White Paper

Cloud Networking applications
Cloud Applications, Security, and Wireless Communications

Cloud Native Data Plane

Data Plane development for 5G Mobility Edge made easy.

Authors
Edwin Verplanke
Senior Principal Engineer
Serge Maskalik
VP Engineering

Overview

It is no secret that mainstream Data Center applications are migrating to cloud. Similarly in the Security and Wireless Communications industry, this trend is emerging as well. Security applications such as Firewall, DDoS prevention, WAF, etc., have a natural affinity since the business logic has already moved in this direction and requires protection and portability. Over the last two decades, all major security and networking appliances become virtualized on-premise and in the public cloud, untethered and agnostic from hardware they run on. Also, as telecom and enterprise customers adopted the hybrid cloud strategy, the need for platform portability gained more momentum as customers may have a different cloud platform on-premise and in the public cloud while using the same networking functions.

Fast forward to today, where we build the reality of edge cloud with low latency applications of AR/VR, video data acquisition with local inferencing engines, robotics controllers in manufacturing, clinical communications and imaging for healthcare, etc. Cloud vendors transformed from regional clouds to near-edge clouds with lots of sites near the enterprise locations. The making of the next hybrid 5G cloud is the intersection of the following trends:

- Multi-cloud – in order to get near-edge proximity, it will take a village (AWS Local Zones, Azure Edge, VMware on-premise, Google Edge, Vapor I/O, etc.)
- Cloud Native principle adoption by all network function vendors
- 5G technology integration with Software-Defined Networking domains as 5G enters the hands of enterprise IT operation
- Cloud Native Data Plane (CNDP) can be used to create or migrate Data Center applications to the cloud infrastructure.

An example of such a solution is the Highway9 Networks Virtual Mobility Cloud, allowing cloud-consumption of 5G services for the Enterprise. The Virtual Mobility Cloud (VMC) workflows simplify and automate the provisioning of private CBRS wireless networks with templates for common use cases to minimize user input. The VMC-dedicated instances are rapidly deployed for each enterprise tenant to provide a secure cloud-first enterprise deployment model. The Virtual Mobility Edge provides the control and user plane for private CBRS networks – flexibly deployed on-premise or on a proximate Edge or Public Cloud. Simple to operate, the indoor and outdoor radios are activated by the Highway9 Networks cloud service and integrated into corporate AAA services like Network Access Control (NAC), Active Directory and Mobile Device Management.

Highway9 Networks Virtual Mobility Edge

The Mobility Edge houses the 5G controller with fastpath user plane functions like 5G UPF and features like MEC local breakout which applies to all NG RAN instances in a given site. This system also manages the subscriber authentication and policies by integrating with IT systems of record such as Active Directory to map the hardware root of trust in SIM to enterprise users, Network Access Control policies for ACLs and VXLAN/VLAN routing and end-user devices grouping from Mobile Device Management systems. To satisfy the enterprise multi-cloud requirements, the Mobility Edge has to integrate with all common flavors of private and public cloud: VMware with VMXNET3, AWS with Nitro, baremetal for embedded appliances and Google cloud (GCP) as well.

In order to build an open multi-cloud compatible solution without any technology "lock-in," adoption of CNDP using AF_XDP is highly attractive. Previously, Highway9 Networks Mobility Edge solution had to be cognizant of all SDN choices. It is further exacerbated with different ways to manage SRIOV access in clouds. CNDP toolkit bring a level of portability and interoperability required to provide multi-cloud high speed, low latency networking solutions.

Cloud Native principles, furthermore, make the portability possible, but also come with its own set of challenges for data place applications. Can you maintain combability with all populate Kubernetes flavors, with variance of container storage and networking (CNI/CSI) subsystems? Can the security posture be maintained without requiring privileged container access? How can

> The next generation of enterprise campus SDNs are being built on multi-cloud, 5G technologies and cloud-native design principles – CNDP enables the ultimate portability and feature velocity to bring this to market.

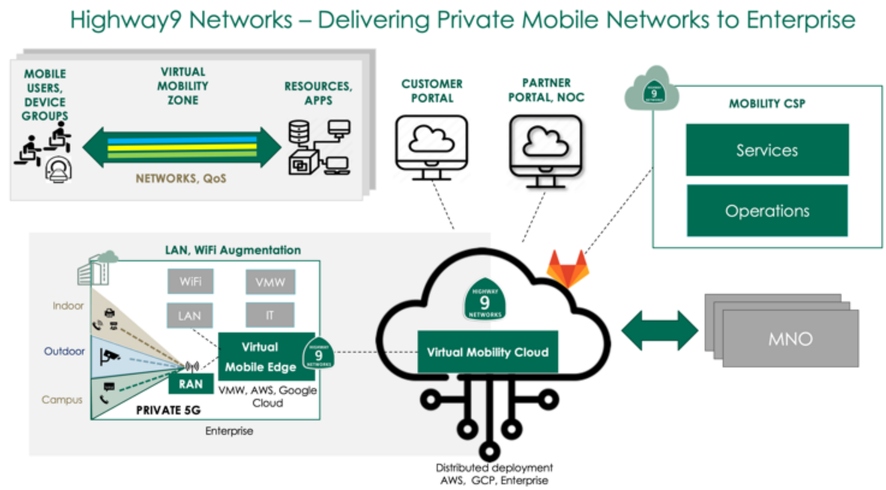


Figure 1: Highway9 5G Mobility Edge

common deployment be achieved with tools like Helm across all K8s cloud platforms? Is there a standard way to do logging, metrics, event handling and other transparency requirements for telemetry? How can upgrades be managed in a Cloud Native fashion for data plane applications while meeting the low downtime and resilience goals? Previously, deploying cloud native data plane solutions meant sacrificing portability and compatibility across platforms – but now CNDP offers an elegant solution where the developer can easily balance performance & compatibility. The multi-cloud abstraction via AF_XDP, the Kubernetes orchestration and the availability of CNDP deployment operator with CNDP device plugin and the CNI largely solves the multi-cloud problem. The native integration into Kubernetes telemetry tool chain via Prometheus sidecar that is included.

Migration Made Easy

Like other high speed networking systems, Highway9 Networks leverages VPP to achieve low latency and high speed for its 5G Mobility Edge instances. Adopting CNDP is made easy because it is a gradual process without knife edge cutovers required. CNDP includes a VPP plugin that allows for existing VPP apps to begin to leverage CNDP benefits without a rewrite. A good strategy is to sediment functionality natively into CNDP as the business objectives are met without a costly re-platforming exercise. Similarly, lots of functionality from DDPK is available. One other aspect Highway9 Networks found attractive about CNDP was the redefined relationship with Linux Networking. Previously, Linux networking was effectively bypassed with SRIOV/DPDK – but now it is embraced. This enables additional installation, orchestration and automation benefits that were previously lost. The visibility and accounting for networking pseudo-devices and new capabilities in testing is possible.

Wireless applications

For Wireless types of applications, it is less obvious, however the 5G control plane has an affinity toward a cloud native deployment since it has adopted HTTP/2 for its application layer. However, the cloud migration for communications centric applications does come without challenges. For example, a communications application often maintains elements of (legacy) monolithic code that don't lend itself to be refactored in micro services. Code migrating from an appliance type of solution where it is the expectation that the access to Layer 2 and/or Layer 3 network protocol access may be hampered since public cloud may not allow access. These are just a few examples, and clearly there are many more, such as the data plane migration element. The latter is reviewed in the next section.

Whether the communications application addresses a wireless or security requirement, there is one thing they have in common: each needs to move large volumes of packets or data frames between interfaces, commonly referred to as the data plane. We can break down the data plane migration challenges into the following categories:

- Data Plane or User Plane functionality often the high-performance data movement requirement and are typically designed following a run to completion model and do not lend themselves to be refactored in micro services.
- The Linux Networking infrastructure is comprehensive but small packet movement performance is typically not sufficient for network infrastructure applications.
- To guarantee the performance requirements dedicated resource allocation is typically required. For example, huge page memory allocation, limit scheduling overhead by utilizing core pinning and dedicated network resource assignment is required.
- Because of the above two challenges, the orchestration complexity has increased. An operator would have to find the hardware spec that aligns with the requirements above to maintain the performance requirements.

CNDP

CNDP is a collection of user space libraries for accelerating packet processing for cloud applications. It aims to provide better performance than that of standard network socket interfaces using an I/O layer primarily built on AF_XDP, an interface that delivers packets directly to user space, bypassing the kernel networking stack. CNDP provides a custom TCP/UDP stack, libraries for RIB, FIB, ACL, Hash, etc. It also provides JSON parsing and libraries to expose metrics and telemetry with examples to deploy services on Kubernetes.

Addressing challenges

In Figure 2: CNDP Overview, it shows the high-level overview of CNDP and where it sits in the system. A CNDP application sits in user space and interacts with the kernel to send/receive packets via the AF_XDP sockets interface. Using AF_XDP sockets for packet I/O allows the application to run without having to know or require specific hardware, which makes the application more portable. The application still has access to other system interfaces, if needed.

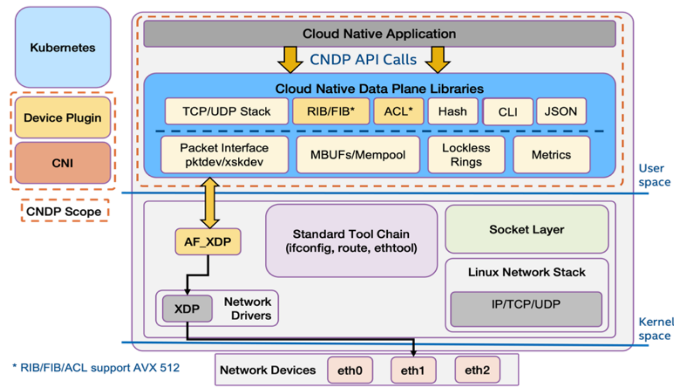


Figure 2: CNDP Overview

Cloud integration

The Kubernetes integration via the device plugin and CNI helps the developer install the application without having to know about most of the details in the system to run the application. Having the orchestration available and accessed by the Kubernetes standard interface allows the developers to understand the application's performance and configuration.

Performance

The CNDP set of libraries provide a lightweight integration of CNDP components into the application to enhance the performance and/or deployment. The libraries provide performance and features to give the developers a solid foundation to build applications while achieving the best performance. Some of these performance features utilize Intel architecture enhancements i.e., Intel® AVX-512 instructions and performance optimizations. Highway9 Networks learned over time to get the best performance from applications running on Intel architecture. Because the best-known methods (BKMs) are put into CNDP libraries, the developer can focus on the business aspects of the application, which provide the features and solutions cloud applications require.

Cloud orchestration

Orchestration of cloud applications is complex and is critical to having a good application, as it needs to integrate with the management systems to make deployment easy and clean.

CNDP provides new integration plugins and tools to allow systems like Kubernetes to deploy the application in the cloud. Making it easier for the developers is one of the primary goals of CNDP.

With these tools and plugins, Kubernetes can configure the container or Pod to better deploy the application into the system for performance and functionality.

Faster Time to Market

In order to achieve fast time to market and maintain feature velocity for Highway9 Mobility Edge, CNDP has lots of features and primitives available, beneficial to the edge applications like UPF and enterprise routing.

CNDP Toolkit

Having a toolkit, shown in Figure 3: CNDP Libraries, gives an optimized RIB/FIB primitives for fast forwarding lookups, precision timers, graph library, ACL support and data-inflight encryption engines. This allows Highway9 Networks to focus on the application and not the scaffolding.

Performance monitoring cloud native applications are critical to the success of these types of applications. CNDP provides and maintains several performance metrics and allows exporting these metrics to external applications i.e., Prometheus and other cloud native monitoring tools. Metrics for the application is critical to understand if the application is performing well and what changes could be done to improve the performance. Adding more metrics and performance statistics should be enabled within the application process, plus being able to collect information about the system performance is required.

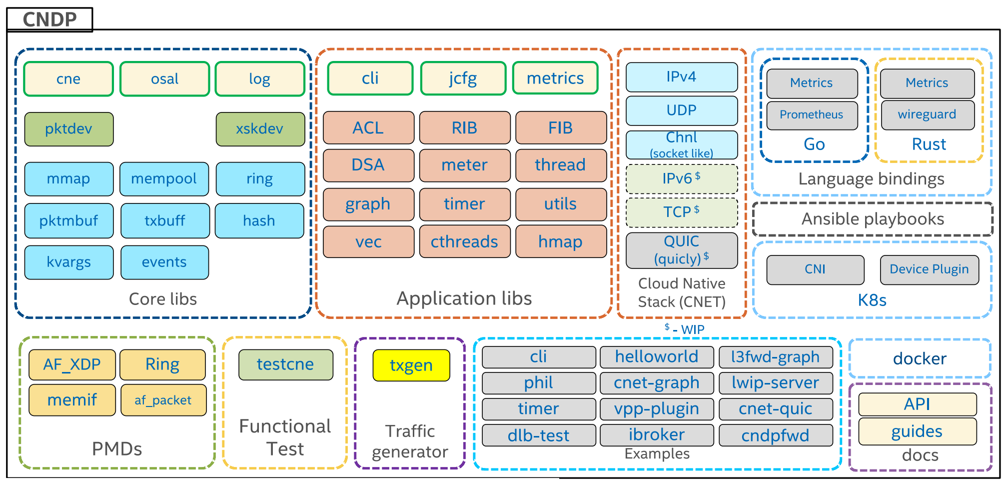


Figure 3: CNDP Libraries

Network Stack

Many applications require access to a network stack as the application may have been using the kernel network stack. When using AF_XDP packets bypass the kernel network stack, which means some applications need a user space stack to function properly. CNDP provides a user space network stack called CNET (Cloud Networking) and currently provides an IPv4/UDP protocols. Work is in progress for TCP/IPv6 as new features. The CNET stack is created using graph nodes and allows the developer to create new nodes to process packets as needed by the application (see Figure 4: CNET stack high level view).

The CNET stack contains a channel interface/API is similar to standard sockets, but it uses a multi-packet design for better performance and zero copy of packet data unlike the standard socket interface.

Language Bindings

The CNDP libraries also provide a set of language bindings to integrate common languages used in cloud native applications i.e., Go and Rust. The language bindings are a work in progress to allow migrating the application using languages.

Vector Packet Processor (VPP)[1]

The Vector Packet Processor (FD.io VPP)[1] is a user space network stack. Packets are passed through a directed graph, where each node in the graph processes multiple packets at a time. The graph is extensible using shared libraries, or "plugins," that are loaded when the VPP process starts. Any plugin can modify the graph by introducing new nodes or rearranging existing nodes.

The CNDP plugin uses the CNDP API to interact with AF_XDP sockets. When requested, the CNDP plugin opens an AF_XDP socket on each requested queue using the CNDP API and inserts an input node into the VPP packet processing graph. VPP calls the CNDP input node to receive packets, which are passed through the graph for processing like any other packet.

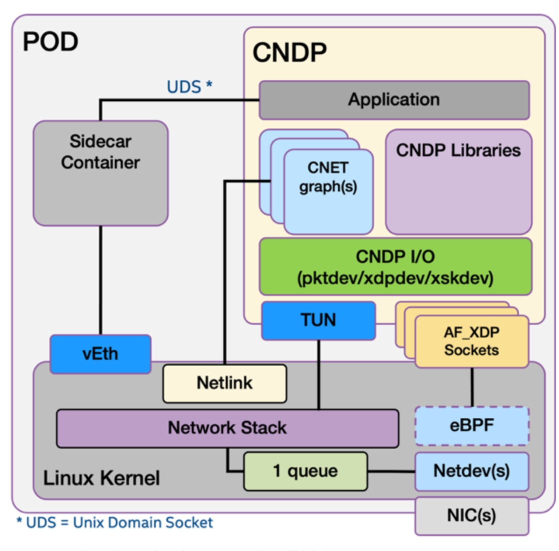


Figure 4: CNET stack high level view

[1] FD.io VPP https://fd.io/gettingstarted/technology

The plugin uses a feature of CNDP called Transparent Buffers that makes it possible to pass the VPP packet buffer addresses directly to the kernel through the AF_XDP socket, avoiding any packet copies between hardware and the CNDP plugin.

Enabling Data Plane CI/CD

As the Highway9 Networks Mobility Edge adopted CNDP, further simplifications happened around the continuous development and testing became possible. The fully cloud-native deployment of the data plane application allowed for standard pipelines to be used which greatly simplify testing and deployment.

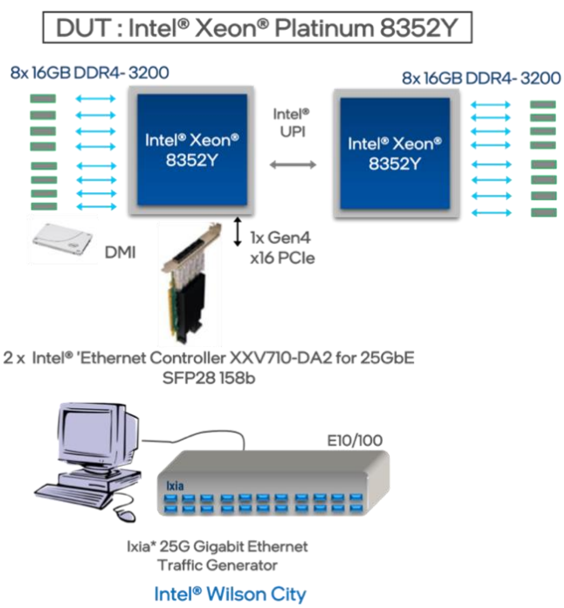


Figure 5: Test Layout

DUT (Platinum 8352Y)

- Processor: Intel® Xeon® Platinum 8352Y CPU @ 2.20GHz
- Platform: Intel® Wilson City
- Memory: DDR4 3200 MT/s RDIMMs 8x16GB per socket (total 128 GB), 8 Channels/socket
- NIC: 2x Intel® Ethernet Controller XXV710

Boot Configuration:

- BOOT_IMAGE=/vmlinuz-5.11.0-49-generic root=/dev/mapper/ubuntu--vg-ubuntu--lv ro

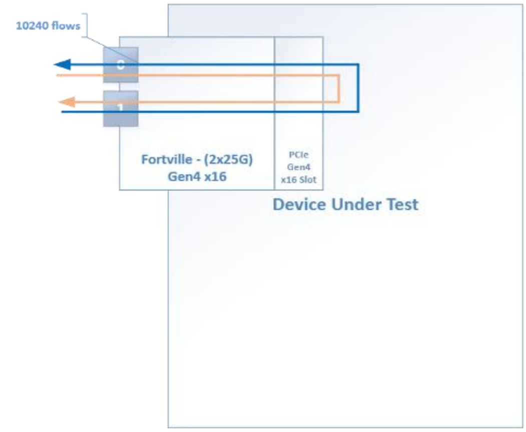


Figure 6: Test Setup

IXIA* Traffic Parameters:

- IxNetwork* : 9.00
- Test: RFC2544 quick test
- Acceptable Frame Loss: 0.1%, 0.01%, 0.001%, 0.0001%
- UDP traffic
- Resolution: 0.1
- Traffic Duration: 20 Seconds
- Flows per port for CNDP/Ip tables: 10K

CNDP 22.01.0 2x25G ports (2C/2T) 1 Queue/Core

Queue/Interrupt Cores:

- CPU 6 (Socket 0, Core 6) polls port 0
- CPU 7 (Socket 0, Core 7) polls port 1

Application Cores:

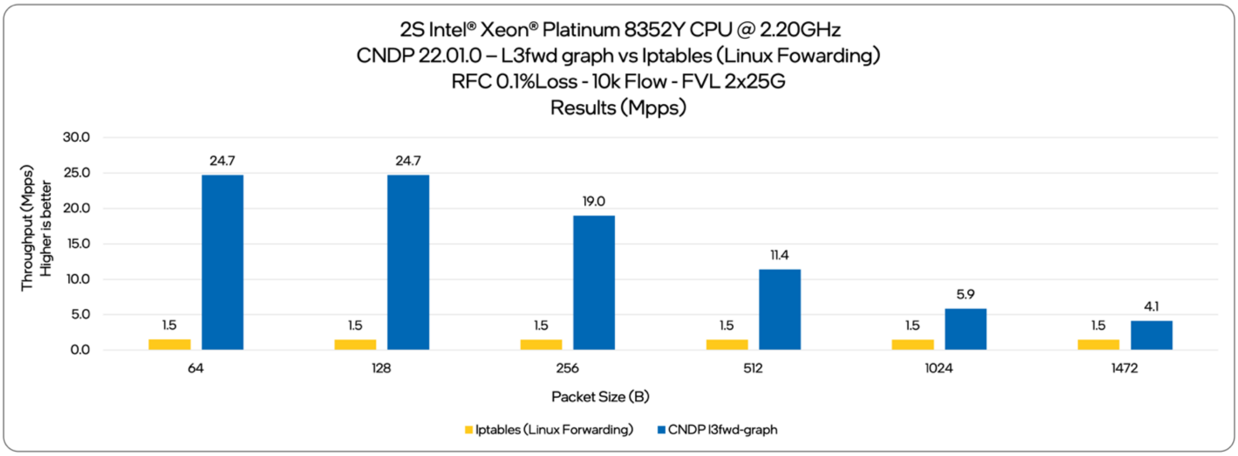
- CPU 8 (Socket 0, Core 8)
- CPU 9 (Socket 0, Core 9)

Iptables

2x25G ports(2C/2T)

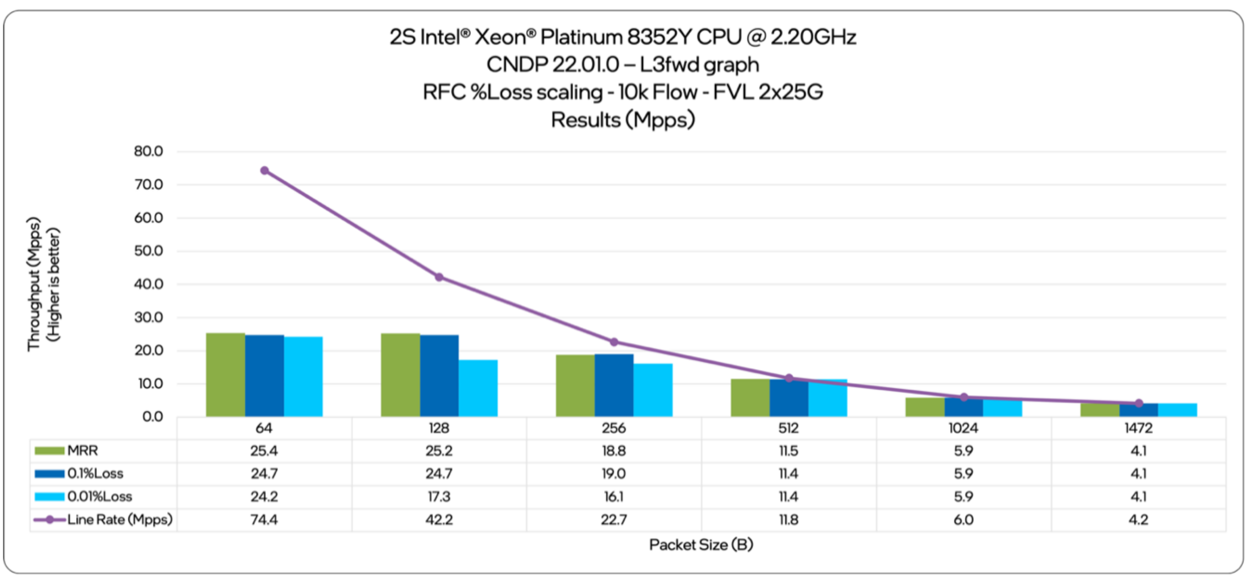
- CPU 6 (Socket 0, Core 6) polls port 0
- CPU 7 (Socket 0, Core 7) polls port 1

Intel® Xeon® Platinum 8352Y CPU @ 2.20GHz
Iptables vs CNDP l3fwd-graph – FVL 25G – RFC 0.1%Loss
Raw Results(Mpps) – 10 K Flows



- 16x Performance Gain observed between Linux forwarding and CNDP l3fwd-graph

Figure 7: Relative Performance



	64	128	256	512	1024	1472
MRR	25.4	25.2	18.8	11.5	5.9	4.1
0.1%Loss	24.7	24.7	19.0	11.4	5.9	4.1
0.01%Loss	24.2	17.3	16.1	11.4	5.9	4.1
Line Rate (Mpps)	74.4	42.2	22.7	11.8	6.0	4.2

Figure 8: Packets per second

Relative Performance

The Figure 7: Relative Performance shows the packets per second performance compared to a standard Linux socket-based application and the performance increase achieved by using CNDP. Using a 64-byte frame we show CNDP/AF_XDP can provide almost a 24x improvement in packet performance compared to the forwarding packets using Linux iptables.

In Summary

CNDP in conjunction with customers like Highway9's 5G Mobility Edge are building Cloud Native applications giving the developer and applications the following:

CNDP Consumers:

- **Cloud Network Function (CNF) and Cloud Application developers**: Those who create applications based on CNDP. CNDP hides the low-level I/O, allowing the developer to focus on their application.

- **CNF and Cloud Application consumers**: Those who consume the applications developed by the CNF developer. CNDP showcases deployment models for their applications using Kubernetes.

CNDP follows a set of principles:

- **Functionality**: Provide a framework for cloud native developers that offers full control of their application.

- **Usability**: Simplify cloud native application development to enable the developer to create applications by providing APIs that abstract the complexities of the underlying system while still taking advantage of acceleration features when available.
- **Interoperability**: The CNDP framework is built primarily on top of AF_XDP. Other interfaces, such as memif, are also supported, however building on AF_XDP ensures it is possible to move an application across environments wherever AF_XDP is supported.
- **Portability/stability**: CNDP provides ABI stability and a common API to access network interfaces.
- **Performance**: Take advantage of platform technologies to accelerate packet processing or fall-back to software when acceleration is unavailable.
- **Observability**: Provide observability into the performance and operation of the application.
- **Security**: Security for deployment in a cloud environment is critical.

0522/KW/351569-001EN ♻ Please Recycle