TECHNOLOGY GUIDE

Intel Corporation

Intel® Ethernet 700/800 Series - Dynamic Device Personalization Support for CNF with Kubernetes

Authors

Abdul Halim

Killian Muldoon

Martin Kennelly

1 Introduction

Cloud Native is fast becoming a first-class deployment model for next generation network operations. Performance tuning can be difficult to manage in these setups, however, as gaps in hardware awareness exist at the Orchestration level.

Dynamic Device Personalization (DDP), a feature available in some Intel® Ethernet Controllers including Intel® Ethernet Controller X710, XXV710, XL710, E810 and Intel® Ethernet Network Adapter X710, XXV710, XL710, X722, E810, offers a straightforward way to customize packet processing on a network card, improving throughput and performance.

This document provides an overview of DDP technology use cases and describes how to orchestrate container workloads on a Kubernetes (K8s) cluster that can take advantage of these performance improvements by automating the application of DDP packages and provisioning them to specific containerized applications. This technology guide is applicable to cluster administrators and data center architects.

DDP enables certain Intel® Ethernet 700 and 800 Series to reprogram its packet processing pipeline to identify new protocols. When the specific packet type is detected, protocol-specific flow rules can be automatically added into its control plane to improve key performance characteristics.

Communication protocol definitions are aggregated into DDP package software. For example, Intel® Ethernet 810 series telecommunications DDP package contains five communication protocols including GTP and PPPoE. Several different DDP packages are available to download and to be loaded into an adapter to suit specialized use cases across a broad swathe of network functions.

This document is part of the Network Transformation Experience Kit, which is available at https://networkbuilders.intel.com/network-technologies/network-transformation-exp-kits.

Table of Contents

Figures

Tables

Document Revision History

REVISION	DATE	DESCRIPTION
001	April 2020	Initial release of document.
002	February 2021	Added Intel® Ethernet Network Adapter 800 Series DDP Orchestration support information utilizing SR-IOV Device Plugin and Container Bare Metal Reference Architecture (BMRA).

1.1 Terminology

Table 1. Terminology

ABBREVIATION	DESCRIPTION
CNFs	Cloud-Native Network Functions
DDP	Dynamic Device Personalization
DPDK	Data Plane Development Kit
GPRS	General Packet Radio Service
GPU	Graphics Processor Unit
K8s	Kubernetes
NFV	Network Function Virtualization
NIC	Network Interface Card
OOM	Out of Memory
NPWG	Network Plumbing Working Group
PF	Physical Function
PPPoE	Point-to-Point Protocol over Ethernet
RAM	Random Access Memory
SR-IOV	Single-Root Input / Output Virtualization
Tbps	Terabytes per second
vBNG	Virtual Broadband Network Gateway
vEPC	Virtual Evolved Packet Core
VF	Virtual Function
UPF	User Plane Function

1.2 Reference Documentation

Table 2. Reference Documents

REFERENCE	SOURCE
Container Experience Kit GitHub repository	https://github.com/intel/container-experience-kits
Container Bare Metal for 2nd Generation Intel® Xeon® Scalable Processor Reference Architecture	https://builders.intel.com/docs/networkbuilders/container-bare-metal-for-2nd-generation-intel-xeon-scalable-processor.pdf
Dynamic Device Personalization (DDP) Brief	https://www.intel.com/content/www/us/en/architecture-and-technology/ethernet/dynamic-device-personalization-brief.html
Intel® Ethernet Controller 700 Series GTPv1 - Dynamic Device Personalization Technology Guide	https://builders.intel.com/docs/networkbuilders/intel-ethernet-controller-700-series-gtpv1-dynamic-device-personalization.pdf
Intel® Ethernet 800 Series Dynamic Device Personalization Telecommunications Technology Guide	https://cdrdv2.intel.com/v1/dl/getContent/618651
Intel® Ethernet - Dynamic Device Personalization (DDP) - Overview Training Video	https://networkbuilders.intel.com/intel-ethernet-dynamic-device-personalization-ddp-overview-training-video
Low Latency 5G UPF Using Priority Based 5G Packet Classification	https://builders.intel.com/docs/networkbuilders/low-latency-5g-upf-using-priority-based-5g-packet-classification.pdf
5G User Plane Function (UPF)Performance Benefits	https://builders.intel.com/docs/networkbuilders/5g-user-plane-function-upf-performance-with-astri-solution-brief.pdf
SR-IOV Network Device Plugin GitHub repository	https://github.com/k8snetworkplumbingwg/sriov-network-device-plugin
SR-IOV Network Device Plugin – DDP documentation	https://github.com/k8snetworkplumbingwg/sriov-network-device-plugin/tree/master/docs/ddp

2 Overview

DDP is a customizable packet filtering feature in 700/800 series Intel® Ethernet Controllers and Intel® Ethernet Network Adapters (Foundational NICs). DDP, along with the enhanced Data Plane Development Kit (DPDK), supports advanced packet forwarding and highly efficient packet processing for cloud and network function virtualization (NFV) workloads. It allows workload-specific optimization on the NIC, using a programmable packet-processing pipeline.

Note: For more information on enabling DDP packages and which packages are available, refer to: https://www.intel.com/content/www/us/en/architecture-and-technology/ethernet/dynamic-device-personalization-brief.html

As delivered, these Foundational NICs can classify packets for well-known protocols. As the complexity of networking more broadly has grown, so has the number of protocols that underpin network traffic. Identifying combinations of the possible protocols a network is carrying at the correct point is a hard problem, and the lack of information about the structure and nature of these packets can result in reduced performance.

Different protocols will require different actions to be performed — but network packets that are not identified at the Foundational NIC hardware level will only have these required actions revealed after being processed by software on the host. Identifying protocols in the network card allows for processing even before reaching the host, potentially increasing performance across several dimensions.

By looking deeper inside a packet, DDP allows a Foundational NIC to run with packages that empower it to classify new protocols. Analysis based on predetermined packages allows the NIC to take an early decision on the specific path to be followed by the packet. It does this without needing to travel through the software stack — saving CPU cycles and improving overall throughput. Several packages are available and network administrators can load custom DDP packages based on specific use case.

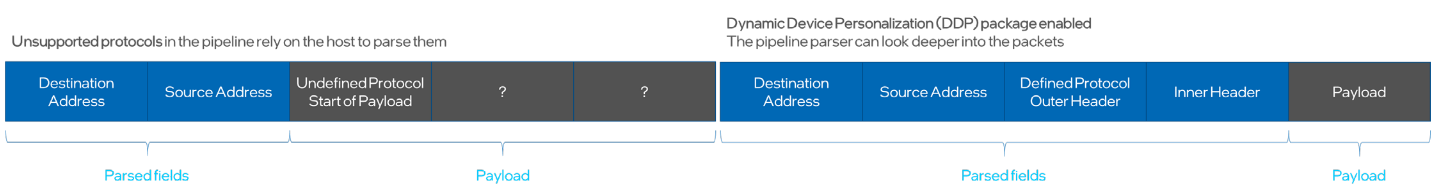

Figure 1. DDP – Packet Classification

Kubernetes (K8s) offers the ability to manage an entire cluster of machines using a single programming interface. With DDP package enablement, cluster operators can choose what DDP package to load onto the individual network cards in a cluster. A single DDP package can program each network card, and every virtual function (VF) using that card as its root will use the same DDP package when piped through into a container. Workloads then can be declared as users of individual DDP packages, causing them to be linked to devices with those packages enabled. This allows an automated, declarative workflow that makes DDP usage scalable and performant in a Kubernetes cluster.

2.1 Challenges Addressed

There are two major facets to orchestrating DDP packages in K8s:

1. Manage and apply DDP packages in K8s running on bare metal deployments.
2. Schedule K8s workloads that request a network interface with a specific DDP package.

Our solution, which is implemented in publicly available open-source projects, addresses both issues and allows:

- DDP package management and loading of package into Foundational NICs on demand
- DDP capability discovery and making K8s aware of available DDP empowered resources
- Dynamic workload placement based on device location

2.2 Use Cases for Intel Ethernet 700 and Intel Ethernet 800 Series

The following use cases are more generally applicable to packages on DDP-enabled Intel® Ethernet 700/800 series network adapters. These features can be enabled for Cloud-Native Network Functions (CNFs) using the process described in subsequent sections.

2.2.1 GTPv1 for Virtualized Enhanced Packet Core

GTPv1 is the General Packet Radio Service (GPRS) Tunneling Protocol used for encapsulation of GPRS packets in wireless communications. A key domain for processing this type of packet is in the virtualized Enhanced Packet Core (vEPC), though it is also a common protocol in the Multi-access Edge. The solution provided here allows CNFs in either the vEPC or Mobile Edge Computing (MEC) to identify packets using GTPv1 before they hit the machine. The technology allows a reduction in packet processing time in a vEPC user plane process, by removing the time spent in software queues.

For more on the use of DDP to process GTPv1 packets, refer to *Intel® Ethernet Controller 700 Series GTPv1 - Dynamic Device Personalization Technology Guide* and *Intel® Ethernet 800 Series Dynamic Device Personalization Telecommunications Technology Guide* (Table 2).

2.2.2 PPPoE for Virtual Broadband Network Gateway

Point-to-Point Protocol over Ethernet (PPPoE) is a protocol for encapsulating point-to-point frames inside Ethernet frames. This allows network providers to take advantage of several Ethernet-specific technologies. PPPoE is one of the dominant protocols operating in virtual Broadband Network Gateway (vBNG) use cases at the network core. As with vEPC and GTP workloads, utilizing the PPPoE DDP package allows for processing to be done in the card itself, keeping packets out of software queues on the target machine.

2.3 Use Cases for Intel® Ethernet 800 Series

The following use cases are applicable to packages on DDP-enabled Intel® Ethernet 800 series network adapters. These features can be utilized to enable Cloud-Native Network Functions (CNFs).

2.3.1 Low Latency User Plane Function (UPF) using Priority Based 5G Packet Classification

Intel® Ethernet 800 series network adapters demonstrate that low latency and jitter can be achieved for a range of applications when using standard server infrastructure for 5G SA UPF. DDP can effectively classify and steer traffic within the server based on control plane, user plane, or between UPF handover interfaces. By applying a DDP package to the network controller the following use cases can be addressed by extending support for protocols:

- 5G GTP support for 5G user plane
- 5G SDAP/PDCP support for 5G NR user plane
- 5G/4G PFCP (CP-UP separation) support
- IP protocols as new flow types, for example L2TPv3, ESP/AH for IPsec

For more information on low latency 5G UPFs on DDP-enabled Intel® Ethernet 800 series, refer to *Low Latency 5G UPF Using Priority Based 5G Packet Classification* (Table 2).

2.3.2 5G User Plane Function Performance Benefits (UPF)

Intel® Ethernet 800 series provides a 100G network interface that delivers almost line rate throughput into each of the UPF instances while allowing users to offload some of the classification, acceleration, and scheduling from your precious hosts cores.

Demonstration of a UPF scaling to over 1.3 Tbps of network traffic and hitting key performance indicators required by network operators around latency in a scalable and management manner shows significant improvement over previous generations.
For more information, refer to the solution brief *5G User Plane Function (UPF) Performance Benefits* (Table 2).

3 Technology Description

This technology guide describes the orchestration of workloads to utilize DDP inside a K8s cluster.

Two projects are used to achieve this:

- Container Bare Metal Reference Architecture (BMRA) script to apply DDP package on a cluster.
- Single-Root Input / Output Virtualization (SR-IOV) Device Plugin to manage devices after they have been configured with DDP packages.

3.1 Deployment with Container Bare Metal Experience Kit (BMRA)

The Container Bare Metal Reference Architecture Experience Kit script is a set of Ansible playbooks that provisions a bare metal K8s cluster. For high performance networking workloads, it installs and configures some of the required advanced features and components. The script is used to automate the provisioning of DDP package and DDP-enabled network cards at cluster creation time.

This deployment uses the Container Bare Metal Reference Architecture (BMRA) script for provisioning the SR-IOV networking with DDP. Refer to *Container Experience Kit GitHub repository* (Table 2) for further details on how to provision your Kubernetes cluster.

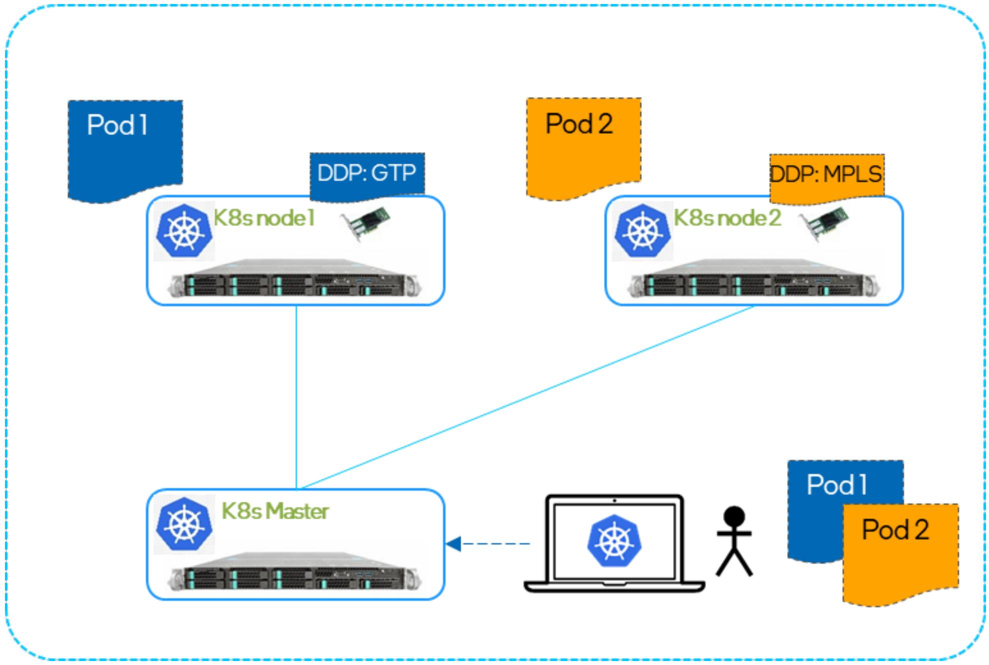

Figure 2. Workloads Deployment in K8s Cluster with SR-IOV Networking and DDP Package Request

3.1.1 Prerequisites

Refer to *Container Bare Metal for 2nd Generation Intel® Xeon® Scalable Processor Reference Architecture* (Table 2) for full prerequisites. Container Experience Kits setup scripts provide a simplified mechanism for installing and configuring Kubernetes clusters on Intel architecture using Ansible.

3.1.2 Intel® Ethernet 700 Series in a Kubernetes Cluster

3.1.2.1 Provisioning DDP

Refer to section 4.4.1 titled 'DDP on Intel 700 Series Network Adapters' of the *Container Bare Metal for 2nd Generation Intel® Xeon® Scalable Processor Reference Architecture* (Table 2) for step-by-step DDP configuration and provisioning guide.

To use specific DDP-enabled virtual functions for 700 series in BMRA, create a resource with a *ddpProfiles* selector by extending the *sriovdp_config_data* variable before deployment.

3.1.2.2 Kubernetes Workload Scheduling with DDP

After deployment, request the resource name *intel_sriov_dpdk_700_series_gtpgo* with the previously configured *ddpProfiles* selector in the workloads pod manifest. Scheduling this pod now is provisioned on an Intel® Ethernet 700 series SR-IOV VF with the DDP profile/package selected in the selector.

3.1.3 Intel® Ethernet 800 Series in a Kubernetes Cluster

3.1.3.1 Provisioning DDP

Refer to section 4.4.2 titled 'DDP on Intel 800 Series Network Adapters' of the *Container Bare Metal for 2nd Generation Intel® Xeon® Scalable Processor Reference Architecture* (Table 2) for full configuration and provisioning guide.

This Foundational NIC supports a broad range of protocols in DDP package images. You can choose the default image or the telecommunications (comms) image that supports market-specific package in addition to protocols in the OS-default package.

Table 3. DDP Package Images

DEFAULT PACKAGE
MAC
EtherType
VLAN
IPv4
IPv6
TCP
ARP

DEFAULT PACKAGE
UDP
SCTP
ICMP
ICMPV6
CTRL
LLDP
VXLAN-GPE
VxLAN (non-GPE)
Geneve
GRE
NVGRE
RoCEv2

COMMS PACKAGE
Extends default profile package with the following protocols:
GTP
PPPOE
L2TPv3
IPsec
PFCP

SR-IOV devices (VFs included) are exposed to the Kubernetes pods via SR-IOV Device Plugin. In Container Experience Kit version 2.0, the plugin cannot yet discover a loaded DDP plugin on Intel Ethernet 800 Series Network Adapters (See *SR-IOV Network Device Plugin GitHub repository* for latest Intel Ethernet 800 Series Network Adapters DDP support (Table 2)). As a result, one cannot fully orchestrate a workload that will require a DDP package on an Intel Ethernet 800 Series Network Adapter.

However, you can work around this issue in the following ways:

1. You can deploy the comms package on every node with an Intel Ethernet 800 Series Network Adapters, or
2. You can deploy the comms package on a subset of Intel Ethernet 800 Series Network Adapters and label the nodes accordingly using Kubernetes labels. Then deploy the workload that requires a specific DDP package with specific Node Selector.

3.1.3.2 Kubernetes Workload Scheduling with DDP

After deployment, request the resource name *intel_sriov_dpdk_800_series* in the workloads pod manifest. Scheduling this pod now is provisioned on an Intel Ethernet 800 Series SR-IOV VF with the DDP package selected from *ddp_profile*. Refer to section 4.4.2 titled 'DDP on Intel 800 Series Network Adapters' of *the Container Bare Metal for 2nd Generation Intel® Xeon® Scalable Processor Reference Architecture* (Table 2) for sample pod manifest.

3.2 Provisioning Workloads on DDP-Enabled Interfaces Using SR-IOV Network Device Plugin

Kubernetes provides a device plugin framework so that devices can be added to manage extended hardware resources such as GPUs, accelerators, FPGAs, or any other PCIe add-ons in a platform.

The SR-IOV network device plugin is an NPWG K8s device plugin that discovers and registers SR-IOV networking resources. This gives Kubernetes a global view of all available SR-IOV virtual functions (VFs) available in the cluster.

A workload can request for extended resources in their manifest. The Kubernetes Scheduler can then identify a node that has the required resources available and execute the new workload to that node. The device plugin is used to manage DDP-enabled virtual functions at runtime in a Kubernetes cluster.

The SR-IOV network device plugin is an NPWG open-source project; refer to *SR-IOV Network Device Plugin GitHub repository* (Table 2).

3.2.1 Architecture

Below is an overall view of a Kubernetes system with DDP-enabled SR-IOV virtual functions managed by Kubernetes Device Plugins and CNIs. The key components of the system are labeled and explained below.

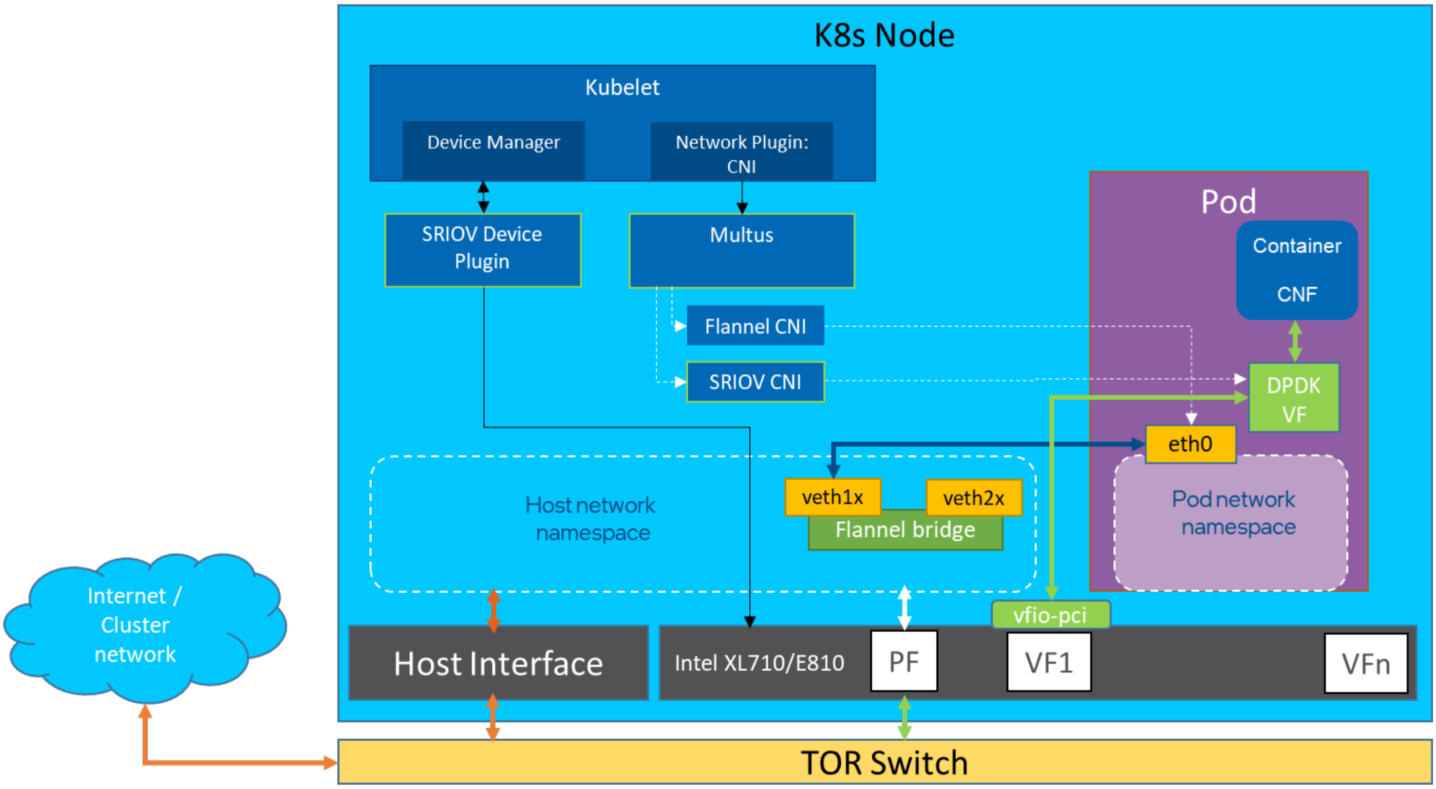

Figure 3. Architecture Diagram

- **Intel® Ethernet Network Adapters X710/E810 (DDP Enabled)** is an Intel® Ethernet Network Adapter 700/800 series with a DDP package applied. The DDP package is applied to a PF and all VFs related to this PF share the same DDP package.
- **SR-IOV Network Device Plugin** detects which DDP package the VFs are running on and advertises the VFs with associated DDP package in the cluster.
- **Multus** handles the creation of pods with multiple network interfaces. In the above example, Multus handles one default Flannel interface and one SR-IOV VF interface.
- **SR-IOV CNI** configures individual SR-IOV VFs to be used as interfaces in a pod.

The other components shown in the diagram above are standard parts of the K8s cluster, like the K8s API-Server, or common default add-ons, such as the CNI plugin and Flannel.

3.2.2 DDP with SR-IOV Network Device Plugin

The SR-IOV network device plugin can be used to identify currently running DDP packages, allowing it to filter virtual functions by their DDP package names. For further information on how to configure Intel NICs using this open-source NPWG component, refer to *SR-IOV Network Device Plugin – DDP documentation* in Table 2.

4 Summary

This guide demonstrates the provisioning and management of DDP-enabled virtual functions in cloud orchestrator Kubernetes. DDP eliminates the need for the CPU to perform certain kinds of work, such as load balancing or packet classification, on specific widely-used network protocols. This reduces latency by keeping network packets out of queues.

The automated workflow described in this document requires little intervention by cluster administrators to put the advanced performance capabilities of DDP devices to work. Users can modify and customize the associated documents and configurations from open-source repositories according to need and are encouraged to do so.

Following the outlined steps, cloud network functions can be provisioned quickly to take advantage of DDP technology. This allows high performance and improves utilization, not just on the network card, but on the overall hardware platform.

0221/DN/WIPRO/PDF 621832-002US