White Paper

# Enhancing Performance and Power Efficiency of a Cloud-Native Broadband Network Gateway (BNG) Using 4th Gen Intel® Xeon® Scalable Processors


## Authors

Jakub Pažej
Intel Platform Applications Engineer

Andrew Duignan
Intel Platform Solutions Architect

Jasvinder Singh
Intel Sr Software Engineer


## Table of Contents



## Executive Summary


The ever-growing consumer demand for more public network bandwidth and services at lower cost has driven service provider networks to their economic limit. In addition, Communications Service Providers (CoSPs) need to support multiple access technology types (xDSL, PON, FWA, and DOCSIS), while making better use of existing fiber networks and increasing service delivery performance—all against a backdrop of declining revenues. With customer traffic estimated to grow at a 20 percent rate peak usage this year, increasing to 95 percent by 2027[1] (and more in certain geographical regions, for example, North America), future networking solutions must address tomorrow's data growth challenges in a cost-effective and scalable manner, delivering needed performance and reducing costs.

Pushing the boundary of performance requires next-generation technology that can be easily and holistically deployed. Using the following design principles for advanced networking solutions based on Intel® processors and Network Functions Virtualization (NFV), Intel shows that service networks can achieve next-level performance, power, efficiency, and network automation.

- Optimized server configuration
- Data/user plane "run to completion" model
- Control plane and data/user plane packet distribution
- Independent scaling of the control and data/user planes
- Hierarchical quality of service (HQoS) considerations
- Cloud-native deployment and data/user plane power management technology on 4th Generation Intel® Xeon® Scalable processors

This paper describes such an effort and proposes a virtual Broadband Network Gateway (vBNG) architecture for building network infrastructure and network functions that take advantage of the underlying infrastructure and addresses the challenge of the data growth. To complement the shift of the broadband data/user plane into a virtual ecosystem, this paper also presents a deployment architecture using Kubernetes (K8s), the container orchestration engine.

Using this approach, Intel has demonstrated 710 Gbps[2] of routing RFC2544 (zero packet loss) performance for a vBNG running on a single, dual-socket server with an Intel® Xeon® Gold 6438 processor (32 cores/64 threads). Furthermore, the effort shows it is possible to achieve server power savings of up to 14 percent over a typical 24-hour traffic period, with peak server power savings of up to 25 percent on the same Intel platform.[2]

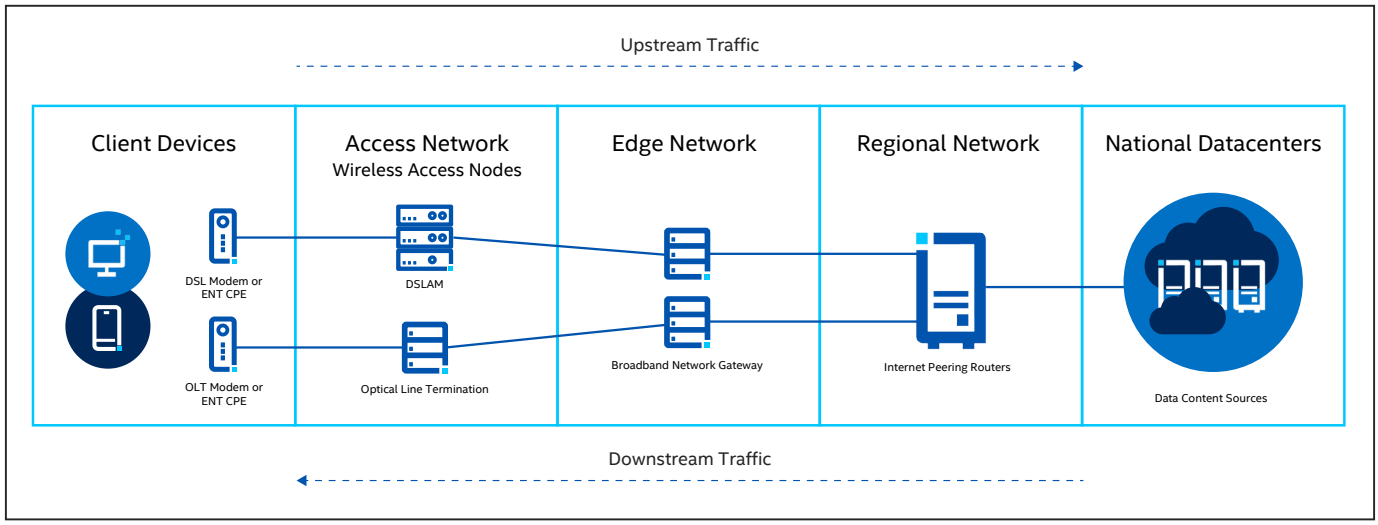


**Figure 1.** Example of a network connecting clients to a data center through a BNG

## The Broadband Network Gateway

The Broadband Network Gateway (BNG) is the network edge aggregation node subscribers use to access the Internet Service Provider (ISP) network (Figure 1). The BNG is traditionally a large, ASIC-based, fixed-function multi-line card appliance, usually located in a central office or metro Point of Presence (PoP). Through the BNG, subscribers connect to the ISP network to download Internet-originating traffic and ISP services (e.g., web, voice, data, and video).

A vBNG is a virtualized software instantiation of all the functionality of a BNG. Virtualization is possible due to the advanced capabilities of modern microprocessors and software and common-off-the-shelf (COTS) servers available today.

## Intel Reference Application Architecture

Each generation of Intel technologies (e.g., CPU, NIC, SSD, and accelerators) brings new opportunities to improve performance, power efficiency, and quality of experience (QoE) for users. Illustrating how to take advantage of these technologies, Intel designs reference architectures. Reference architectures draw on the capabilities of and optimizations for Intel processors and other silicon, devices, and servers to deliver optimum application performance and power efficiency.

### vBNG Reference Architecture

Figure 2 illustrates a reference architecture for the representative stages of vBNG control and data/user plane functions.

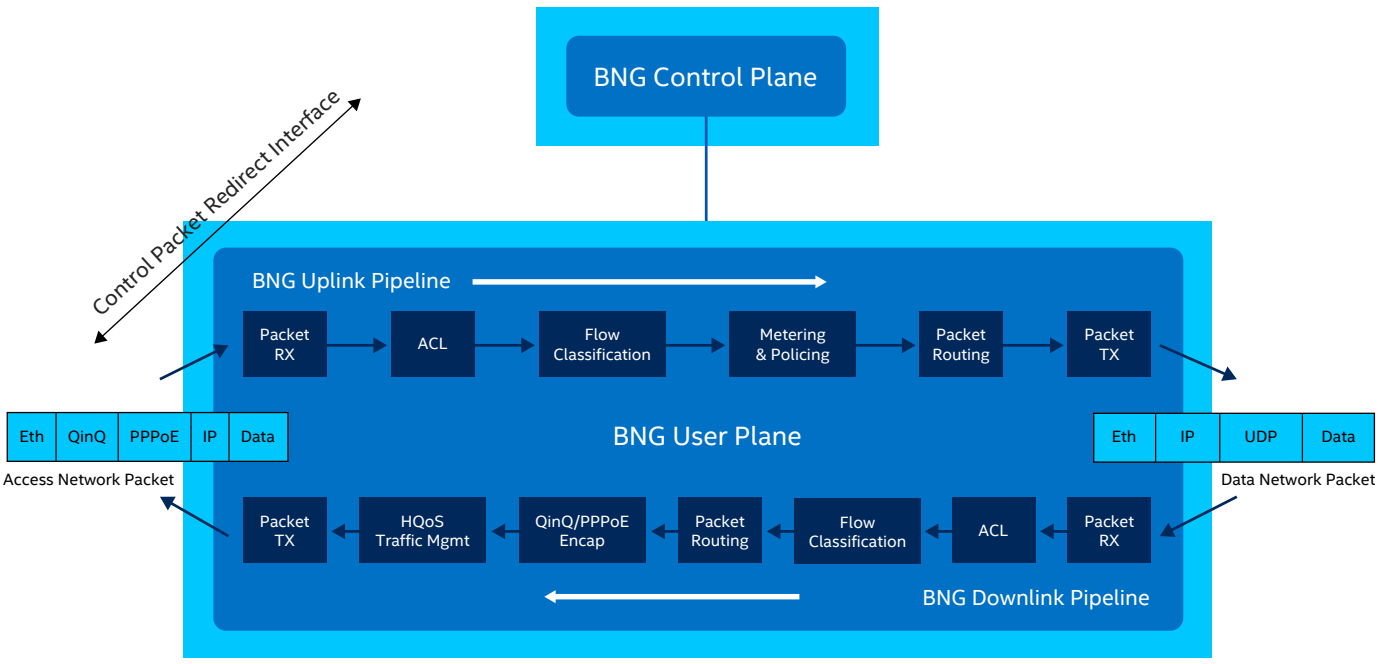


**Figure 2.** Virtual Broadband Network Gateway (vBNG) control and data/user plane blocks

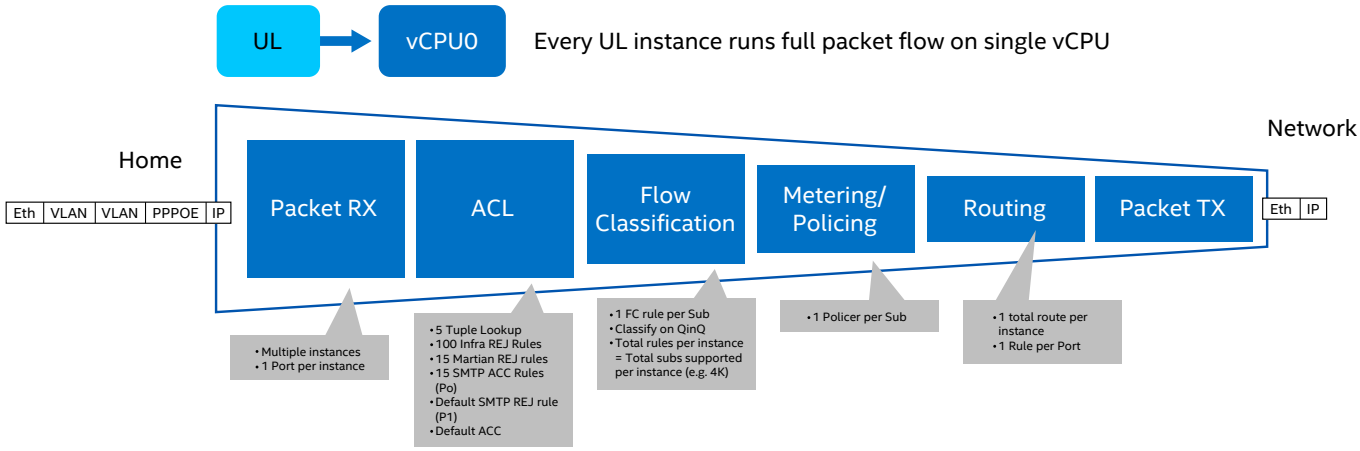


**Figure 3.** Upstream processing stages

**Control plane:** This plane is responsible for subscriber authentication and IP address management, including monthly usage service access and data/user plane configuration, based on subscriber profiles.

**Upstream data/user plane:** This pipeline manages the flow of traffic from users' home routers to the ISP network. The average packet size for upstream traffic is generally smaller than downstream packets, and the volume of upstream traffic is normally five to eight times less than downstream traffic. In recent years, data and content creation have reduced the gap between upstream and downstream bandwidth usage. Yet, while applications like Instagram, TikTok, and Snapchat have seen larger swathes of data being pushed onto the ISP network by users, broadband users overwhelmingly are still net consumers (downstream) of data.

**Downstream data/user plane:** This pipeline is the flow of traffic and data from the Internet and ISP network to the end user. It manages and schedules traffic to users attached to the BNG. The downstream function optimizes bandwidth and resource usage to maximize users QoE, based on user tariff class and traffic priorities.

The goal of the ISP is to ensure all subscribers are receiving services to the highest standard, while maximizing the utility of the network infrastructure. 22 Global IP video traffic is forecast to grow four-fold from 2023 to 2027, a CAGR of 20 percent.[3] This trend will drive up the average packet size of the downstream link.

## Upstream Processing Stages

The Intel vBNG reference architecture implements the upstream processing stages shown in Figure 3 and described in the following:

- **Packet Rx (Receive):** The Data Plane Development Kit (DPDK) poll mode driver (PMD) is used to receive bursts of frames from the network interface controller (NIC) port and send them directly into an uplink thread to begin vBNG packet processing, which is described in the next stages.
- **Access Control Lists:** The DPDK Access Control List (ACL) library is used to apply an ordered list of filters (e.g., masks, ranges, etc.) to the frame. These comprise of permit and deny filters. All filters are evaluated per packet.
- **Flow Classification:** The DPDK Flow Classification Library is used to identify the session and classify the packet based on selected fields (e.g. 5-tuple).
- **Metering Policing:** The DPDK Traffic Metering and Policing API is used to apply a two-rate, three-color marking and policing scheme to the traffic.
- **Routing:** Access network encapsulations are stripped from data plane packets, and the packets are routed to the correct data network interface for transmission. Any data network encapsulations, such as MPLS, are applied either here or in the packet Tx block.
- **Packet Tx (Transmit):** The DPDK PMD is used to send bursts of frames to the NIC port.

## Downstream Processing Stages

The Intel vBNG reference architecture implements the downstream processing stages shown in Figure 4 and described in the following:

- **Packet Rx:** The DPDK PMD is used to receive frames from the NIC port and sends them directly into a downlink thread to begin vBNG packet processing, described in the next stages.
- **Access Control Lists (ACL):** The DPDK ACL library is used to apply an ordered list of filters (e.g., masks, ranges, etc.) to the frame. This stage blocks reverse path forwarding.
- **Traffic Management:** Each packet runs through a hierarchical QoS (HQoS) block to ensure high priority packets are prioritized when transmitting packets to the access network. The block supports scalable five-level hierarchical construction (port, subport, pipe, traffic class, and queues) of traffic shapers and schedulers to guarantee the bandwidth for different services used by subscribers. Each pipe is assigned to a single subscriber.

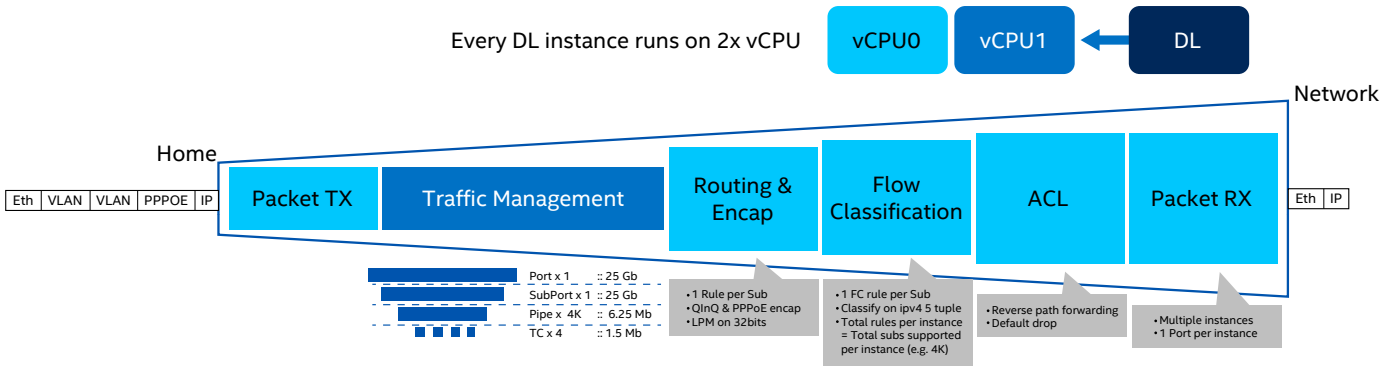


**Figure 4.** Downstream processing stage examples

- **Routing:** Access network encapsulations are stripped from data plane packets, and the packets are routed to the correct data network interface for transmission. Any access network encapsulations, such as VLAN, PPPoE, etc., are applied either here or in the packet Tx block.
- **Packet Tx:** Using a DPDK PMD, bursts of frames are transmitted to the NIC port.

## Architectural Considerations

To effectively deploy a BNG workload on a general-purpose server, the following architectural and implementation aspects should be considered.

### Using a Run to Completion Model for Data/User Plane

One of the key considerations when designing a software-based BNG (vBNG) is ensuring performance scalability. The vBNG should be assigned the minimal number of resources needed to support the current number of active subscribers at any time of the day. This means the vBNG must be able to scale both up and down based on the current traffic load presented.

The Intel vBNG reference architecture uses a run to completion model to process the uplink and downlink pipelines. As a result, all data/user plane packet processing functions per direction (uplink and downlink) executed on a packet are run on the same core. Creating this affinity keeps packets from moving between cores, thereby minimizing cache misses and overall latency. Scaling beyond a single core is done by creating a new K8 BNG instance that runs on a different core. The NIC is programmed (using customer headers supported by the Comms DDP package) on the fly to direct specific subscribers to each new individual BNG instance (more on this further on in the paper).

The system orchestrator can scale capacity up or down by increasing or decreasing the number of vBNG K8s pods/instances (3 vCPUs per pod/instance with K8s), enabling linear scalability across a given server.

### Separating Uplink and Downlink Processing

CPU resource usage by the vBNG uplink and downlink pipelines are not symmetric, since the downlink normally requires more cycles per packet due to inherently larger packet sizes. To effectively schedule a BNG, the Intel vBNG reference architecture splits the uplink and downlink into two separate containers that can be instantiated and then scheduled separately. This separation provides greater flexibility in scheduling and CPU resource usage. For example, a downlink pipeline can be assigned a full physical core (two sibling hyper-threads/virtual cores), while an uplink pipeline might only require half a physical core (one hyper-thread/virtual core).

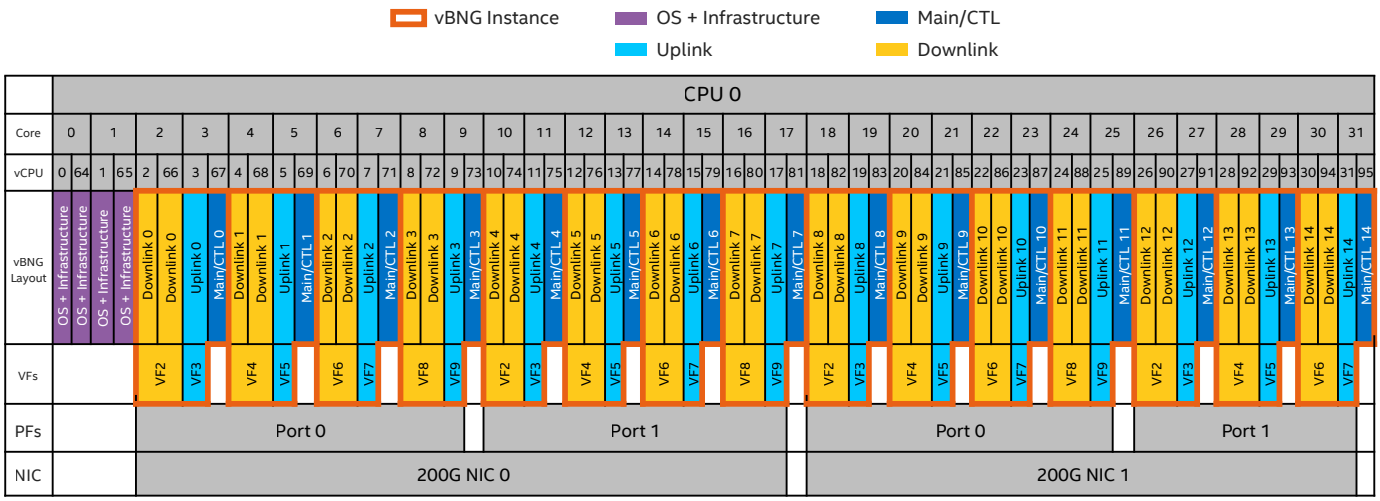


**Figure 5.** CPU core allocation example for 15 vBNG instances running per socket

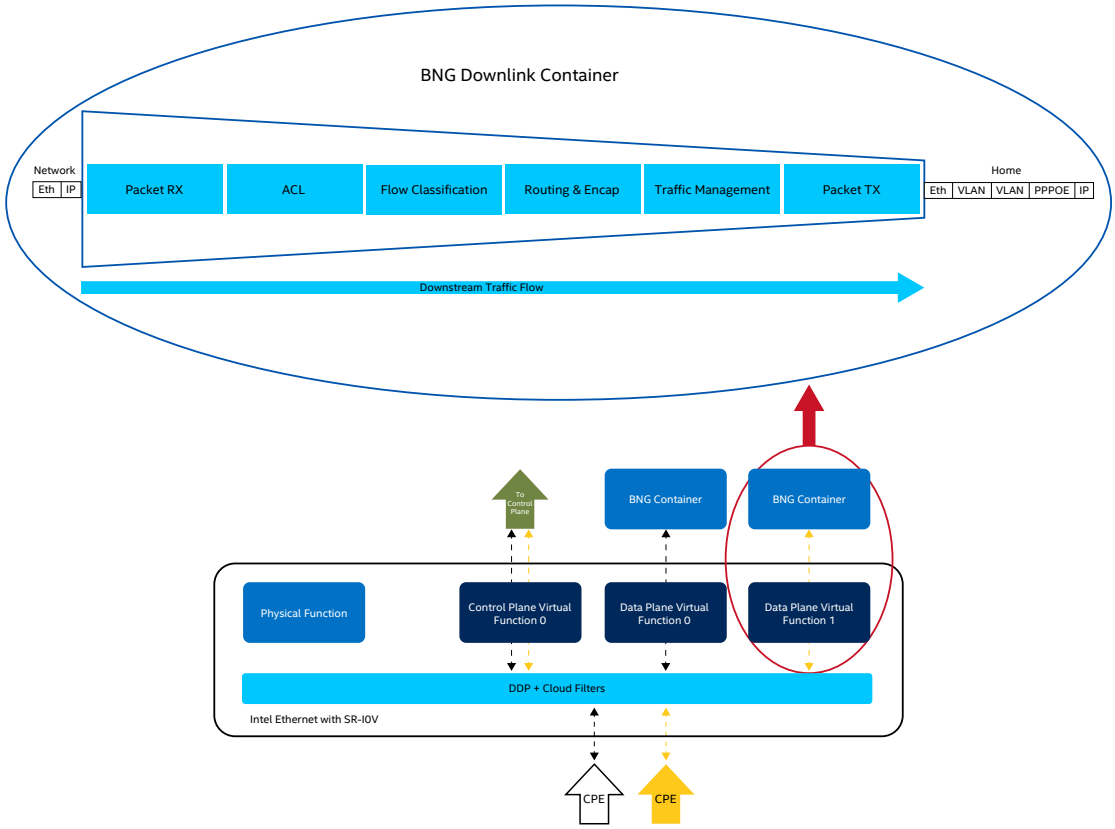


**Figure 6.** BNG downlink using a single SR-IOV VF per container for both Rx and Tx

Figure 5 shows how the CPU resources of a dual-socket server could be partitioned when running fifteen vBNG instances per socket in a Docker-only configuration (i.e., not using K8s). Note that using a 32-core Intel Xeon Gold 6438 processor, we deploy up to fifteen instances per CPU socket, reserving two CPU cores for the OS and infrastructure and leaving 30 CPU cores for the vBNG data/user plane instances.

Each pipeline can report telemetry individually. The telemetry database can be used to maintain all relevant usage statistics. It is also worth noting that while uplink and downlink pipelines are deployed as separate VPP data/user plane applications and containers, this paper puts forward the architecture of coupling them together using Kubernetes and its pod deployment API.

## Assigning a Single I/O Connection per Pipeline

The Intel vBNG reference architecture should be run on a BNG data/user plane server connected to a basic leaf switch that can route both access and data network traffic. A BNG that uses separate, dedicated physical ports for access and data network port connections is likely to underutilize the available I/O bandwidth of the uplink ports. Instead, sharing physical ports on a NIC between upstream and downstream traffic allows full utilization of the I/O bandwidth.

As a vBNG instance is split into two separate pipeline applications, each pipeline only handles traffic for a single direction. All traffic is routed to and from the server through the simple L2 switch, such that each pipeline does not require dedicated access and data network ports. The server effectively needs just a single I/O connection on which it receives traffic from the switch and returns processed traffic to the switch for forwarding as shown in Figure 6.

The routing of subscriber traffic to a vBNG instance is done via a dedicated Single Root Input/Output Virtualization (SR-IOV) connection that can send arriving packets to the vBNG in accordance with its SR-IOV switch (with DDP). SR-IOV allows a single physical NIC port to be split and shared among multiple pipeline instances, each with its own I/O port i.e., a Virtual Function (VF). SR-IOV also provides flexibility in the use of physical NICs, such as dedicating a physical NIC to downlink traffic only or sharing a NIC between uplink and downlink traffic.

## Distributing Control and Data/User Plane Packets via Network Interface Card (NIC)

### Device Config Function

The Intel® Ethernet Network Adapter E810 is a NIC that supports flow distribution using the Device Config Function (DCF) technology. DCF sets Flow Rules through a trusted VF, allowing the user to keep the SR-IOV Physical Function bound to the Linux driver for management and metric collection as seen below in Figure 8.

When distributing flows in the Intel Ethernet Network Adapter E810, it must be noted that for distributing flows amongst VFs, the SR-IOV eSwitch is used; for distributing flows amongst Queues, Flow Director (FDIR) or Receive Side Scaling (RSS) is used. In the vBNG deployment, flows are distributed using VFs, thus SR-IOV eSwitch is used. Information on RSS and FDIR can be found in other Intel white papers.

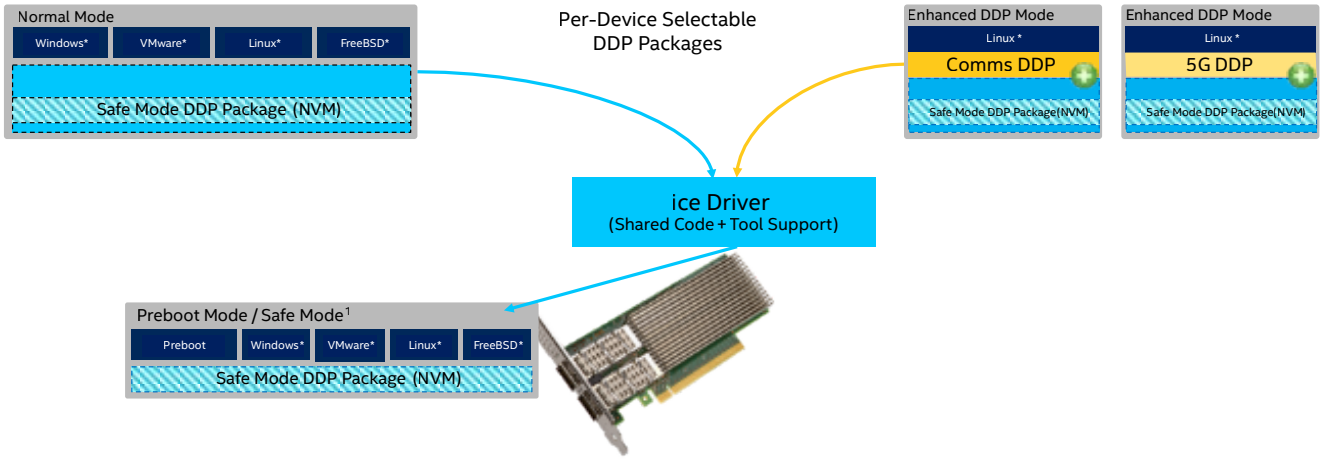


**Figure 7.** DDP packages for Intel® Network Ethernet Adapter E810 NIC

### Dynamic Device Personalization (DDP)

Alongside DCF is Dynamic Device Personalization (DDP), as shown in Figure 7, which allows the SR-IOV eSwitch to filter on more packet header types than the default amount without reloading the Ethernet Controller NVM image. For the BNG application deployment, the Telecommunication (Comms) Dynamic Device Personalization (DDP) Package is used. Once added, this package allows the Ethernet Controller to steer traffic based on PPPoE header fields to the control plane offload VF. The DDP PPPoE profile enables the NIC to route packets to specific VFs and queues (Figure 8) based on the unique PPPoE header fields (described more in the next section).

## Control and Data/User Plane Packet Distribution

For control plane traffic, like PPPoE session setup or PPPoE link control packets, the BNG data/user plane must identify and forward these packets to the control plane for processing. In a traditional BNG, the control plane and the data/user plane are located in the same place, and a local software queue is used to move packets between them. With the advent of Control and User Plane Separation (CUPS), the control plane and the corresponding data/user plane are most likely located in different physical locations in the network. In this case, the BNG data/user plane needs to pass control packets to the control plane by generating a physical link to forward them. Intel Ethernet Network Adapter E810 is able to identify these control packets using the Comms DDP package and forward them to separate VFs and queues (I/O), relieving the data/user plane of this task. Figure 8 shows a higher level view of how this works with the vBNG application deployment.

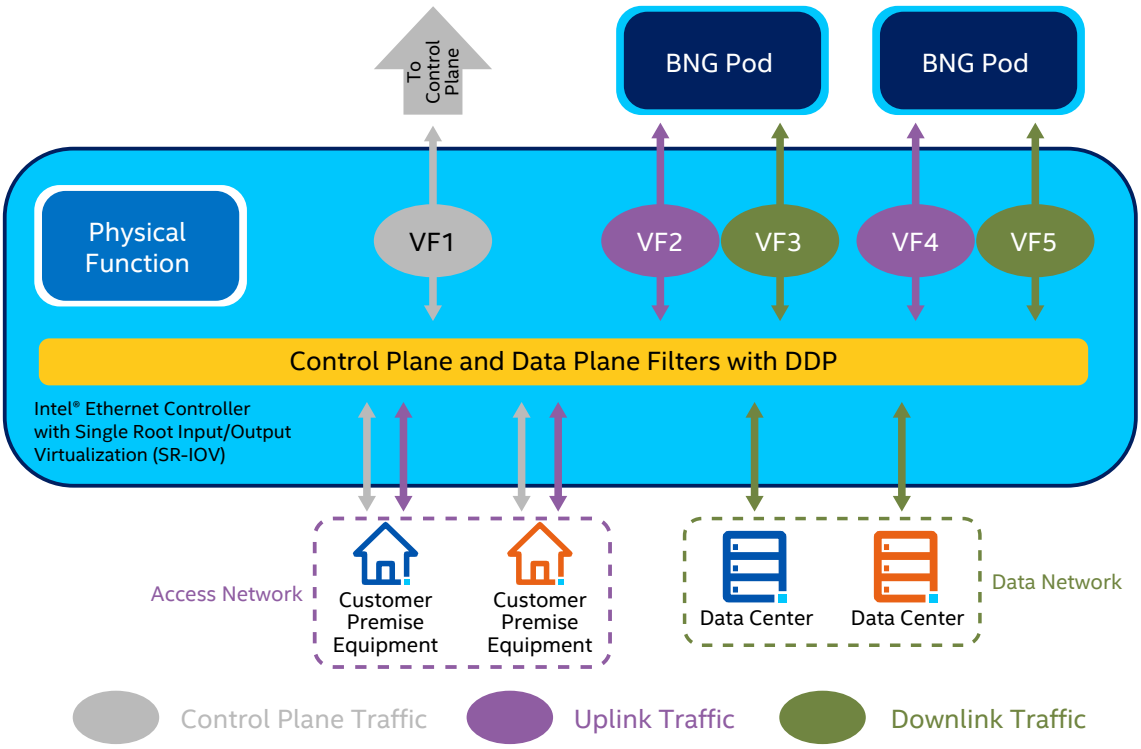


**Figure 8.** Control plane traffic forwarding

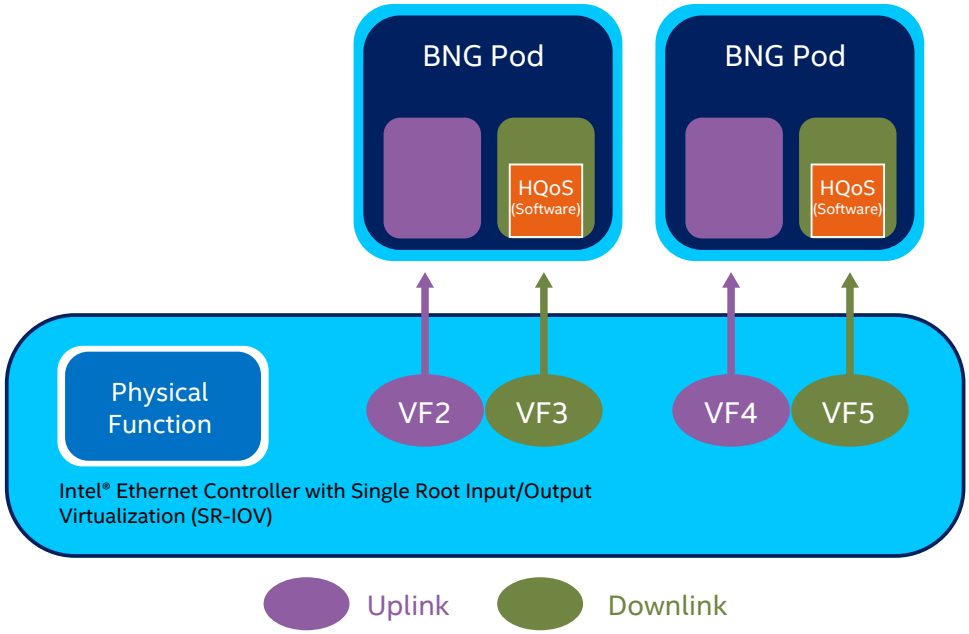


**Figure 9.** Software HQoS model

## Hierarchical QoS (HQoS) Traffic Management

HQoS is implemented within the vBNG downlink pipeline (Figure 9). It ensures traffic priority is preserved when traffic coming from the core network is scheduled for transmission on the reduced bandwidth access network pipes to a subscriber. It also ensures available bandwidth on a given port is shared efficiently across all users.

The HQoS scheduler can either be implemented in the NIC, if support is available, or as a software function in the packet processing pipeline before the transmit function. As discussed previously, each downlink pipeline has a single VF connection for I/O.

In the vBNG reference architecture, the HQoS scheduler is implemented in software. Each vBNG downlink pipeline is apportioned part of the port's overall bandwidth and shapes its traffic to that sub-port rate. The advantage of this method is that no hardware support is needed, and it allows scaling the HQoS scheduler instances with downlink packet processing pipelines. The disadvantage of this method is that unused bandwidth in one vBNG instance cannot be shared with another instance, which may lead to sub-optimal use of the port's bandwidth.

## Date Plane Deployment and Orchestration

Deployment of the vBNG follows a strict microservice model, where each element of application deployment is separated into the smallest possible execution unit that will not affect performance. As can be seen in Figure 10, the full data/user plane deployment (not control plane) is separated into three components:

- **BNG Data/User Plane Management**
  - This section is the interface between the control plane and the data/user plane in a full CUPs deployment.
  - This section is responsible for the following:
    - Receiving data/user plane configuration (PFCP agent)
    - Setting and storing the data/user plane configuration (etcd)
    - Retrieving telemetry data from the data/user plane instances
    - Managing the scale of vBNG pods/instances
- **BNG Data/User Plane**
  - Discussed previously, this section is responsible for the following:
    - The vBNG forwarding pods uplink and downlink
    - The Ligato agent (this agent is seen to be shared between the BNG data/user plane management and BNG data/user plane, but physically it is deployed in the data/user plane)
- **Infrastructure**
  - This section uses K8's CPU and device managers to provide all the EPA features required by the CNF specification for performant throughput.
  - This section is responsible for the following:
    - The K8s kubelet (manages container state of the node)
    - The K8s CPU manager (supplies exclusive vCPUs to the vBNG containers)
    - The SR-IOV device plugin (supplies SR-IOV virtual functions on demand to the vBNG containers)
    - The topology manager (ensures resources received from the host are NUMA topology aligned)

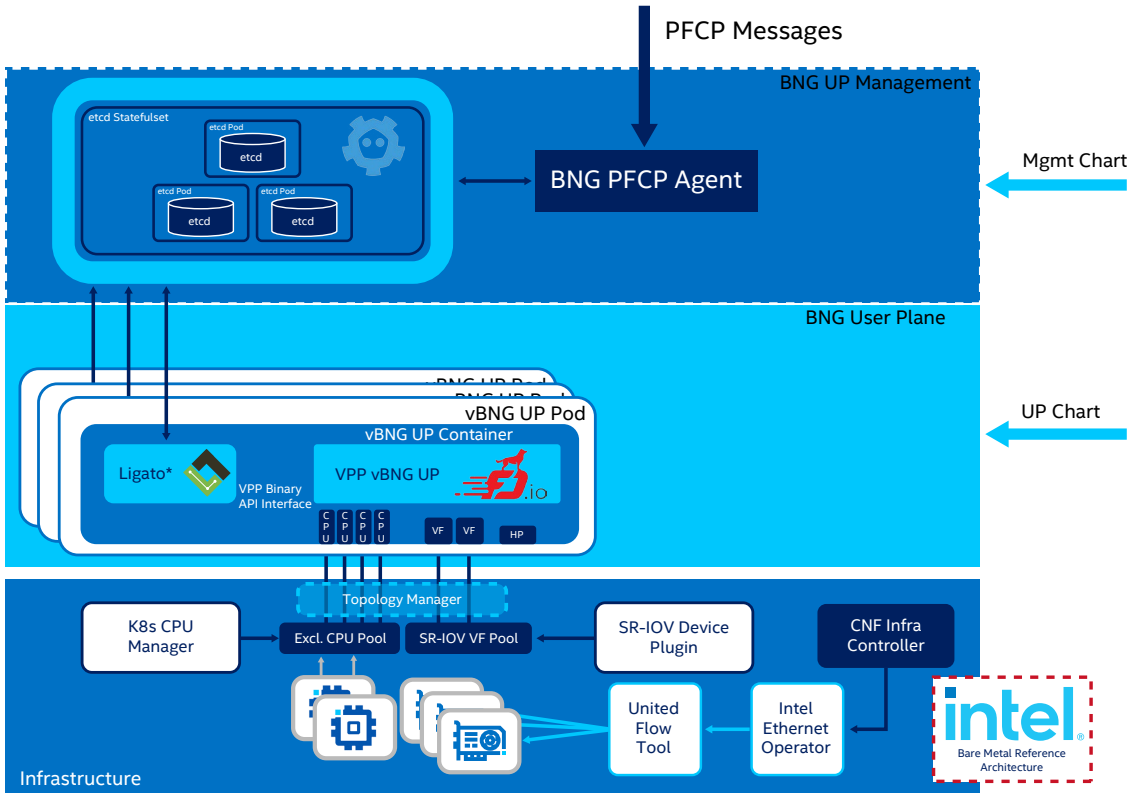


**Figure 10.** Full microservice architecture of a vBNG deployment

## Control Plane Deployment

The control plane used in the vBNG deployment is built by BISDN (Figure 11). The design follows the same microservice architecture as the other components in the vBNG deployment.

Each element is deployed as a container entity. The BNG control plane may be deployed on the same cluster as the BNG data/user plane management and BNG data/user plane, or in a separate remote cluster for commanding multiple BNG clusters. If deployed in the same cluster, the control plane utilises the same container network interfaces (CNIs) as the other BNG components. If the network operations engineers require the BNG control plane to be placed in a remote cluster, it would be expected of them to set up something like the K8s Ingress Controller on the data/user plane cluster to ensure that external BNG control plane access is regulated and load balanced for the data/user plane PFCP agent to receive and parse messages.

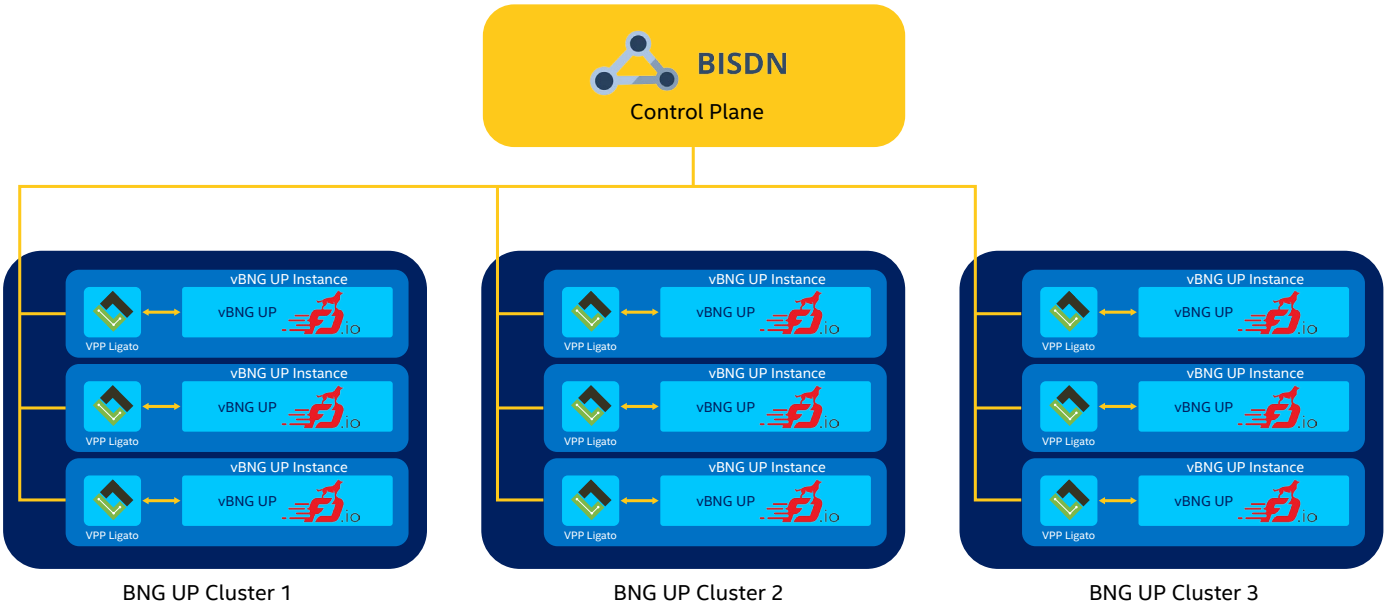


**Figure 11.** 1:N mapping of control plane to data/user plane on multiple clusters

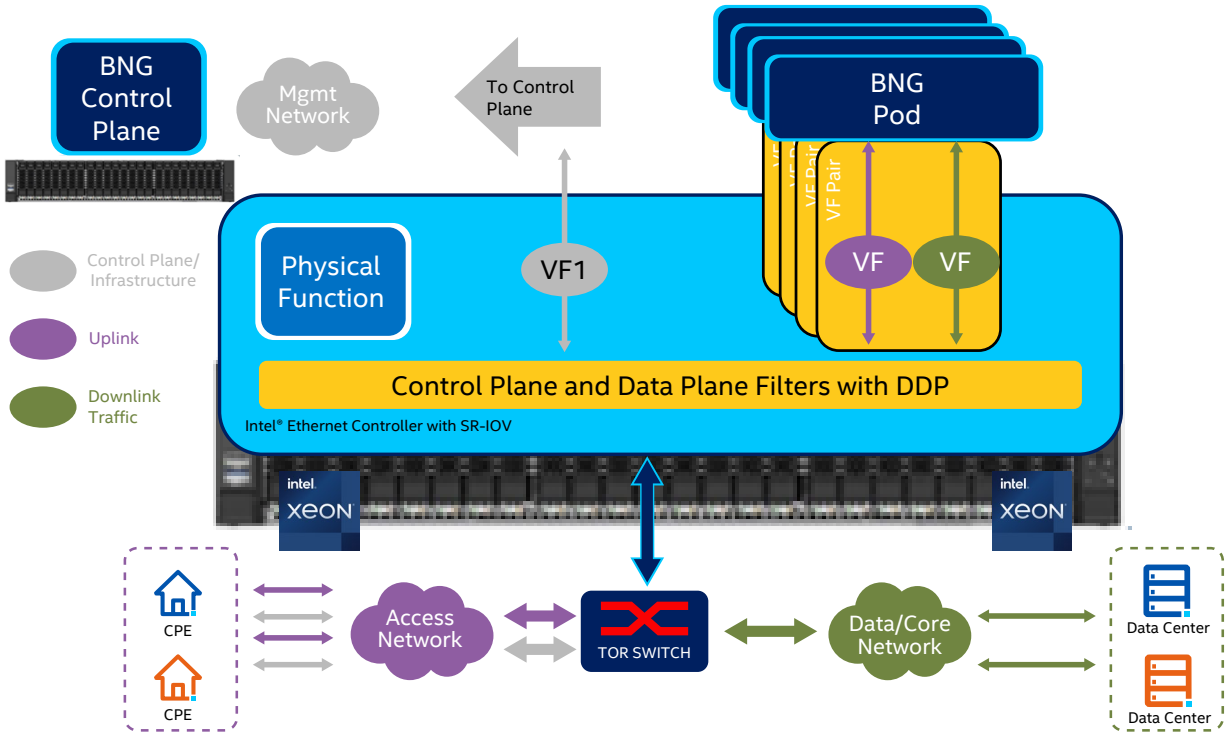


**Figure 12.** Scalable CUPS-enabled vBNG architecture

## Cloud Native vBNG Deployment

By combining all the architecture proposals previously discussed, it is possible to build a scalable, orchestrate-able, and CUPS-enabled BNG solution that efficiently uses the I/O and compute resources of an Intel® processor-based server. Such a solution can help CoSPs address the need to deliver ever-increasing bandwidth at lower cost.

Figure 12 shows a high-level overview of a full CUPs deployment alongside all ingress and egress broadband traffic.

## Performance Benchmarking[7]

Performance measurements on the pipeline blocks shown in Figures 3 and 4 were taken on a dual-socket server running Intel Xeon Gold 6438N processors with the following configuration/benchmark profile:

- Intel® Hyper-Threading Technology (Intel® HT Technology) enabled
- Enhanced Intel SpeedStep® Technology and Intel® Turbo Boost Technology disabled
- Traffic profile applied to all instances: 4K flows, downlink/uplink packet size = 512B/128B, DL/UL ratio of 8:1)

The cumulative throughput (downlink+uplink) across all instances was then measured.

Figure 13 shows the throughput of vBNG application. The throughput scales linearly as four through thirty vBNG instances were deployed with increments of four instances.

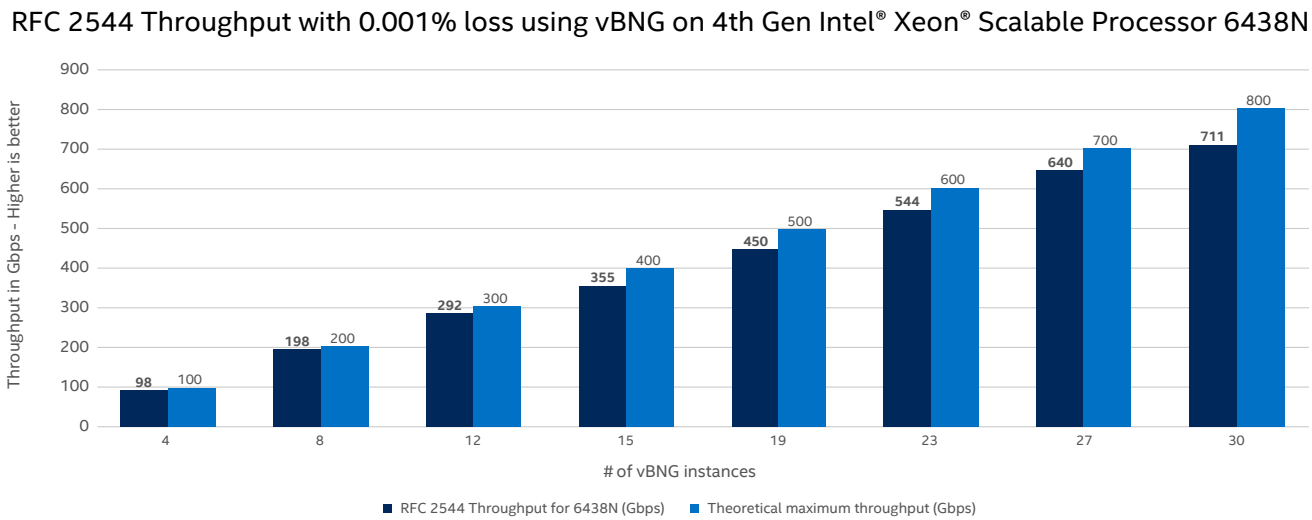


**Figure 13.** 4th Gen Intel® Xeon® Scalable processor-based server (dual-socket) throughput running various vBNG instances

**NOTE: Performance results are based on testing by Intel as of 5/25/2023 and may not reflect all publicly available security updates. See configuration disclosure for details. No product or component can be absolutely secure. Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks. Configurations: See system configuration in Appendix.**

With thirty instances deployed, the throughput is 711 Gbps when using RFC2544 test methodology with 0.001% packet loss. This is achieved using 60 data processing physical cores (1.5 cores per instance for thirty instances). Two cores on each processor were reserved for OS and infrastructure purposes. All resources used by the vBNG application are local to the socket. The application was found to be bound by the number of available CPU cores on the system.

## Power Savings and Efficiency

With the rising costs of energy, power efficiency has become a much higher priority in recent years. Intel provides both technology and software solutions to help reduce power demand and create power savings in the vBNG. These savings are possible without incurring packet loss or introducing latency. The following describes these technologies and software solutions.

### Key Capabilities in Intel Platforms for Power Management

This section describes power management capabilities in Intel® Xeon® processor architectures. Many of these capabilities have been available but not utilized in wireless infrastructure network functions. We can apply these features to improve energy efficiency along with newer capabilities to help reduce both power and transition latencies.

**Core Frequency Scaling (P-state):** This capability (also called core performance state, abbreviated as P-state) allows the frequency of each core in the CPU to be changed independently and dynamically at runtime. The core frequencies in 4th Gen Intel Xeon processors can be adjusted at a resolution of 100 MHz, allowing fine-tuning the frequency to match the actual traffic load. With a vBNG running on a COTS server with 4th Gen Intel Xeon Scalable processors, effectively controlling P-states as traffic load allows can result in electrical power savings.

**Uncore Frequency Scaling:** This capability enables changing the frequency of the CPU logic that interconnects cores, L3 caches, memory, and I/O controllers. The optimal uncore frequency selection can be determined by CPU hardware or by a software-driven selection.

The 4th Gen Intel Xeon processor has the following additional improvements that reduce the latency associated with power state transitions. These include:

- **Fast core frequency change:** This allows the core frequency to move from the current to the target frequency in a continuous sweep without stopping the clocks. This allows fast P-state transitions to optimize power versus performance without the latency cost.
- **Coherent fabric ("mesh") drainless frequency change:** This allows the uncore mesh PLL to transition from the current to the target frequency without draining the buffers, reducing the frequency transition time by about 3X.

**Per-Core C-States:** Typically, each CPU core executes instructions in the C0-state. Execution can be halted, and the core can be transitioned into a lower power state, such as C1 or C6, for a period of time. Deeper power states have greater power savings but also longer latency when transitioning back to C0 to resume execution. Controlling C-states to put the cores into a shallow sleep state during gaps in traffic can result in additional power savings. Software instructions can directly initiate a transition to C1 or C6, or software can provide hints to the processor.

4th Gen Intel Xeon processors implement additional sub-C0 power states called C0.1 and C0.2. These states are shallower than C1 with reduced exit latencies. They can be exercised by user space software through a new timed pause instruction, called TPAUSE, first available in the 4th Gen Intel Xeon Scalable processors. The instruction tells the processor to enter an implementation-dependent optimized state until the time stamp counter of the processor reaches a predetermined value.

**Intel® Infrastructure Power Manager (Intel® IPM):** Intel IPM is a cloud-native software solution that dynamically controls per-core power state at runtime in response to VNF traffic flows. It is built to be deployed as a Kubernetes pod on each worker node as shown in Figure 14.

vBNG data/user plane traffic fluctuates significantly over time, with somewhat predictable patterns of activity over the course of the day. At a more granular level, traffic volume also varies substantially among individual data/user plane flows. Those individual fluctuations offer opportunities to save energy using Intel IPM to optimize the granularity and responsiveness of power policy at the per-core level.

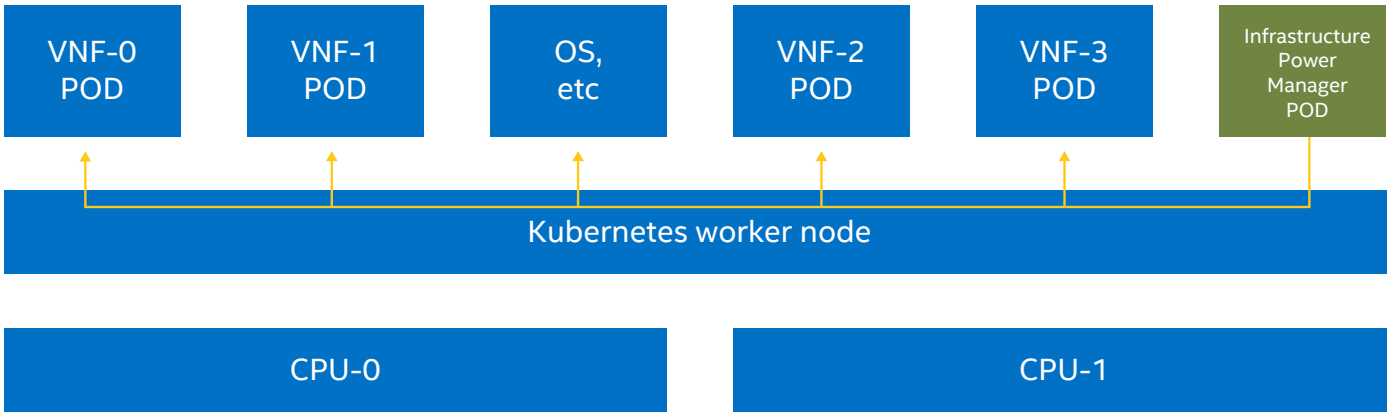


**Figure 14.** Deployment of Intel Infrastructure Power Manager as a Kubernetes pod.

Packet rate is strongly correlated with how the core is utilized to process packet flow. Intel IPM tracks vBNG worker core utilization at millisecond granularity, a level of detail enabling significant power reduction without impacting performance KPIs. Per-core frequency (P-state) can therefore be modulated in accordance with core utilization as a proxy for packet flow. The solution continually re-computes the target P-states of processor cores to set higher or lower core frequencies as needed, with a configurable bias toward slow reduction and fast increase. Implementation does not require software changes to the vBNG, other than patch integration for the VPP/DPDK plugin.

## Power Saving Measurements on the vBNG Reference Software

The above power savings technologies were implemented for the Intel vBNG reference software in order to measure achievable power savings. Four tests were run on the system as follows using a typical 24-hour traffic profile shown in Figure 15.

**1: Default setup (no power savings):** For this test case, the vBNG was run in a similar setup as the performance benchmarking test above. The data/user plane traffic was evenly sent to the 30 active vBNG instances at various fractional percentages of the maximum achievable throughput of the RFC 2544 test that was run previously. In this way, system power used at varying levels of traffic can be measured, while still ensuring no packet loss. Traffic loads of 10, 20, 30 percent, and so on, up to 100 percent of RFC2544 throughput were sent to the system under test. The wall power (server power) in watts was measured for each of the throughput levels. Also, average cut-through latency was measured.

**2: C-states enabled:** For this test case, a patch was applied to the vBNG software instances to use the TPAUSE instruction in cases when traffic fell below a threshold. For this test, if there were more than 16 empty polls in the NIC polled mode driver and HQoS module, the TPAUSE would send the relevant CPU cores into the C0.1 sleep state, thus saving power.

**3: P-states enabled:** For this test case, Intel IPM was used to regulate the CPU core P-states, based on the telemetry data from the vBNG application. When the vBNG was less busy due to less traffic, the CPU cores were transitioned to a lower P-state, reducing core frequency, and thus saving power. The uncore frequency on the CPU socket was also reduced if resource utilization was down, enabling further power savings.

**4: C-states and P-states enabled:** For this test case, a combination of the technologies in test cases 2 and 3 was enabled, enabling further power savings.

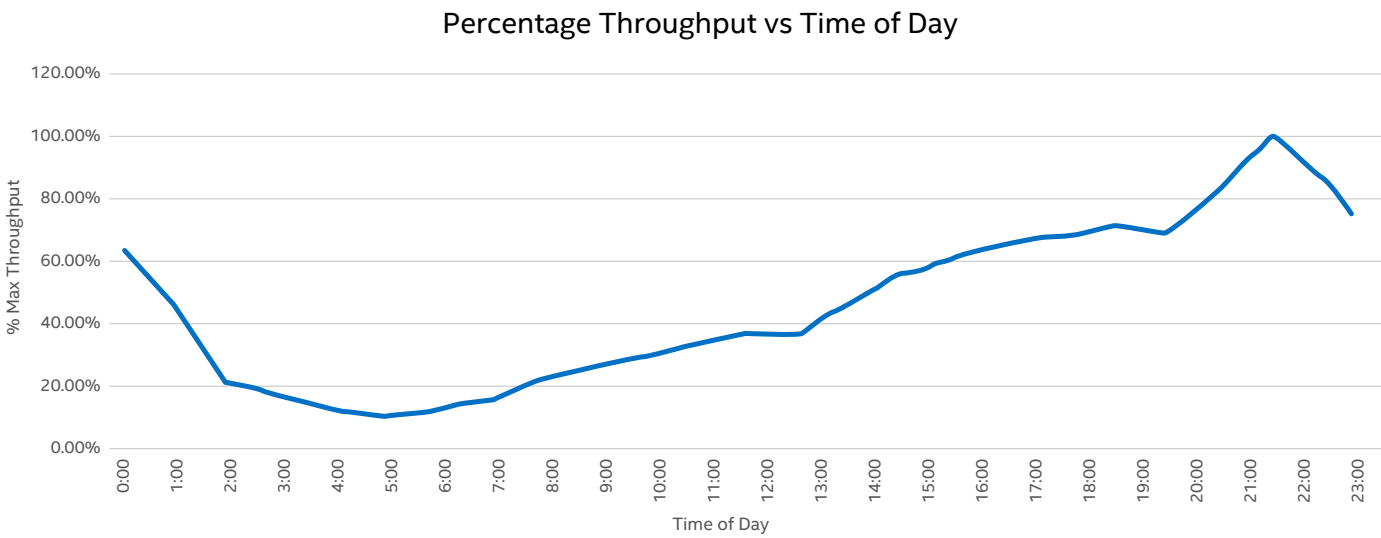


**Figure 15.** vBNG traffic throughput variation over a typical 24 hour period

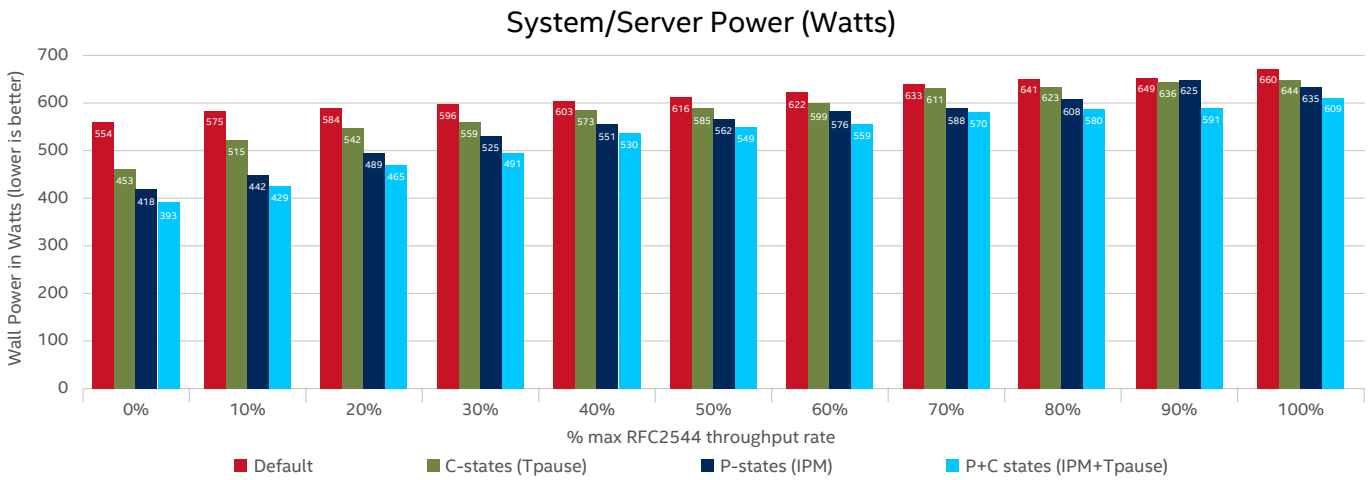


**Figure 16.** System power used vs percent max throughput of vBNG, with various power savings technologies enabled.

The results of the four test cases can be seen in the chart in Figure 16. The cumulative effect of the various power saving technologies is evident in the chart.

Figure 17 shows the Average Cut-Through Latency of the downlink pipeline. It can be observed that the use of power savings technologies does not have a significant effect on latency. Also, it is worth noting that there are no packet losses beyond the 0.001 percent threshold in the RFC2544 test.

Figure 18 below shows the power savings possible with the four different test case scenarios over a typical 24 hour traffic period.

These measurements show that a peak system (server) power saving of up to 25 percent can be achieved. Over a typical 24 hour period, an average system (server) power saving of up to 14 percent can be achieved.

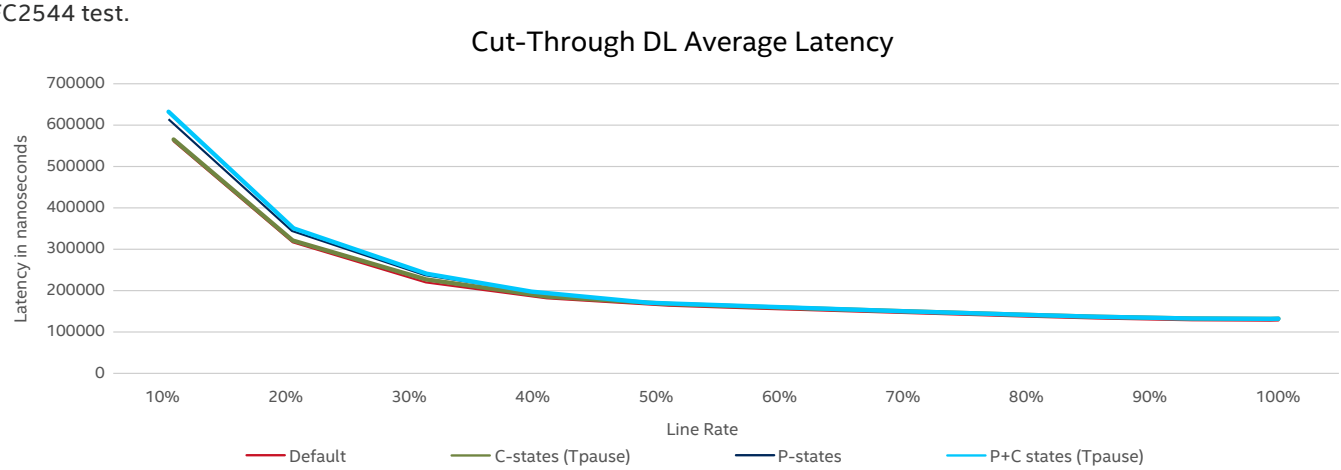


**Figure 17.** Chart of cut-through average latency for downlink vs percent throughput.

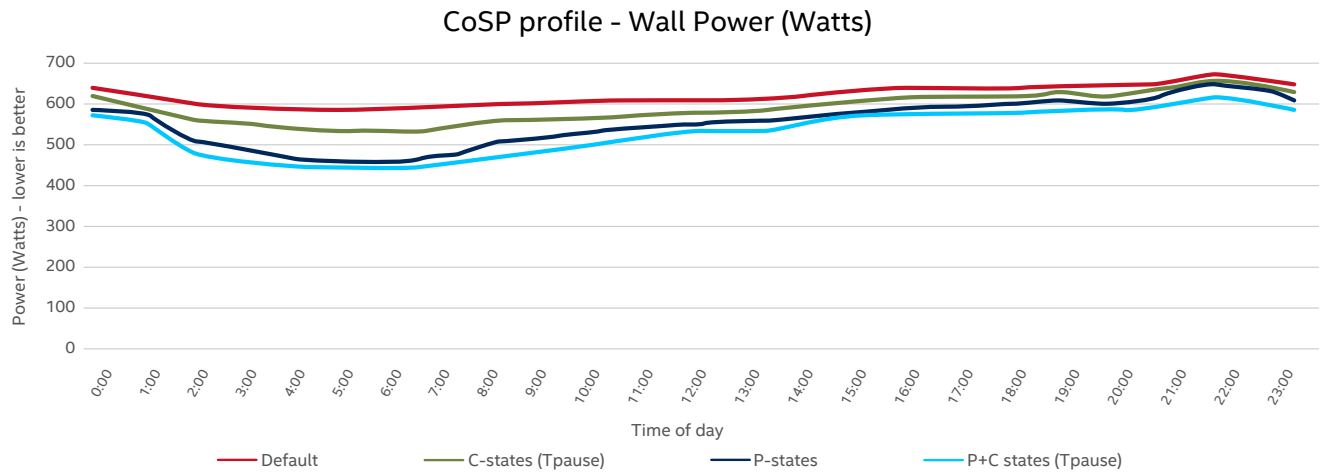


**Figure 18.** Power savings vs time of day using typical CoSP vBNG network traffic profile, comparing the various power saving test cases.

## Summary

The future viability of NFV-based networking equipment running on general-purpose/COTS servers hinges on the ability to service ever-increasing traffic volume in a cost-effective manner.

This paper presented the architectural considerations and associated benchmarking data that demonstrate the significant potential for cloud-native vBNG data/user plane deployment on 4th Generation Intel Xeon Scalable Processors. The CUPS-based vBNG architecture allows CoSPs to predictably, reliably, and independently scale control and data/user planes on general purpose servers. The vBNG data/user plane partitioning into uplink and downlink packet processing pipelines facilitates high throughput and scalability on multi-core CPUs.

Test results show that the maximum overall performance reaches 710 Gbps on the Intel Xeon Gold 6438N processor under an operator's standard traffic test model with 128-byte uplink traffic packets and 512-byte downlink traffic packets without hardware acceleration. In addition, by leveraging processor P-states and C-states, peak power savings of up to 25 percent and average power savings of up to 14 percent can be achieved over a typical 24-hour traffic profile. The benchmarking illustrates that CoSPs can attain exceptional performance while maintaining power efficiency, resulting in reduced operational costs.

## Appendix

The tables in this appendix list the various hardware and software configurations of the vBNG reference architecture described in this paper.

| Power Test vBNG Server | |
|---|---|
| **Platform** | Intel Archer City Reference Server |
| **CPU** | 2x Intel® Xeon® Gold 6438N Processor, 2.0 GHz, 32 Cores |
| **Memory** | 16x32 GB DDR4 |
| **Hard Drive** | INTEL SSDSC2BA40 (400GB SSD), WDC WDS100T2B0A (1TB SSD) |
| **Network interface Card** | 4x Intel® Ethernet Network Adapter E810 2CQDA2 |
| **Software** | |
| **Host OS** | Red Hat Enterprise Linux 9.1 (Plow) |
| **vBNG** | 22.08 |
| **Linux Container** | Podman version 4.2.0 |
| **VPP** | v22.02 |
| **BIOS Settings** | P-state Enabled, HT ON, C-States Enabled, Turbo Boost Disabled, SRIOV and VT-d enabled |

| Application Configuration per Instance | |
|---|---|
| **Uplink** | |
| Frame Size: 256B*; 11% of Overall Traffic; Subscribers: 4K/Instance; 1x vCPU per Instance | |
| **ACL** | Blacklist with 150 Rules |
| **Flow Classification** | Flows Classified on VLAN Tag Pair |
| **Policer/Metering** | Two Rate Three Colour Marker |
| **Routing** | Single Forwarding Rule |
| **Downlink** | |
| Frame Size: 1000B*; 89% of Overall Traffic; Subscribers: 4K/Instance; 2x vCPU per Instance | |
| **ACL** | Reverse Path forwarding – One Rule per Subscriber (4k) |
| **HQoS** | 5 Level HQOS Port, Subport, Pipe, Traffic Class, and Queue |
| **Routing** | One Route per Subscriber (4K) |

| Performance Test vBNG Server | |
|---|---|
| **Platform** | Intel Archer City Reference Server |
| **CPU** | Intel® Xeon® Gold 6438N Processor, 2.0 GHz, 32 Cores |
| **Memory** | 16x32 GB DDR4 |
| **Hard Drive** | Intel® SSD D3-S4510 Series (480G) |
| **Network interface Card** | 4x Intel® Ethernet Network Adapter E810-2CQDA2 |
| **Software** | |
| **Host OS** | Red Hat Enterprise Linux 8.2 (Ootpa) |
| **vBNG** | vBNG 20.11 |
| **Linux Container** | Docker version 20.10.5, build 55c4c88 |
| **VPP** | 22.02 |
| **DPDK** | 21.11 |
| **BIOS Settings** | P-state Disabled, HT ON, C-States Disabled, Turbo Boost Disabled, SRIOV and VT-d enabled |

| Application Configuration per Instance | |
|---|---|
| **Uplink** | |
| Frame Size: 128B*;11% of Overall Traffic; Subscribers: 4K/Instance; 1x vCPU per Instance | |
| **ACL** | Blacklist with 150 Rules |
| **Flow Classification** | Flows Classified on VLAN Tag Pair |
| **Policer/Metering** | Two Rate Three Colour Marker |
| **Routing** | Single Forwarding Rule |
| **Downlink** | |
| Frame Size: 504B*;89% of Overall Traffic; Subscribers: 4K/Instance; 2x vCPU per Instance | |
| **ACL** | Reverse Path forwarding – One Rule per Subscriber (4k) |
| **HQoS** | 5 Level HQOS– Port, Subport, Pipe, Traffic Class, and Queue |
| **Routing** | One Route per Subscriber (4K) |

*Frame size quoted is max size of frame at any point in processing. (e.g. uplink 128Byte =120byte +{2x4Byte access VLAN tags})

| Test Environment Configuration Information and Relevant Variables | |
|---|---|
| **Traffic Generator** | **IXIA* NOVUS* 100GE8Q28** |
| **Connection Details** | Ixia Ports and DUT Ports Connected Back-to-Back (Eight Connections) |

## References


1,3. "Broadband ISP TalkTalk UK Predicts Data Traffic Growth to 2027" https://www.ispreview.co.uk/index.php/2023/02/broadband-isp-talktalk-uk-predicts-data-traffic-growth-to2027.html

2,7. Benchmark and performance tests ("benchmarks") measure different aspects of processor and/or system performance. While no single numerical measurement can completely describe the performance of a complex device like a microprocessor or a personal computer, benchmarks can be useful tools for comparing different components and systems. The totally accurate way to measure the performance of your system, however, is to test the software applications you use on your computer system. Benchmark results published by Intel are measured on specific systems or components using specific hardware and software configurations, and any differences between those configurations (including software) and your configuration may very well make those results inapplicable to your component or system.

4. "Architecture for Control and User Plane Separation on BNG", https://datatracker.ietf.org/doc/draft-wadhwa-rtgwg-bng-cups/

5. Huawei* website, "NE40E V800R010C00 Feature Description - User Access 01, https://support.huawei.com/enterprise/en/doc/EDOC1100027162?section=j01i&topicName=pppoe-packet-format

6. "Intel® Ethernet Controller 800 Series - Dynamic Device Personalization (DDP) for Telecommunications Workloads" , https://builders.intel.com/docs/networkbuilders/intel-ethernet-controller-800-series-device-personalization-ddp


## Notices & Disclaimers

Performance varies by use, configuration and other factors. Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks. Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade. You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein. The products described 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. No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

© 2023 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.