different for each installation environment, so care should be taken to understand the required proxy settings for a particular environment as misconfiguration may greatly disrupt this installation.

The settings below need to be configured if the vCMTS dataplane server is behind a proxy.

Firstly, configure HTTP proxy servers and the no_proxy setting in the ~/.bashrc file.
The example entries below should be replaced with actual proxy address and port number and the actual hostname of the vCMTS dataplane server.

```
export http_proxy=http://myproxy.example.com:8080
export https_proxy=http://myproxy.example.com:8080
export no_proxy=localhost,127.0.0.1,vcmstd-hostname,trafficgen-hostname,ansible-hostname
```

For Linux package installations, HTTP proxy server entries must also be added to the /etc/apt/apt.conf file.

```
Acquire::http::Proxy "http://myproxy.example.com:8080/";
Acquire::https::Proxy "http://myproxy.example.com:8080/";
```

You may need to restart your bash shell to source the settings added to ~/.bashrc file.

```
exec bash
```

### 2.1.11 Install Linux packages

A number of Linux packages must be installed on the vCMTS dataplane server which are required for the vCMTS reference dataplane runtime environment.

Run the environment function as shown below to install Linux packages required for the vCMTS dataplane server.
A power management utilities script is also required on the vCMTS dataplane server. Run the environment function as shown to download and install it.

```
source $MYHOME/vcmts/tools/vcmts-env/env.sh
install_base_ubuntu_pkgs
install_power_mgmt_utilities
```

QAT drivers also need to be installed to the latest. Do this by running the following command on the vCMTS dataplane server.

```
install_qat_drivers
```
2.2 vCMTS traffic-generator server preparation

A number of steps are required to prepare the vCMTS traffic-generator server for software installation.

⚠️ Please note that you must log in as root user to the vCMTS traffic-generator server to perform these steps.

2.2.1 Configure system BIOS settings

The following System BIOS settings are required on the vCMTS traffic-generator server.

<table>
<thead>
<tr>
<th>BIOS Setup Menu</th>
<th>BIOS Option</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced-&gt;Processor Configuration</td>
<td>Intel(R) Hyper-Threading Tech</td>
<td>Enabled</td>
</tr>
<tr>
<td>Advanced-&gt;Integrated IO Configuration</td>
<td>Intel(R) VT for Directed I/O</td>
<td>Enabled</td>
</tr>
<tr>
<td>Advanced-&gt;Power &amp; Performance</td>
<td>CPU Power and Performance profile</td>
<td>Balanced Performance</td>
</tr>
<tr>
<td>Advanced-&gt;Power &amp; Performance-&gt;CPU P State Control</td>
<td>Intel(R) Turbo Boost Technology</td>
<td>Disabled</td>
</tr>
<tr>
<td>Advanced-&gt;Power &amp; Performance-&gt;CPU P State Control</td>
<td>Energy Efficient Turbo</td>
<td>Disabled</td>
</tr>
<tr>
<td>Advanced-&gt;Power &amp; Performance-&gt;CPU P State Control</td>
<td>Enhanced Intel Speed Select Technology</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

⚠️ Please note: Bifurcation configuration in BIOS is required when using Intel 200G (E810-2CQDA2) NIC’s. Please contact Intel for details.
2.2.2 Check NIC Firmware and Driver versions

2.2.2.1 Intel 710 25G or 10G NIC's

If using Intel 710 25G or 10G NIC's, check the i40e base driver and firmware versions by running `ethtool` as per the example below for a dataplane network interface.

E.g. for network interface `ens785f1`

```
ethtool -i ens785f1
      driver: i40e
      version: 2.10.19.30
      firmware-version: 8.10 0x80009424 1.1747.0
      expansion-rom-version:
      bus-info: 0000:18:00.1
      supports-statistics: yes
      supports-test: yes
      supports-register-dump: yes
      supports-priv-flags: yes
```

The driver version should be 2.10.19 or later.
The firmware version should be 8.10 or later.

If need to update, the required 710 NIC driver and firmware can be downloaded at the links below.

**710 NIC Driver**

**710 NIC Firmware**

2.2.2.2 Intel 810 100G NIC's

If using Intel 810 100G or 200G NIC's, check the ICE base driver and firmware versions by running `ethtool` as per the example below for a dataplane network interface.

E.g. for network interface `ens801f0`
The latest driver version available at the time of this vCMTS reference dataplane release is: 1.5.8
The compatible firmware version for this driver version is: 3.00

If need to update, the required 810 NIC driver and firmware can be downloaded at the links below.

810 NIC Driver  

810 NIC Firmware  

2.2.3 Check disk space settings

At least 400GB of disk-space should be available on the vCMTS traffic-generator server for the installation of the vCMTS reference dataplane software and related components.

See typical disk info below for a fully installed vCMTS traffic-generator server.

```
df -h
Filesystem           Size  Used  Avail  Use% Mounted on
udev                  12G   0   12G   0% /dev
tmpfs                 9.5G  1.1G  8.5G  11% /run
/dev/sdb1             330G  150G  164G  48% /
tmpfs                  48G   0   48G   0% /dev/shm
tmpfs                  5.0M   0   5.0M   0% /run/lock
tmpfs                  48G   0   48G   0% /sys/fs/cgroup
```
It is recommended to disable swap space as described below.

First check if swap space enabled, by running the `blkid` command.

*e.g.*

```
blkid
/dev/sda1: UUID="08070332-90b8-45a5-891d-e76167ee876d" TYPE="ext4" PARTUUID="ee39ea61-01"
/dev/sda5: UUID="34c0a658-225b-4b28-abcb-8ac33c30b819" TYPE="swap" PARTUUID="ee39ea61-05"
```

If there is an entry such as `TYPE="swap"` in the above, this needs to be disabled by running the following command.

```
swapoff -a
```

Furthermore, swap space should be permanently disabled by commenting out any swap entry in the `/etc/fstab` file.

```
sed -i.bk '/ swap / s/^\(.*\)$/#1/g' /etc/fstab
```

On reboot, the disabling of swap space can be verified by running the following command, which should display the output below.

```
free -h
```

<table>
<thead>
<tr>
<th></th>
<th>total</th>
<th>used</th>
<th>free</th>
<th>shared</th>
<th>buff/cache</th>
<th>available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mem:</td>
<td>62G</td>
<td>57G</td>
<td>239M</td>
<td>2.4M</td>
<td>4.7G</td>
<td>4.4G</td>
</tr>
<tr>
<td>Swap:</td>
<td>0B</td>
<td>0B</td>
<td>0B</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2.2.4 Check OS and Linux kernel versions

The recommended Linux kernel version is 5.4.x or later, and recommended OS distribution is Ubuntu 20.04 LTS.

Run the following command to check the OS distribution version

```
lsb_release -a
```

No LSB modules are available.
Distributor ID: Ubuntu
Description: Ubuntu 20.04.1 LTS
Release: 20.04
Codename: bionic
Run the following command to check the kernel version of the system:

```bash
uname -r
```

The following type of output is expected:

```
5.4.0-47-generic
```

If a kernel version older than 5.4.x is displayed, the required kernel can be installed by running the following:

```bash
apt-get update
```

### 2.2.5 Disable automatic Linux package updates

First, create the automatic upgrades system file by running the following command.

```bash
dpkg-reconfigure --priority-low unattended-upgrades
```

Select "Yes" to create the file.

Next, disable automatic package upgrades by editing the system file `/etc/apt/apt.conf.d/20auto-upgrades`, which was created above, as follows:

```text
APT::Periodic::Update-Package-Lists "0";
APT::Periodic::Unattended-Upgrade "0";
```

Automatic package updates must also be disabled by editing the system file, `/etc/apt/apt.conf.d/10periodic`, as follows:

```text
APT::Periodic::Update-Package-Lists "0";
```
2.2.6 Change file size limit settings

Add the following lines to the system file /etc/security/limits.conf

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>soft</td>
<td>fsize</td>
<td>unlimited</td>
</tr>
<tr>
<td>root</td>
<td>hard</td>
<td>fsize</td>
<td>unlimited</td>
</tr>
<tr>
<td>root</td>
<td>soft</td>
<td>nofile</td>
<td>unlimited</td>
</tr>
<tr>
<td>root</td>
<td>hard</td>
<td>nofile</td>
<td>unlimited</td>
</tr>
</tbody>
</table>

Changes will be applied upon next login to a bash shell.

This change is required as it has been known to cause deployment issues on some systems.

2.2.7 Configure Linux GRUB settings

The following Linux GRUB settings are required on the vCMTS traffic-generator server.

Table 4 Linux GRUB settings – vCMTS Traffic-generator server

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>isolcpus=2-31,34,63,66-95,98-127</td>
<td>Isolate vCMTS dataplane cores from the Linux kernel task scheduler</td>
</tr>
<tr>
<td>nr_cpus=128</td>
<td>Total number of logical cores on the system (aka hyper-threads)</td>
</tr>
</tbody>
</table>

The sample isolcpus and nr_cpus settings shown here are for a 32-core dual-processor CPU package. If not using a 32-core dual-processor CPU, the nr_cpus and isolcpus settings need to be adapted to the core-layout of the CPU package being used, as shown in the instructions that follow below. Note also that for the isolcpus setting, all except the first 2 cores on each CPU in the package must be isolated from the Linux kernel task scheduler.

Based on the above table for a 32-core dual-processor CPU package, the Linux kernel GRUB file /etc/default/grub should be edited as follows to set the appropriate GRUB_CMDLINE_LINUX options.

```
GRUB_CMDLINE_LINUX="isolcpus=2-31,34,63,66-95,98-127 nr_cpus=128"
```

⚠️ It is very important to verify the core layout of the CPU package being used on the target system and to adapt isolcpus and nr_cpus GRUB settings based on this.

The core layout of the CPU package can be verified by running the lscpu command as shown below.

The output shown here is for a 32-core dual-processor CPU package.
Once editing of the Linux kernel GRUB file is complete, run the commands below to compile the GRUB configuration and reboot the server for updated settings to take effect:

```
update-grub2
reboot
```

### 2.2.8 Install public key for remote access to dataplane server and Ansible host

A public key must be installed to allow remote access from the vCMTS traffic-generator server to the dataplane server and Ansible host. Replace the entry below with the actual vCMTS dataplane server hostname and ansible hostname.

```
ssh-keygen -b 4096 -t rsa
ssh-copy-id -i ~/.ssh/id_rsa.pub root@vcmtd-hostname
ssh-copy-id -i ~/.ssh/id_rsa.pub root@ansible-hostname
ssh-copy-id -i ~/.ssh/id_rsa.pub root@localhost
```

Select default options at each prompt.

### 2.2.9 Load the vCMTS reference dataplane package

Next, copy the vCMTS reference dataplane release package to the vCMTS traffic-generator server and extract into the root directory of the installation, which is assumed to be /opt below.

```
cp intel-vcmtdsv21-10-0.tar.gz /opt
cd /opt
tar -zxvf intel-vcmtdsv21-10-0.tar.gz
cd vcmts
ls -1R
```
All files from the vCMTS reference dataplane package files as described in section 1.5, "Release Package Contents" should have been extracted to the /opt/vcms directory.

Note that the vCMTS reference dataplane release package contains iMix style traffic-profiles as well as a range of fixed packet sizes for RFC 2544 style benchmarking.

Set the MYHOME and PKTGEN_HOST environment variables as follows (assuming the vCMTS reference dataplane package is installed in the /opt directory)

```
export PKTGEN_HOST=y
export MYHOME=/opt
```

These must also be added as environment settings to the root bashrc file ~/.bashrc (again assuming that the vCMTS reference dataplane release package has been installed into /opt ).

```
export PKTGEN_HOST=y
export MYHOME=/opt
```

### 2.2.10 Configure proxy servers

⚠️ Please note that proxy servers are different for every installation environment, so care should be taken to understand the required proxy settings for a particular environment as misconfiguration may greatly disrupt this installation.

The settings below need to be configured if the vCMTS traffic generator server is behind a proxy.

Firstly, configure HTTP proxy servers and the no_proxy setting in the ~/.bashrc file.

The example entries below should be replaced with actual proxy address and port number and the actual hostname of the vCMTS traffic generator server.

```
export http_proxy=http://myproxy.example.com:8080
export https_proxy=http://myproxy.example.com:8080
export no_proxy=localhost,127.0.0.1,trafficgen-hostname,vcmtsd-hostname,ansible-hostname
```

For Linux package installations, HTTP proxy server entries must also be added to the /etc/apt/apt.conf file.
2.2.11 Install Linux packages

A number of Linux packages must be installed on the vCMTS traffic-generator server which are required for the vCMTS reference dataplane runtime environment.

Run the environment function as shown below to install Linux packages required for the vCMTS traffic-generator server (assuming installation in /opt).

```bash
source $MYHOME/vcmts/tools/vcmts-env/env.sh
install_base_ubuntu_pkgs
```

2.2.12 Install the vCMTS platform management tool

The `vcmts-pm` tool is provided with the vCMTS reference dataplane release package. The `vcmts-pm` tool simplifies the configuration, running and management of vCMTS applications on the reference platform. It provides the following functionality:

- configure vCMTS dataplane platform environment settings
- configure vCMTS traffic-generator platform environment settings
- configure BMRA settings based on platform hardware detection
- configure vCMTS dataplane application instances
- start/stop vCMTS infrastructure components
- start/stop vCMTS and traffic-generator instances
- control traffic simulation
- measure vCMTS dataplane throughput and latency capability
- query vCMTS platform and service-group configurations

The `vcmts-pm` tool should only be run from the vCMTS traffic-generator server when starting/stopping vCMTS and traffic-generator instances or when controlling traffic. All configuration commands should be run from the Ansible Host.
2.3 vCMTS Ansible Host preparation

A number of steps are required to prepare the vCMTS Ansible Host for software installation.

⚠️ Please note that you must log in as root user to the vCMTS Ansible Host to perform these steps.

Note also that since the Ansible host is mostly an ancillary device the system requirements and OS/BIOS configuration is not prescribed in detail in this document. Any platform with a minimum of 8GB RAM, 64GB of diskspace and a Ubuntu OS should suffice and specific BIOS settings should not be important.

2.3.1 Install public key for remote access to dataplane and traffic-generator servers

A public key must be installed to allow remote access from the vCMTS Ansible host to the dataplane server and traffic-generator server. Replace the entry below with the actual vCMTS dataplane server hostname and vCMTS traffic-generator server hostname.

```
ssh-keygen -b 4096 -t rsa
ssh-copy-id -i ~/.ssh/id_rsa.pub root@vcmtd-hostname
ssh-copy-id -i ~/.ssh/id_rsa.pub root@traffic-generator-hostname
ssh-copy-id -i ~/.ssh/id_rsa.pub root@localhost
```

Select default options at each prompt.
2.3.2 Load the vCMTS reference dataplane package

Next, copy the vCMTS reference dataplane release package to the vCMTS Ansible host and extract into the root directory of the installation, which is assumed to be /opt below.

Note that the same package is used for vCMTS ansible host installation as was used for the vCMTS dataplane and traffic-generator servers.

```bash
cp intel-vcmtsd-v21-10-0.tar.gz /opt
cd /opt
tar -zxvf intel-vcmtsd-v21-10-0.tar.gz
cd vcmts
ls -lR
```

All files from the vCMTS reference dataplane package files as described in section 1.5, "Release Package Contents" should have been extracted to the /opt/vcmts directory.

Set the MYHOME and ANSIBLE_HOST environment variables as follows (assuming the vCMTS reference dataplane package is installed in the /opt directory)

```bash
export ANSIBLE_HOST=y
export MYHOME=/opt
```

These must also be added as environment settings to the root bashrc file ~/.bashrc (again assuming that the vCMTS reference dataplane release package has been installed into /opt).

```bash
export ANSIBLE_HOST=y
export MYHOME=/opt
```

2.3.3 Configure proxy servers

⚠️ Please note that proxy servers are different for every installation environment, so care should be taken to understand the required proxy settings for a particular environment as misconfiguration may greatly disrupt this installation.

The settings below need to be configured if the vCMTS ansible host is behind a proxy.

Firstly, configure HTTP proxy servers and the no_proxy setting in the ~/.bashrc file. The example entries below should be replaced with actual proxy address and port number and the actual hostname of the vCMTS ansible host.
For Linux package installations, HTTP proxy server entries must also be added to the /etc/apt/apt.conf file.

```bash
export http_proxy=http://myproxy.example.com:8080
export https_proxy=http://myproxy.example.com:8080
export no_proxy=localhost,127.0.0.1,trafficgen-hostname,vcmtsd-hostname,ansible-hostname
```

You may need to restart your bash shell to source the settings added to ~/.bashrc file.

```bash
exec bash
```

### 2.3.4 Install Linux packages

A number of Linux packages must be installed on the vCMTS ansible host which are required for the vCMTS reference dataplane runtime environment.

Run the environment function as shown below to install Linux packages required for the vCMTS ansible host (assuming installation in /opt).

```bash
source $MYHOME/vcmts/tools/vcmts-env/env.sh
install_base_ubuntu_pkgs
```

### 2.3.5 Install the vCMTS platform management tool

The vcmts-pm tool is provided with the vCMTS reference dataplane release package. The vcmts-pm tool simplifies the configuration, running and management of vCMTS applications on the reference platform. It provides the following functionality:

- configure vCMTS dataplane platform environment settings
- configure vCMTS traffic-generator platform environment settings
- configure BMRA settings based on platform hardware detection
- configure vCMTS dataplane application instances
- start/stop vCMTS infrastructure components
- start/stop vCMTS and traffic-generator instances
- control traffic simulation
- measure vCMTS dataplane throughput and latency capability
- query vCMTS platform and service-group configurations
The \texttt{vcmts-pm} tool should mainly be run from the Ansible Host.

\begin{verbatim}
cd \$VCMTS_ROOT/tools/vcmts-pm
sudo apt-get install python3-pip
pip3 install virtualenv
python3 -m virtualenv env
source env/bin/activate
pip3 install -e .
deactivate
\end{verbatim}

The \texttt{vcmts-pm} tool is now installed.
2.4 Configure vCMTS platform settings

Platform configurations must be generated for the vCMTS dataplane and traffic-generator servers in order for the Intel vCMTS reference dataplane runtime environment to operate correctly.

⚠️ Please note that both platform configurations must be generated from the vCMTS ansible host and you must log in as root user to perform the steps described below.

Follow the instructions in the next two sections to configure the runtime environments for vCMTS dataplane and traffic-generator platforms, using the vcmts-pm tool provided in the vCMTS reference dataplane release package.

2.4.1 vCMTS dataplane platform configuration

Firstly, the vCMTS dataplane platform environment configuration is performed.

While the tool is run on the vCMTS ansible host, the actual platform details are gathered remotely from the vCMTS dataplane platform.

The following platform details need to be gathered for the vCMTS dataplane server runtime environment.

- Hostname of the vCMTS Dataplane server
- CPU information - number of CPU's and cores per CPU
- NIC information - NIC physical functions per NUMA node and their PCI addresses
- QAT information - QAT physical functions per NUMA node and their PCI addresses

Run the vcmts-pm tool provided in the vCMTS reference dataplane release package as follows to specify the settings for the vCMTS dataplane platform configuration.

```bash
  cd $MYHOME/vcmts/tools/vcmts-pm
  source env/bin/activate
  vcmts-pm config-platform vcmtsd
  deactivate
```

Follow the menu and prompted instructions to generate the vCMTS dataplane platform configuration.
2.4.2 vCMTS traffic-generator platform configuration

Next the vCMTS traffic-generator platform environment configuration is performed.

The following platform details need to be gathered for the vCMTS traffic-generator (also known as Pktgen) server runtime environment.

- Hostname of the vCMTS traffic-generation server
- CPU information - number of CPU's, cores per CPU (isolated and not isolated) and Pktgen application instance mappings
- NIC information - NIC physical functions per NUMA node and their PCI addresses

Run the vcmts-pm tool provided in the vCMTS reference dataplane release package as follows to specify the settings for the vCMTS traffic-generator (pktgen) platform configuration.

```
cd $MYHOME/vcmts/tools/vcmts-pm
source env/bin/activate
vcmts-pm config-platform pktgen
deactivate
```

Follow the menu and prompted instructions to generate the vCMTS traffic-generator (Pktgen) platform configuration.
2.5 Connect NIC Interfaces

If a switch is not used in the setup, the network interfaces of the vCMTS application and vCMTS traffic-generator servers may need to be directly connected to allow simulation traffic to flow.

In this case, the corresponding links must be connected correctly to one another.

The following steps ensure that the correct connections are made. This section can optionally be ignored if a switch is configured between vCMTS dataplane and traffic-generator servers.

1. Run the following commands on the vCMTS dataplane server then go to step 2.

   ```
source $MYHOME/vcmts/tools/vcmts-env/env.sh
cable_pf_helper
   ```

2. Run the following commands on the vCMTS traffic-generator server then go to step 3.

   ```
source $MYHOME/vcmts/tools/vcmts-env/env.sh
cable_pf_helper
   ```

3. There will be console output from both vCMTS dataplane and traffic-generator servers at this point. A NIC interface on both the vCMTS dataplane and traffic-generator server will also be flashing. Using optical fiber cables, connect the flashing interface on the vCMTS dataplane server to the flashing interface on the vCMTS traffic-generator server.

4. Press Enter key on the console on the vCMTS dataplane server once.

   ```
   This function helps to correctly cable vCMTS and PKTGEN nodes
   Run this function in parallel on both vCMTS and PKTGEN nodes
   Wire the flashing ports on each node together
   Then press enter on both systems and continue
   press enter once link 1 is cabled:
   ```

5. Press Enter key on the console on the vCMTS traffic-generator server once.

   ```
   This function helps to correctly cable vCMTS and PKTGEN nodes
   Run this function in parallel on both vCMTS and PKTGEN nodes
   Wire the flashing ports on each node together
   Then press enter on both systems and continue
   press enter once link 1 is cabled:
   ```

6. Go back to step 3 until all interfaces have been connected and the script exits.
2.6 Install BMRA Components

vCMTS dataplane and pktgen (traffic-generator) application instances are orchestrated using Kubernetes*. Kubernetes* is installed by using the Intel Container Bare Metal Reference Architecture (BMRA) that is described in the System Overview Section. BMRA simplifies and automates the installation of the Kubernetes* cluster using ansible playbooks.

The following sections cover the steps for BMRA installation on the vCMTS reference dataplane system. Note that the installation is further simplified through environment functions and scripts which are provided in the vCMTS reference dataplane package.

⚠️ Please note that you must log in as root user to the respective vCMTS traffic-generator, dataplane and ansible nodes to perform these steps.

2.6.1 Download BMRA Components

To clone the BMRA git run the following commands on the vCMTS ansible host.

```
cd $MYHOME/vcmts
export ANSIBLE_HOST=y
source tools/vcmts-env/env.sh
build_bmra_k8s
```

2.6.2 Apply vCMTS Reference dataplane settings to BMRA playbooks

Some specific settings need to be applied to the BMRA configuration before running the ansible playbooks. These settings are specific to the vCMTS dataplane and traffic-generator platforms and were learned by the pm-tool during the previous steps.

Apply the settings to the BMRA playbooks by running the following commands on the vCMTS ansible host.

```
cd $MYHOME/vcmts/tools/vcmts-pm
source env/bin/activate
vcmuts-pm config-bmra
deactivate
```
2.6.3 Install Intel® IPSEC MB Library

While BMRA installs many components it doesn't install the Intel® IPSEC MB Library which is required by the vCMTS dataplane. This must be installed by the user before deploying the BMRA ansible scripts.

Install Intel® IPSEC MB Library on the vCMTS dataplane node host OS by running the following command on the **vCMTS dataplane server**.

```
cd $MYHOME
build_baremetal_ipsec_mb
```

⚠️ Please note that this command is run on the **vCMTS dataplane server** while all other commands in this section are ran on the **vCMTS ansible host**.

2.6.4 Deploy BMRA ansible playbooks

The next step is to deploy the ansible playbooks. This step is the core BMRA installation procedure and afterwards a Kubernetes® cluster will be installed consisting of the vCMTS traffic-generator server (which also acts as Kubernetes® controller node) and vCMTS dataplane server.

To deploy the BMRA ansible playbooks, run the following command on the **vCMTS ansible host**.

```
source $MYHOME/vcmts/tools/vcmts-env/env.sh
cd $MYHOME/container-experience-kits
ansible-playbook -i inventory.ini playbooks/remote_fp.yml -vv
```

⚠️ Please note that the BMRA ansible playbooks will take several minutes and should NOT be interrupted. If there are any errors encountered they will be printed to screen.

Once this is complete, BMRA is installed and a Kubernetes cluster is created.

2.6.5 Verify Kubernetes installation and label nodes

A successful Kubernetes installation may be verified by running the following command on the Kubernetes controller node (**traffic-generator server**).

```
kubectl get nodes
```

The output of this command should indicate both nodes, traffic-generator (pktgen) and vCMTS dataplane as ‘Ready’. Each Kubernetes node must be labelled as a pktgen or vcmts node.
Label the nodes by running the commands below on the Kubernetes controller node.

It is important that the node name used for `pktgen_node` and `vcmtsd_node` is the name returned from the `kubectl get nodes` command above.

```
kubectl label nodes pktgen_node vcmtpktgen=true
kubectl label nodes vcmtsd_node vcmts=true
```

The `kubeadm` configuration also needs to be updated to easily get the logs from pods.

Run the following command to do this.

```
kubeadm upgrade node
```
2.7 Install vCMTS dataplane node software

The following sections describe installation of software components required on the vCMTS dataplane node, in addition to those already installed as part of the BMRA installation.

These steps should be performed on the vCMTS dataplane server after logging in as root user.

After logging in to the vCMTS dataplane server, first set the vCMTS reference dataplane environment by running the following command.

```bash
source $MYHOME/vcmts/tools/vcmts-env/env.sh
```

2.7.1 Generate openssl certificates

A cli application is provided as part of the release package that communicates with vCMTS dataplane applications over TLS connection allowing for detailed statistics to be queried. OpenSSL certificates are required to facilitate this connection. Generate them with the following commands.

```bash
cd $MYHOME
generate_openssl_certs
```

⚠️ Please note that it is important that this command completes successfully, as any error here may later cause vCMTS dataplane application initialization failure.

2.7.2 Build vCMTS power manager

Run the following commands to build container images for the vCMTS power management.

```bash
build_container_power_mgr
```

2.7.3 Install Intel pcm-memory tool

The Intel pcm utility is required to monitor memory bandwidth utilization on Grafana. Run the following command to install it on the vCMTS dataplane server.

```bash
install_pcm_tool
```
2.7.4 Build vCMTS dataplane application container
Build the vCMTS dataplane container image by running the following command.

```
build_container_vcmtd
```
2.8 Install vCMTS traffic-generator node software

The following sections describe installation of software components required on the vCMTS traffic-generator node. These steps should be performed after logging in as root user to the **vCMTS traffic-generator server** (which also acts as the Kubernetes controller node).

After logging in to the vCMTS traffic-generator node, set the vCMTS reference dataplane environment by running the following command.

```bash
source $MYHOME/vcmts/tools/vcmts-env/env.sh
```

### 2.8.1 Build Pktgen application container

Run the following command to build the DPDK Pktgen container image.

```bash
build_container_pktgen
```
2.9 Configure vCMTS dataplane

Each vCMTS dataplane instance (POD in Kubernetes*) handles upstream and downstream traffic for a single service-group, which includes a number of cable subscribers, typically hundreds. The following service-group options may be configured.

<table>
<thead>
<tr>
<th>Configurable Service-Group Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment Type</td>
</tr>
<tr>
<td>Power Management</td>
</tr>
<tr>
<td>Power busy hours</td>
</tr>
<tr>
<td>DOCSIS Upstream Scheduler</td>
</tr>
<tr>
<td>SST-BF Configuration</td>
</tr>
<tr>
<td>SST-BF-SGs</td>
</tr>
<tr>
<td>Number of Service-Groups</td>
</tr>
<tr>
<td>Number of Subscribers</td>
</tr>
<tr>
<td>Core configuration</td>
</tr>
</tbody>
</table>
### Installation Guide

#### Channel Configuration

<table>
<thead>
<tr>
<th>Config</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1us1t_1ds2t | 1 US per core (single threaded), 1 DS per core (dual threaded, DEFAULT)  
**NOTE:** stats thread runs on US hyper-thread sibling |
| 1us1t_1ds1t | 1 US per core (single threaded), 1 DS per core (single threaded)  
**NOTE:** stats thread runs on US hyper-thread sibling |
| 2us1t_1ds2t | 2 US per core (single threaded), 1 DS per core (dual threaded)  
**NOTE:** baremetal mode only, stats threads run on shared core |
| 2us1t_1ds1t | 2 US per core (single threaded), 1 DS per core (single threaded)  
**NOTE:** baremetal mode only, stats threads run on shared core |

**Default:** 1us1t_1ds2t

#### One of the following channel configurations:

- 1 x OFDM, 32 x SC-QAM
- 2 x OFDM, 32 x SC-QAM
- 4 x OFDM, 0 x SC-QAM
- 6 x OFDM, 0 x SC-QAM

**Default:** 6xOFDM

#### CM Crypto Configurations

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES or OFF</td>
<td>AES or OFF</td>
</tr>
</tbody>
</table>

**Default:** AES

#### AES Key Size

<table>
<thead>
<tr>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>128, 256</td>
<td>AES key size (applies to all service groups using AES crypto type)</td>
</tr>
</tbody>
</table>

**Default:** 128

#### CPU Cycle Count Stats Capture

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
</table>
| Enable or Disable | Enable or Disable CPU Cycle count statistics capture per service group  
**NOTE:** Enabling CPU Cycle count statistics has an impact on dataplane performance |

**Default:** Disable Cycle count statistics capture

#### Application Stats Capture

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
</table>
| Enable or Disable | Enable or Disable detailed application statistics capture per service group  
**NOTE:** Enabling detailed application statistics has an impact on dataplane performance |

**Default:** Disable detailed application statistics capture

#### Latency Stats Capture

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable or Disable</td>
<td>Enable or Disable Latency statistics capture per service group</td>
</tr>
</tbody>
</table>

**Default:** Disable Latency statistics capture

#### Downstream CRC re-generation

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
</table>
| Enable or Disable | Enable or Disable CRC re-generation for downstream DOCSIS frames  
**NOTE:** CRC is disabled for upstream traffic |

**Default:** Enable CRC

#### Downstream Crypto Offload

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable or Disable</td>
<td>Enable or Disable QuickAssist offload for downstream encryption</td>
</tr>
</tbody>
</table>

**Default:** Disable QAT
Select iMix or fixed-sized packets for cable traffic.

**iMix1:**
- **Upstream:** 65% : 84B, 18% : 256B, 17% : 1280B
- **Downstream:** 15% : 84B, 10% : 256B, 75% : 1280B

**iMix2:**
- **Upstream:** 81.5% : 70B, 1% : 940B, 17.5% : 1470B
- **Downstream:** 3% : 68B, 1% : 932B, 96% : 1520B

**Fixed-size Packets:**
- 64B, 256B, 512B, 640B, 768B, 1024B, 1280B or 1536B

**Default:** iMix2

The following settings are applied to all service-groups and are not configurable.

**Table 6  vCMTS dataplane fixed service-group options**

<table>
<thead>
<tr>
<th>Fixed Settings</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Subscriber Lookup</strong></td>
<td>4 IP addresses per subscriber</td>
</tr>
<tr>
<td><strong>DOCSIS Filtering</strong></td>
<td>16 filters per cable-modem</td>
</tr>
<tr>
<td></td>
<td>10% matched (permit rule), 90% unmatched (default action - permit)</td>
</tr>
<tr>
<td><strong>DOCSIS Classification</strong></td>
<td>16 IPv4 classifiers per cable-modem</td>
</tr>
<tr>
<td></td>
<td>10% matched, 90% unmatched (default service-flow queue)</td>
</tr>
<tr>
<td><strong>Downstream Service-Flow Scheduling</strong></td>
<td>8 service-flow queues per cable-modem (4 active)</td>
</tr>
<tr>
<td><strong>Downstream Channel Scheduling</strong></td>
<td>42.24 Mbps Bandwidth per SQ-QAM channel</td>
</tr>
<tr>
<td></td>
<td>1.89 Gbps Bandwidth per OFDM channel</td>
</tr>
<tr>
<td></td>
<td>99% profile density ratio for OFDM channels</td>
</tr>
<tr>
<td><strong>Downstream Channel Bonding Groups</strong></td>
<td>1xOFDM, 32xSC-QAM</td>
</tr>
<tr>
<td></td>
<td>BG1: Channel 0-23 (SC-QAM), Channel 32 (OFDM)</td>
</tr>
<tr>
<td></td>
<td>BG2: Channel 0-15, 24-31 (SC-QAM), Channel 32 (OFDM)</td>
</tr>
<tr>
<td></td>
<td>BG3: Channel 0-7, 16-31 (SC-QAM), Channel 32 (OFDM)</td>
</tr>
<tr>
<td></td>
<td>BG4: Channel 8-31 (SC-QAM), Channel 32 (OFDM)</td>
</tr>
<tr>
<td></td>
<td>2xOFDM, 32xSC-QAM</td>
</tr>
<tr>
<td></td>
<td>BG1: Channel 0-23 (SC-QAM), Channel 32,33 (OFDM)</td>
</tr>
<tr>
<td></td>
<td>BG2: Channel 0-15, 24-31 (SC-QAM), Channel 32,33 (OFDM)</td>
</tr>
<tr>
<td></td>
<td>BG3: Channel 0-7, 16-31 (SC-QAM), Channel 32,33 (OFDM)</td>
</tr>
<tr>
<td></td>
<td>BG4: Channel 8-31 (SC-QAM), Channel 32,33 (OFDM)</td>
</tr>
<tr>
<td></td>
<td>4xOFDM</td>
</tr>
<tr>
<td></td>
<td>BG1: Channel 0,1 (OFDM)</td>
</tr>
</tbody>
</table>
The maximum amount of downstream traffic that needs to be handled for a service group is determined by its number of OFDM and SC-QAM channels as shown in the table below. It is assumed that upstream traffic is 10% of downstream traffic.

### Table 7  Service-group bandwidth guide

<table>
<thead>
<tr>
<th>Number OFDM Channels (1.89 Gbps)</th>
<th>Additional SC-QAM Channels (42.24 Mbps)</th>
<th>Total Downstream Bandwidth per SG * (Gbps)</th>
<th>Max Total BW (US = 10% DS) (Gbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32</td>
<td>3.24</td>
<td>3.56</td>
</tr>
<tr>
<td>2</td>
<td>32</td>
<td>5.13</td>
<td>5.64</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>7.56</td>
<td>8.32</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>11.34</td>
<td>12.47</td>
</tr>
</tbody>
</table>

* Actual downstream channel bandwidth is reduced by DOCSIS MAC and PHY overhead

**NOTE:** Key points to consider when selecting service-group options.

- total available QAT crypto bandwidth should be considered when selecting number of service-groups for QAT crypto offload option (noting that QAT offload is only used for downstream traffic)
- when using 10G NICs with 6xOFDM channel configuration, the Ethernet port bandwidth limit may be reached before downstream channel bandwidth limit

The required service-group settings may be configured on the system using the `vcmts-pm` tool provided in the vCMTS dataplane release package, as described in the next section.

### 2.9.1 Configure vCMTS dataplane service-group options

Follow the steps below to specify service-group options for vCMTS dataplane instances on the system. These steps should be performed as root user on the **vCMTS ansible host**.

After logging in to the Kubernetes controller node, set the environment for running the `vcmts-pm` tool.

```
source $MYHOME/vcmts/tools/vcmts-env/env.sh
cd $MYHOME/vcmts/tools/vcmts-pm
source env/bin/activate
```
Then, run the `vcmts-pm` tool provided in the vCMTS dataplane release package as follows to specify the required service-group options for the vCMTS dataplane environment.

```bash
cvmts-pm config-service-groups
```

Follow the prompts to select service-group settings and generate Helm charts for the Kubernetes.

Service-group options may also be selected by passing command-line arguments to the `vcmts-pm` tool, as described below.

For example, to configure a BMRA Kubernetes environment with 16 service-groups using the `1us1t_1ds2t` core configuration with 6 OFDM channels, 300 subscribers, all AES-128 CM's, CRC enabled, SG's 8 to 11 with QAT offload, SG 1 with Latency stats and iMix-2 traffic, run the following command:

```bash
cvmts-pm config-service-groups -u -i bmra_k8s --core-config 1us1t_1ds2t -n 16 -c 6ofdm -s 300 -d aes -k 128 -v enabled -q "8,9,10,11" -z "1" -t imix2
```

A full description of `vcmts-pm` command-line arguments for service-group configurations can be examined by running the following command.

```bash
cvmts-pm config-service-groups --help
```
2.10 Run vCMTS dataplane and traffic-generator software

The following sections describe how to start, stop and re-start the vCMTS reference dataplane and traffic-generator software, and also how to verify that the software is running correctly.

These steps should be performed after logging in as root user to the vCMTS ansible host.

Note that these steps must be run on a system that already has the BMRA ansible playbooks deployed. All commands use the vcmts-pm tool.

After logging in to the vCMTS ansible host, set the vcmts reference dataplane environment by running the following command.

```bash
source $MYHOME/vcmts/tools/vcmts-env/env.sh
cd $MYHOME/vcmts/tools/vcmts-pm
source env/bin/activate
```

2.10.1 Start infrastructure and application instances

Run the following environment functions on the Kubernetes controller node to start vCMTS US-Sched, vCMTS dataplane and Pktgen application instances as well as the supporting components (power-mgr if selected).

```bash
vcmts-pm infra-start
vcmts-pm us-sched-start
vcmts-pm vcmtsd-start
vcmts-pm pktgen-start
```

Verify that all components started correctly by running the following commands.

```bash
vcmts-pm infra-status
vcmts-pm us-sched-status
vcmts-pm vcmtsd-status
vcmts-pm pktgen-status
```

The Kubernetes log function can be used to diagnose vcmtsd application issues.

E.g. the following commands may be run to check logs of dataplane containers for vcmtsd pod number 0 (i.e. service-group ID 0)
2.10.2 How to stop infrastructure and application instances

To stop vCMTS dataplane and Pktgen application instances as well as supporting components, run the following commands.

```
kubectl logs pod/vcmtsd-0
```

```
vcmts-pm pktgen-stop
vcmts-pm vcmtsd-stop
vcmts-pm us-sched-stop
vcmts-pm infra-stop
```
2.11 System verification

The following sections describe how to verify that the system is running correctly.

2.11.1 Start dataplane traffic

This section covers how to generate simulated cable traffic into the vCMTS dataplane server. Upstream and downstream DOCSIS dataplane traffic may be simulated by running the `vcmts-pm` tool provided in the vCMTS dataplane release package.

These steps should be performed as root user on the vCMTS ansible host.

After logging in to the Kubernetes controller node, set the environment for running the vcmts-pm tool.

```bash
source $MYHOME/vcmts/tools/vcmts-env/env.sh
cd $MYHOME/vcmts/tools/vcmts-pm
source env/bin/activate
```

Next, start traffic.

Note that because dataplane NW interface VF’s are dynamically allocated to POD’s by the Kubernetes SR-IOV device plugin, an ARP request-response handshake is required between correlating Pktgen and vCMTS application instances for correct routing of traffic. This is implicitly performed before starting to send traffic when the Pktgen start command is sent below.

For example, to start traffic at 5Gbps (20% of 25G line-rate) for downstream and 0.5Gbps (2% of 25G line-rate) for upstream for 16 vCMTS dataplane instances (0 to 15), run the following commands.

```bash
vcmts-pm traffic-rate -i 0-15 us -r 2
vcmts-pm traffic-rate -i 0-15 ds -r 20
vcmts-pm traffic-arp -i 0-15 usds
vcmts-pm traffic-start -i 0 15 usds
```

To reduce the traffic-rate to 2.5Gbps (10% of 25G line-rate) for downstream and 0.25Gbps (1% of 25G line-rate) for upstream for 16 vCMTS dataplane instances (0 to 15), run the following commands.

```bash
vcmts-pm traffic-rate -i 0-15 us -r 1
vcmts-pm traffic-rate -i 0-15 ds -r 10
```

To stop traffic, run the following command.

```bash
vcmts-pm traffic-stop -i 0 15 usds
```
If using 10G NICs the pktgen-rate values above are percentages of 10G line-rate instead of 25G and if using 100G NICs the pktgen-rate values above are percentages of 100G line-rate.

An RFC 2544 throughput measurement test may also be run. For example, to measure downstream throughput for service-group 0, run the following command.

```
vcmts-pm traffic-measure -i 0 ds -m rx
```

### 2.11.2 Check the Grafana dashboard

vCMTS dataplane and platform metrics may be checked on a Grafana dashboard. The Grafana dashboard is exposed on port 30000 on any server in the Kubernetes cluster. To view the Grafana dashboard open a web browser and go to the following URL.

```
https://dataplane-hostname:30000
```

The default credentials for Grafana at this point will be "admin:admin".

⚠️ If accessing from a web-browser on a remote client an SSH tunnel to the Grafana dashboard service port on the vCMTS dataplane node may be required.
Once signed in, select the required vCMTS dataplane dashboard by clicking on Dashboards Home. e.g. the System Summary dashboard is shown below. You can easily search for the vCMTS specific dashboards by using the tag "vCMTS".
Figure 9  Intel vCMTS Dataplane Reference Platform – System Summary Dashboard
2.11.3 vCMTS CLI Tool

The vCMTS reference dataplane system includes a utility application that provides a command line interface to query status and statistics for each vCMTS service-group which is active on the system.

The CLI application should be run from the vCMTS dataplane server. It can be run with the following commands. Type ‘help’ to view the usage.

```
source $MYHOME/vcmts/tools/vcmts-env/env.sh
cd tools/vcmts-cli
python3 vcmts_cli.py
```

In order to connect to a running vCMTS service group, it needs to be added to the CLI with the following command in the CLI shell.

```
vcmds-cli> add vcmts <service-group-id> <ip> <tls-port-1>
```

The service group ID of the required service group should be entered in place of `<service-group-id>`.

The IP address of the vCMTS dataplane pod should be entered in place of `<ip>`. This can be determined by running the following command on the vCMTS traffic-generator server.

```
kubectl describe pod vcmtsd-<service-group-id> | grep IP
```

The TLS port number of the vCMTS application instance should be entered in place of `<tls-port-1>`.

The TLS port number may be calculated as follows: 8100 + service-group-id.

Once added to the CLI, vCMTS service-group statistics can be queried through the CLI tool as shown below.

```
vcmds-cli> help
+-----------------------------------------------------------------------------------+
<table>
<thead>
<tr>
<th>vCMTS CLI Command List</th>
<th>vCMTS CLI Command List</th>
</tr>
</thead>
<tbody>
<tr>
<td>help</td>
<td>Prints help string</td>
</tr>
<tr>
<td>quit</td>
<td>Exits CLI program</td>
</tr>
<tr>
<td>add vcmts &lt;service-id&gt;</td>
<td>Save given Service Group information</td>
</tr>
<tr>
<td>del &lt;service-id&gt;</td>
<td>Deletes saved Service Group information</td>
</tr>
<tr>
<td>list</td>
<td>Displays saved Service group information</td>
</tr>
<tr>
<td>status &lt;service-id&gt;</td>
<td>Requests status of given Service Group</td>
</tr>
<tr>
<td>summary</td>
<td>Displays a summary of Service Groups</td>
</tr>
<tr>
<td>stats &lt;service-id&gt;</td>
<td>Query stats from running vCMTS instance</td>
</tr>
</tbody>
</table>
+-----------------------------------------------------------------------------------+
```
2.11.4 Attach to application instances

The vcmts-pm tool may be used to generate instructions on how to directly attach to specific vCMTS dataplane or pktgen applications. Run the following commands for instructions on how to attach directly to an application instance running in either Kubernetes® or bare-metal Linux® environment.

```
source $MYHOME/vcmts/tools/vcmts-env/env.sh
cd $MYHOME/vcmts/tools/vcmts-pm
source env/bin/activate

vcmts-pm pktgen-attach -i 0 us
vcmts-pm vcmtsd-attach -i 0 us
```

**NOTE:** after attaching to a process issuing Ctrl^C command will terminate the application. Instead use Ctrl^Z to detach.
## 2.12 System reboot

After reboot of vCMTS dataplane and traffic-generation servers, core components managed by BMRA such as Kubernetes and system infrastructure jobs should start automatically.

To restart vCMTS dataplane and traffic-generator software the following commands should be run on the **vCMTS ansible host**, after logging in as root user.

```bash
source $MYHOME/vcmts/tools/vcmts-env/env.sh
vcmts-pm infra-start
vcmts-pm us-sched-start
vcmts-pm vcmtsd-start
vcmts-pm pktgen-start
```

Verify that all components started correctly by running the following commands.

```bash
vcmts-pm infra-status
vcmts-pm us-sched-status
vcmts-pm vcmtsd-status
vcmts-pm pktgen-status
```

Perform system verification as described in previous section.
3 Appendix

3.1 Tuning P-states for optimum performance

Some server compute-board types may require tuning to prevent the host OS from controlling P-states which may lead to unexpected CPU core frequency transitions and subsequent impact to vCMTS dataplane performance. This section should be followed if the power management features of the Intel reference dataplane system (as described in section 1.1.4) are not being used and the user wants to configure the server for maximum performance.

⚠️ Please note that P-states need only be tuned on the vCMTS dataplane server.

3.1.1 Install power management utilities tool

If not already done as part of the main install process, the power management utilities tool to control P-states must be installed. This can be done by logging into the vCMTS dataplane server and running the following steps.

```
source $MYHOME/vcmts/tools/vcmts-env/env.sh
install_power_mgmt_utilities
```

3.1.2 Determine current P-state settings

To determine the current P-state settings, run the newly installed tool. When the menu below appears, select option ‘2’ to “Display Current Settings”.

```
Appending on the CPU of the host platform, the displayed output will be some variation of what is shown below. Note that the minimum and maximum P-states may differ, allowing the OS to vary the frequency of CPU cores, which may have an undesirable performance impact.

```bash
# cd $MYHOME/CommsPowerManagement/
# python power.py
----------------------------------------------------------
[1] Display Available Settings
[2] Display Current Settings
[3] Display Available P-States
[4] Set P-State governor for a range of cores
[5] Set Maximum P-State for a range of cores
[6] Set Minimum P-State for a range of cores
[7] Display Available C-States
[8] Enable C-State for a range of cores
[9] Disable C-State for a range of cores
[h] Show Help Text
[q] Exit Script
----------------------------------------------------------
Option: 2
```

```
P-STATE INFO
Core Max Min  Now Governor
----- ------ ------ ------ -------
 0 2200  800   798 ondemand
 1 2200  800   801 ondemand
  ..
  ..
126 2200  800   962 ondemand
127 2200  800   800 ondemand
```

Press enter to continue ...

Depending on the CPU of the host platform, the displayed output will be some variation of what is shown below. Note that the minimum and maximum P-states may differ, allowing the OS to vary the frequency of CPU cores, which may have an undesirable performance impact.
3.1.3 Configure and Tune P-state settings

To ensure frequency transitions don’t occur, the minimum and maximum P-state frequencies must be set to the same value. This can be done by entering by option 6 in the tool menu as shown below to “Set Minimum P-State for a range of cores”.

When prompted, enter the P-state minimum frequency value, ensuring that it matches the configured P-state maximum frequency. In this example that P-state frequency value is “2200”, but this may differ per CPU. Matching P-state frequency values must be set for the full range of CPU cores, which in this example is “0-127”. Again, this may differ per CPU.

⚠️ Note that increasing the P-state minimum frequency of cores not used for vCMTS dataplane processing will not cause any unnecessary extra power consumption; if cores are idle or low utilization they will enter power saving C-states and thus run at the lower C-state core frequency.

```
---------
[1] Display Available Settings
[2] Display Current Settings

[3] Display Available P-States
[4] Set P-State governor for a range of cores
[5] Set Maximum P-State for a range of cores
[6] Set Minimum P-State for a range of cores

[7] Display Available C-States
[8] Enable C-State for a range of cores
[9] Disable C-State for a range of cores

[h] Show Help Text
[q] Exit Script
---------

Option: 6

Available P-States: [2200, 2100, 2000, 1900, 1800, 1700, 1600, 1500, 1400, 1300, 1200, 1100, 1000, 900, 800]

Input P-State: 2200

Input Range of Cores: 0-127

Writing 2200000 to /sys/devices/system/cpu/cpu0/cpufreq/scaling_min_freq
Writing 2200000 to /sys/devices/system/cpu/cpu1/cpufreq/scaling_min_freq
..
Writing 2200000 to /sys/devices/system/cpu/cpu126/cpufreq/scaling_min_freq
Writing 2200000 to /sys/devices/system/cpu/cpu127/cpufreq/scaling_min_freq
```
3.1.4 Verify P-state settings

Once the P-state settings have been re-configured, menu option 2, "Display Current Settings" should be re-selected to verify that the correct settings have been applied.

The maximum and minimum P-state frequencies should now be set to the same value as shown in the example below.

```
<table>
<thead>
<tr>
<th>Core</th>
<th>Max</th>
<th>Min</th>
<th>Now</th>
<th>Governor</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2200</td>
<td>2200</td>
<td>2200</td>
<td>ondemand</td>
</tr>
<tr>
<td>1</td>
<td>2200</td>
<td>2200</td>
<td>2200</td>
<td>ondemand</td>
</tr>
<tr>
<td>..</td>
<td>..</td>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>126</td>
<td>2200</td>
<td>2200</td>
<td>2200</td>
<td>ondemand</td>
</tr>
<tr>
<td>127</td>
<td>2200</td>
<td>2200</td>
<td>2200</td>
<td>ondemand</td>
</tr>
</tbody>
</table>
```

Press enter to continue ...
3.2 DOCSIS DDP Forwarding Sample Application

3.2.1 Description

This sample application (docsis-ddp-fwd) demonstrates how the Dynamic Device Profile (DDP), Flow-director and RSS features of the Intel 800 series NIC can be used to enable optimum steering of DOCSIS MAC upstream and downstream traffic to Rx (receive) software threads.

DOCSIS MAC dataplane traffic-steering rules may be based on any of the following:

1. L2TPv3 session ID: which indicates the type of upstream frame according to DOCSIS MAC protocol specifications.
2. DSCP value: which indicates flow priority.
3. VLAN ID: which indicates a vCMTS application instance, possibly representing a service-group, to be used for downstream data-plane processing.

**NOTE:** Flow-director support for L2TPv3 session ID rules is enabled in the Intel 800 series NIC through use of the Dynamic Device Profile (DDP) feature of Intel NIC's.

Two distinct modes are used by the docsis-ddp-fwd sample application to steer incoming frames to Virtual Functions (VFs) or NIC Rx (i.e. Receive) Queues as described below. Each Rx queue is polled by a dedicated DPDK PMD thread.

**Mode 1 - traffic steering to multiple Rx queues on a Single VF as follows:**

1. L2TP session ID is used to steer L2TPv3 tunneled upstream DOCSIS traffic to 1 or 2 Rx queues (2 by default):
   a. if 2 queues are selected, upstream frames are steered to separate Rx queues, one for frames with L2TP session ID = 1, and another for frames with L2TP session ID = 2
   b. if 1 queue is selected, all upstream frames are steered to the same Rx queue
2. Downstream (non-L2TPv3) IP frames are distributed by RSS based on IP 5-tuple to either 2 or 3 Rx queues (3 by default).

**Mode 2 - traffic steering to multiple VF's and multiple Rx queues as follows:**

1. VLAN tagged Frames with VLAN ID = 1 are steered to VF 1 Rx queues
   a. frames with DSCP value CS0 (0) are steered to VF 1, Rx queue 0
   b. frames with DSCP value EF (46) are steered to VF 1, Rx queue 1
2. L2TPv3 frames with L2TP session ID 1 are directed to VF 2, Rx queue 0.
3. L2TPv3 frames with L2TP session ID 2 are directed to VF 3, Rx queue 0.

⚠️ Please note that this sample application must be used in conjunction with DPDK Pktgen and custom-generated pcap's are provided to demonstrate the use of NIC rules for DOCSIS MAC dataplane traffic-steering.
Please note also that this application will only work with an Intel 800 series 100G NIC Virtual Function with latest NIC firmware and L2TPv3 DDP enabled.

### 3.2.2 User Guide

The following sections provide instructions to install and run the `docsis-ddp-fwd` sample application and to generate traffic to demonstrate the DOCSIS MAC traffic steering features of Intel 800 series NIC’s.

#### 3.2.2.1 Installing the DOCSIS DDP Fwd sample application

Follow these steps to Install the `docsis-ddp-fwd` sample application (on the vCMTS server):

Ensure the Intel 800 series NIC driver and DDP ICE packages below are installed.

Download links:


Use ICE Driver version &gt;=1.6.7
Use ICE Comms package version &gt;=1.3.30.0

Untar the ICE driver package, go to the driver src directory and install the ICE driver.

```bash
  tar -xvf ice-1.6.7.tar.gz
  cd ice-1.6.7/src/
  make -j 8
  make install
```

Unzip the ICE Comms package, copy the extracted package to the ICE DDP directory. Remove the previous ICE comms package from the ICE DDP directory and rename the new ICE comms package.
If a previous ICE driver is loaded, unload it. Load the new ICE driver and check it was loaded correctly.

```
rmmod ice
gmodprobe ice
modinfo ice
```

If a line containing "version: 1.1.4" is present, the ICE driver was loaded correctly.

Check for successful DDP package installation.

```
journalctl -n 200
dmesg | grep ice
```

If a line containing "The DDP package was successfully loaded: ICE COMMS Package version 1.3.17.0" is present, the DDP package was installed correctly.

Set the environment normally used for vCMTS.

```
export VCMTSD_HOST=y
source $VCMTS_ROOT/tools/vcmts-env/env.sh
```

Install base packages, install DPDK (if not already installed) and build the sample application.

```
install_base_ubuntu_pkgs
build_baremetal_dpdk
cd $VCMTS_ROOT/src/docsis-ddp-fwd
make
```

The *docsis-ddp-fwd* sample application has now been successfully installed on the vCMTS dataplane server.

It is assumed that traffic-generation software has already been installed on the vCMTS traffic-generator server. This can be used to generate traffic for the *docsis-ddp-fwd* sample application.
3.2.2.2 Running the DOCSIS DDP Fwd sample application

The following steps describe how to run the docsis-ddp-fwd sample application.

Select a device from the list of PCI devices on the system and create VF's for this device.

```
dpdk-devbind.py -s
echo 4 > /sys/class/net/<device-name>/device/sriov_numvfs
```

⚠️ Please note that if BMRA has been installed, this procedure will remove all VF's on the selected PCI device. To restore the BMRA setup, the following command should be executed.

```
systemctl restart bmra_sriov_init.service
```

Bind each of the newly created VF's to DPDK for use by the docsis-ddp-fwd sample application using the following command.

```
dpdk-devbind.py -s
dpdk-devbind.py -b vfio-pci <vf-name>
```

Launch the docsis-ddp-fwd sample application in Mode 1 or 2 as shown below using the helper script provided.

**Mode 1:**

```
$VCMTS_ROOT/src/docsis-ddp-fwd/start_docsis_ddp_fwd.sh -l [core_list] -a [pci_address]
```

**Mode 2:**

First, enable flow-director-based VF switching in the NIC by running the following command to set the first of the newly created VF's as a trusted VF.

```
ip link set <device-name> vf 0 trust o
```

Then, launch the docsis-ddp-fwd sample application in Mode 2.

```
```
Below is a description of `docsis-ddp-fwd` helper script command-line arguments.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-l [cores]</code></td>
<td>Comma separated list of cores to use. Example: <code>-l 2,3,4,5,6,7</code>. Note: number of cores must equal the total number of queues + 1. For 4 queues, 5 cores must be allocated. Defaults to 6 cores (5 queues).</td>
</tr>
<tr>
<td><code>-a [pci address]</code></td>
<td>PCI address of virtual function to be used. Example: <code>-a 18:01.1</code>.</td>
</tr>
<tr>
<td><code>-r [nb of RSS queues]</code></td>
<td>Number of RSS queues for downstream traffic. Note: optional, default 3, max 3. Mode 1 only.</td>
</tr>
<tr>
<td><code>-t [nb of flow-director queues]</code></td>
<td>Number of Flow-Director queues for upstream traffic. Note: optional, default 2, max 2. Mode 1 only.</td>
</tr>
<tr>
<td><code>-m [1/2]</code></td>
<td>Select Mode 1 or 2. Note: optional, default 1.</td>
</tr>
</tbody>
</table>
Below are additional command-line arguments used by the `docsis-ddp-fwd` application, which are defaulted by the helper script or the sample application.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--:</code></td>
<td>Used to separate DPDK EAL arguments and application arguments</td>
</tr>
<tr>
<td><code>-p [bitmask]</code></td>
<td>hexadecimal bitmask of VF ports to configure</td>
</tr>
<tr>
<td><code>note:</code> defaults to 0x1</td>
<td></td>
</tr>
<tr>
<td>`-n</td>
<td>--ip-addr-network [ip address]`</td>
</tr>
<tr>
<td><code>note:</code> must be set to 192.168.1.100</td>
<td></td>
</tr>
<tr>
<td>`-y</td>
<td>--ip-address-rphy [ip address]`</td>
</tr>
<tr>
<td><code>note:</code> must be set to 192.168.1.200</td>
<td></td>
</tr>
<tr>
<td><code>-c [period]</code></td>
<td>statistics generation period in seconds</td>
</tr>
<tr>
<td><code>note:</code> defaults to 1 second</td>
<td></td>
</tr>
<tr>
<td><code>-s [period]</code></td>
<td>statistics print period in seconds</td>
</tr>
<tr>
<td><code>note:</code> defaults to 1 seconds</td>
<td></td>
</tr>
<tr>
<td><code>-m [1/2]</code></td>
<td>Select Mode 1 or 2</td>
</tr>
<tr>
<td><code>Note:</code> optional, default 1</td>
<td></td>
</tr>
</tbody>
</table>

Note that the `docsis-ddp-fwd` sample application can be run without the helper script and use all of the available arguments described above.
3.2.2.3 Generating traffic into the DOCSIS DDP Fwd Sample application

Once the `docsis-ddp-fwd` sample application is running, Pktgen is required to send traffic to the application.

First, log into the Pktgen server and set the vCMTS Pktgen environment. Note that it is assumed that the Pktgen environment has already been installed using the vCMTS traffic-generation installation procedure.

```bash
export PKTGEN_HOST=y
source $VCMTS_ROOT/tools/vcmts-env/env.sh
```

⚠️ The following command must be run to disable spoof-checking for the NIC VF’s in order to allow the use of VLAN tagged packets for mode 2 of the `docsis-ddp-fwd` sample application.

```bash
ip link set <device-name> vf 0 spoofchk off
ip link set <device-name> vf 1 spoofchk off
```

Then, run Pktgen as follows using the provided helper script.

```bash
$VCMTS_ROOT/src/docsis-ddp-fwd/start_docsis_ddp_fwd_pktgen.sh \   -l [core list] (Only 4 cores must be used and the selected cores must be on the same socket as the PCI device) \   -a [US pci address] \   -a [DS pci address] \   -p [imix1|imix2] (optional, default is imix2)
```
The full list of Pktgen helper script command-line arguments are as follows:

- `l [core list]`: Comma separated list of CPU cores to which the Pktgen threads will be affinitized
  note: only 4 cores in list are supported and the PCI device and selected cores must be on the same socket. e.g. `-l 2,3,4,5`

- `a [pci address]`: Virtual Function PCI address to use for Pktgen network interface
  note: two ports must be provided, one for the upstream interface and second for the downstream interface. The PCI device and selected cores must be on the same socket.

- `p [imix1|imix2]`: PCAP type to use

- `m [1/2]`: Select Mode 1 or 2
  Note: optional, default 1

Below are additional command-line arguments used by the Pktgen application which are defaulted by the helper script.

- `--file-prefix [string]`: File prefix

- `--`: Used to separate DPDK EAL arguments and application arguments

- `m [string]`: Matrix for mapping ports to logical cores.
  e.g. `-m "[26:27].0"` core 26 and 27 handles port 0 rx/tx.

- `P`: Enable promiscuous mode on all ports

- `T`: Enable color terminal output in VT100

- `s [P:file]`: The pcap packet file to stream for the specified port “P”

- `f [file]`: The script command file (.pkt) or lua script (.lua) to execute upon pktgen startup.
Note that the Pktgen application can also be run without the helper script and use all of the available arguments described above.