White Paper

Architecture Study
Broadband Network Gateway

Re-Architecting the Broadband Network Gateway (BNG) in a Network Functions Virtualization (NFV) and Cloud Native World

Authors

Padraig Connolly
Intel Platform Applications Engineer

Andrew Duignan
Intel Platform Solutions Architect

Shivapriya Hiremath
Intel Platform Applications Engineer

Tommy Long
Intel Software Architect

Jasvinder Singh
Intel Software Engineer

Eoin Walsh
Intel Platform Solutions Architect

Executive Summary

The ever-growing consumer demand for more bandwidth and services for less money has been driving service provider networks to their economic limit for some years now. 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 data traffic estimated to grow at a 26 percent rate (year over year) from 2017 to 2023,[1] future networking solutions must show a path to solving tomorrow's data compound-annual-growth-rate (CAGR) challenge in a cost-effective and scalable way.

Pushing the boundary of performance requires the latest and greatest technology along with deploying this technology in a holistic and easy-to-use way. This paper proposes the following set of design principles for taking solutions based on Intel® architecture processors and network functions virtualization (NFV) to the next level of performance and network automation by utilizing the following:

- Optimized server configuration
- Software "run-to-completion" model
- Intelligent I/O packet and flow distribution
- Independent scaling of the control and user planes
- Hierarchical quality of service considerations
- Deploying using Cloud Native Networking Fundamentals

Following these principles, Intel has demonstrated nearly 661 Gbps[2] of routing RFC2544 (zero packet loss) performance for a virtual broadband network gateway (vBNG) running on a single 3rd Gen Intel® Xeon® Scalable processor server. This paper describes this effort and proposes a vBNG architecture for building network infrastructure and network functions to better take advantage of the underlying infrastructure and address the challenge of the data CAGR.

To complement the shift of the Broadband data plane into a virtual ecosystem, this paper also puts forward a deployment architecture using the container orchestration engine Kubernetes (K8s).

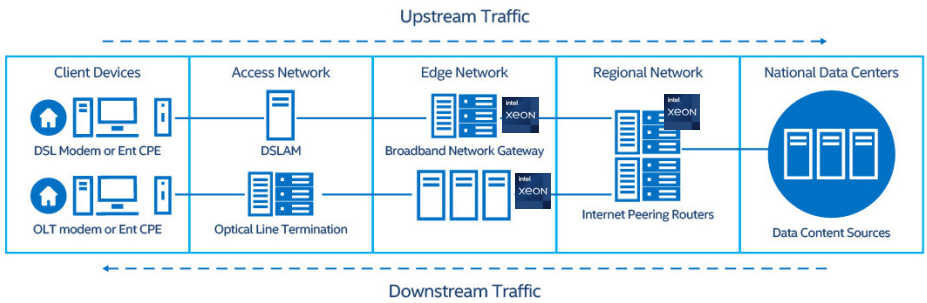

Figure 1. Example of a Network Connecting Clients to a Data Center

Table of Contents

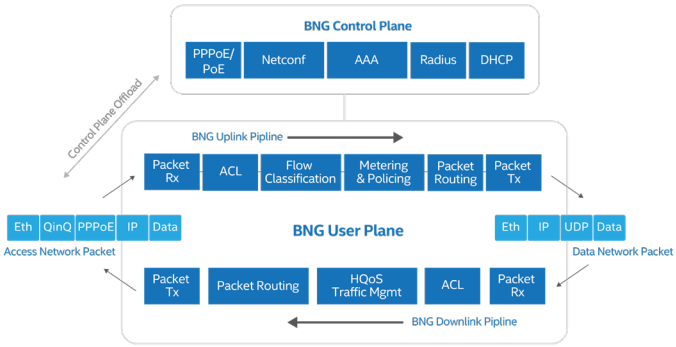

Figure 2. Virtual Broadband Network Gateway (vBNG) Control and Data Plane Blocks

Broadband Network Gateway

The BNG, aka broadband remote access server (BRAS), is the network edge aggregation point used by subscribers to access the Internet service provider (ISP) network. Through the BNG, subscribers connect to the ISP network to download Internet originating traffic and ISP services (e.g., web, voice, data, and video).

The vBNG is a virtualized software instantiation of what is typically a large, ASIC-based, fixed-function appliance usually located in a central office or metro point of presence (PoP).

Reference Application Pipeline

Each generation of Intel technologies (e.g., CPU, NIC, SSD, FPGAs, and accelerators) brings new opportunities to improve performance and quality of experience (QoE) for users. Showing how to take advantage of these technologies, Intel builds reference pipelines, like the representative stages of vBNG control and data plane functions shown in Figure 2.

The control plane is responsible for subscriber authentication and management, including monthly usage service access and data plane configuration, based on subscriber profiles.

The upstream data plane manages the flow of traffic from users' home routers to the ISP network. The average packet size for upstream traffic is generally smaller than for downstream traffic, and the amount of upstream traffic is normally five to eight times less than downstream traffic. While applications like Instagram, Snapchat, and Periscope, have seen larger swathes of data being pushed onto the ISP network by users, broadband users are still overwhelmingly net consumers of data. In recent years, data and content creation has reduced the gap between upstream and downstream bandwidth usage.

Upstream Processing Stages

The Intel reference pipeline 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, 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 permit and deny filters, and 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.

DSCP Rewrite: This stage supports the optional classification of the traffic type and rewrite of the IP differentiated services code point (DSCP) field to map the stream to a network supported class of service (CoS).

NAT: Optionally, NAT 44 is performed to convert private addresses to public addresses.

Routing: Access network encapsulations are stripped from data plane packets, and the packets are routed to the correct core network interface for transmission. Any core 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.

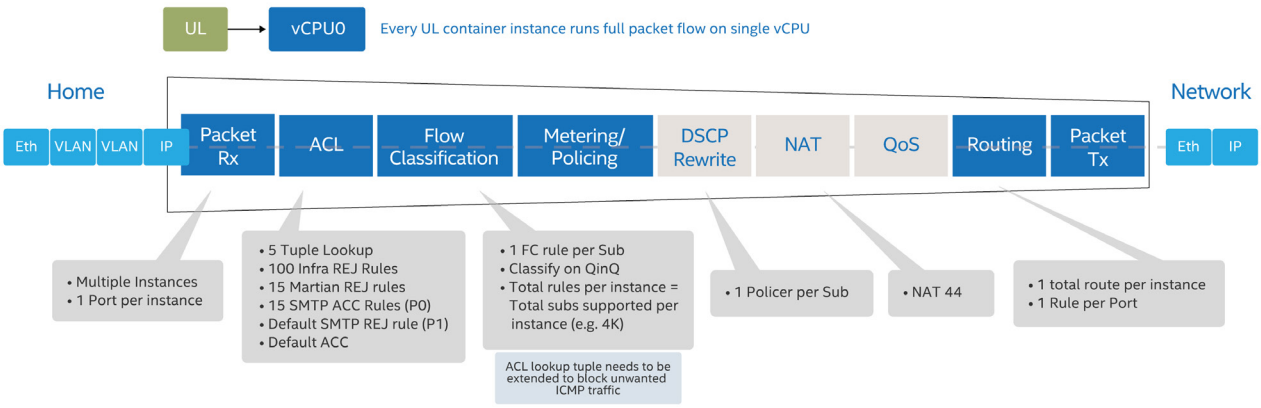

Figure 3. Uplink Packet Flow Stages

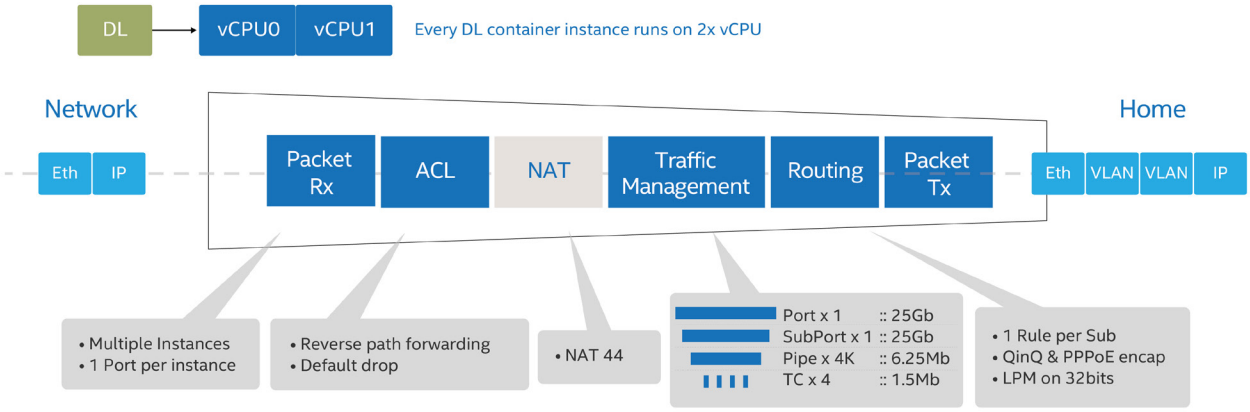

Figure 4. Downlink Packet Flow Stages

The downstream data plane handles 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 their subscribers are receiving services to the highest standard while maximizing the utility of the network infrastructure. By 2022, global IP video traffic is forecast to grow four-fold from 2017 to 2022, a CAGR of 29 percent,[3] and this trend will drive up the average packet size of the downstream link.

Downstream Processing Stages

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

Packet Rx: The DPDK PMD receives 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 Access Control List (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.

NAT: Optionally, NAT 44 is performed to convert public addresses to private addresses.

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. It 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.

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 Txblock.

Packet Tx: Using a DPDK polled mode driver (PMD), bursts of frames are transmitted to the NIC port.

New Architectural Proposal and Reasoning

In order to effectively deploy a BNG workload on a general-purpose server, the following architectural and implementation aspects should be considered:

Implementing a Run to Completion Model

One of the key considerations when designing a software-based BNG is ensuring performance scalability per this paper's opening problem statement. The BNG 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 BNG must be able to scale both up and down based on the current workload.

The Intel reference BNG pipeline uses a run to completion model to process the uplink and downlink pipelines. As a result, all pipeline functions executed on a packet are run on the same core. This has advantages in that packets do not have to move between cores, thereby minimizing cache misses and overall latency. A direct result of this design pattern is that an individual vBNG instance cannot scale out beyond a single core. Scaling beyond a single core is done by creating a new vBNG instance that runs on a different core. The NIC is programmed (using custom headers supported by the Comms DDP package) on the fly to direct specific subscribers to each new individual vBNG instance (More on this further on in the paper).

The combination of a run to completion model and a single core running a single vBNG instance eliminates the need for the orchestrator to understand the internal operation of the vBNG application to scale. The orchestrator can scale capacity up or down by increasing or decreasing the number of CPU cores assigned to the BNG deployment (5 vCPUs per Instance with K8s), enabling linear scalability across a given server. The orchestrator can optimize resource utilization when it is furnished with information regarding the number of subscribers (for a known traffic profile) that a single core instance can support, which may vary by CPU SKU.

Separating Uplink and Downlink Processing

CPU resource usage by the BNG uplink and downlink pipelines are not symmetric since the downlink normally requires more cycles per packet due to inherently larger packet sizes. In order to effectively schedule a BNG, the Intel reference pipeline splits the uplink and downlink into

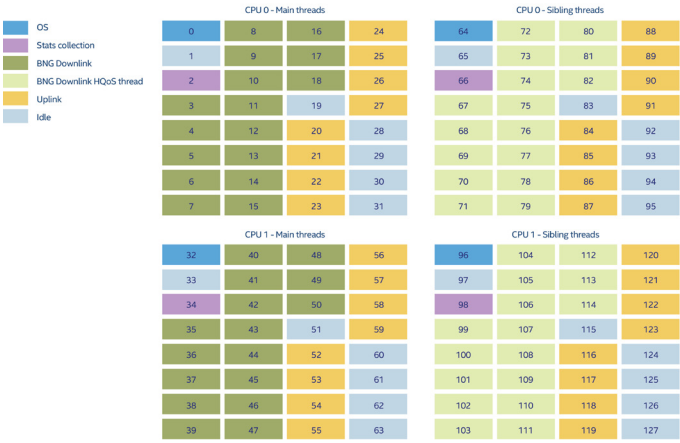

Figure 5. CPU Core Allocation Example for 16vBNGs Instances Running per Socket

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-threaded cores) while an uplink pipeline might only require half a physical core (one hyper-thread core). Figure 5 shows how the CPU resources of a dual socket server could be partitioned when running sixteen vBNG instances per socket in a Docker only configuration (i.e not using K8s). Each pipeline can report telemetry individually, and the telemetry database can be used to maintain all relevant usage statistics. It is also worth noting whilst uplink and downlink pipelines are deployed as separate DPDK 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 reference pipeline should be run on a BNG dataplane server connected to a basic leaf switch that can route both access and data network traffic. With this setup,the switch routes uplink traffic coming from the access network ports to the BNG ports for processing and routes returning packets from BNG uplink pipelines to the data network ports. The flow is reversed for downlink traffic. As mentioned previously, the amount of uplink traffic is increasing over time, but it is generally only an eighth of the downlink traffic in a wireline network. Therefore, a BNG that uses separate,dedicated physical ports for access and data network port connections is likely to under utilize the available I/O bandwidth of the uplink ports. Instead, sharing physical ports on a NIC between upstream and downstream traffic allows I/O bandwidth to be fully utilized. 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.

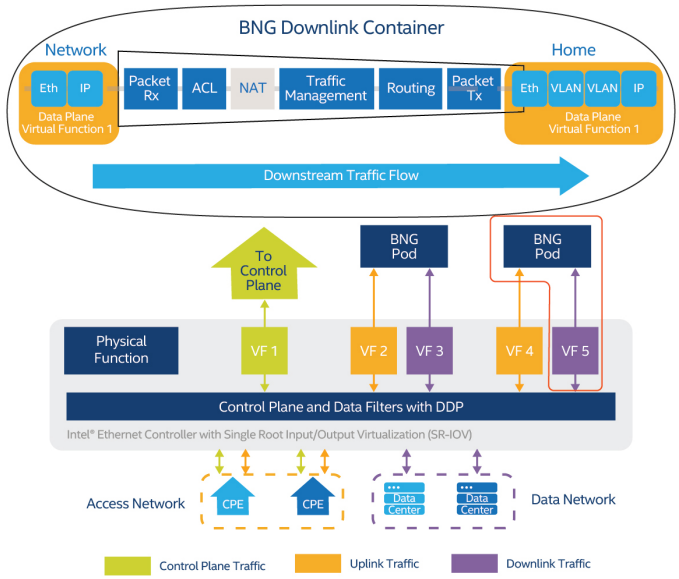

Figure 6. BNG Downlink Container Using A Single SR-IOV VF for Both Rx and Tx

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. As NIC speeds hit 100 gigabit, it is expected that downlink and uplink traffic will share the same physical NIC and these principals influence the deployment architecture of the vBNG today.

Balancing I/O on the Server

With dual socket servers, internal connections such as Platform Controller Hubs (PCHs) and SATA controllers are commonly connected to CPU 0, as shown on the left side of Figure 7. This can result in an uneven distribution of PCIe I/O bandwidth between the CPUs, with most of the bandwidth being connected to CPU 1. To balance the bandwidth, the Intel vBNG application runs control plane functions on CPU 0 and data plane functions on CPU 1, as shown on the right side of Figure 7.

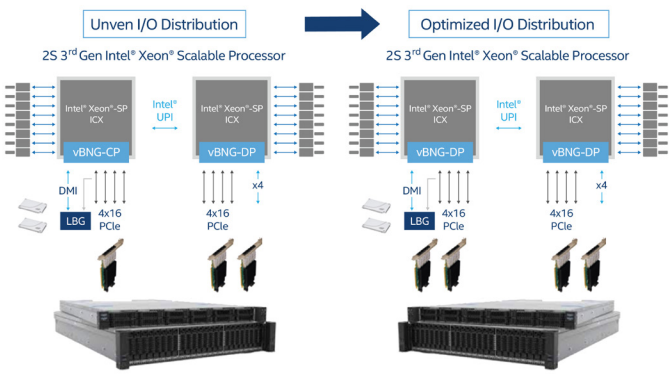

Figure 7. Balancing I/O on a Server

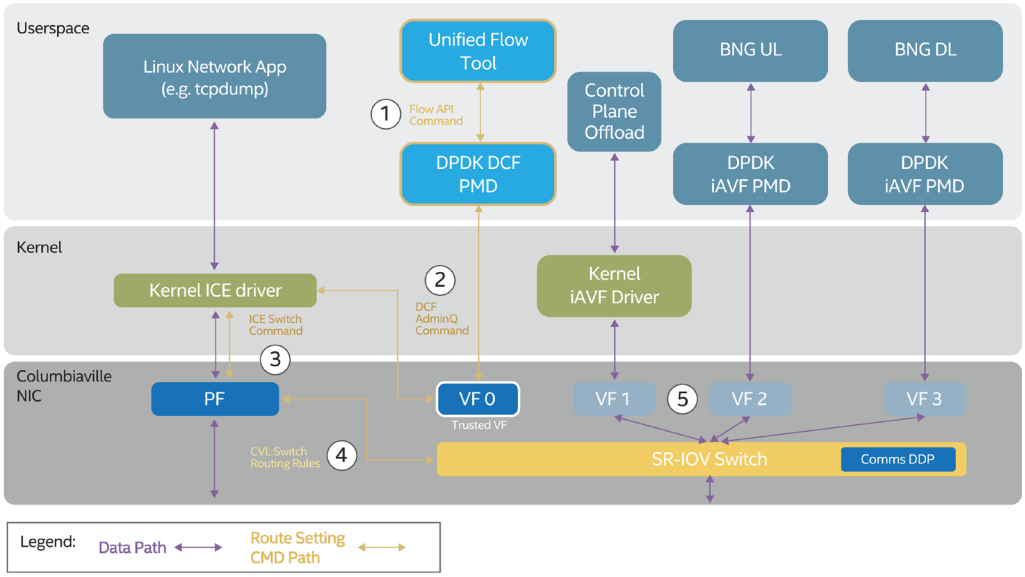

Figure 8. Device Config Function Workflow

When deploying a BNG on a general-purpose server, it is important to ensure there is enough I/O bandwidth to fully utilize the available CPU resources on the platform (i.e., the aim is to be CPU bound and not I/O bound). The advent of Control and User Plane Separation (CUPS)[4] for BNG enables an entire server to be dedicated to running the BNG data plane. All data processing is localized to a single socket for performance efficiency, which necessitates an equal amount of I/O to be connected to each socket to achieve optimal performance. The provisioning of 2x16 or 4x8 lane PCIe Gen 4 (16 GT/s) slots on each socket provides a total I/O bandwidth of 800 Gbps on the server, equally balanced across the two sockets (see the Appendix for server configuration details).

Distributing Flows Via a Network Interface Card (NIC)

As described above each BNG instance has a set number vCPUs processing subscriber traffic. Some form of distributer is required in order split out the subscriber flows among the vCPUs. This distributor task can be performed in software using dedicated cores, but there are several disadvantages to this approach. First, this distributer function can become a performance bottle neck as all flows must pass through this software function. Second, having one or many cores dedicated to performing distribution reduces the amount of CPU cycles available to perform the actual BNG workload processing, thus reducing the number of BNG subscribers the server can handle.

These disadvantages can be overcome by distributing the flows in the NIC to either SR-IOV VFs or Queues in the PMD, which eliminates the software bottle neck, reduces latency through the system, and provides the BNG the CPU cores and cycles that otherwise would be used by the distributor task.

Assigning a Single I/O Connection per Pipeline

Device Config Function (DCF)

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 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 Switch is used and for distributing flows amongst Queues, Flow Director (FDIR) or Receive Side Scaling (RSS) is used. In the BNG deployment, flows are distributed using VFs thus that is the focus of this paper. Further information on RSS and FDIR can be found in other Intel Telco white papers.

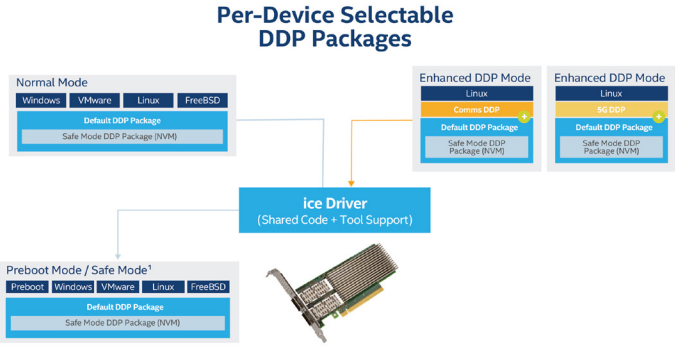

Figure 9. DDP Packages for E810 NIC

Dynamic Device Personalization

Alongside DCF is Dynamic Device Personalization (DDP)[6] which allows the SR-IOV Switch 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 (Figure 9). 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 10) based on the unique PPPoE header fields (described more in the next section).

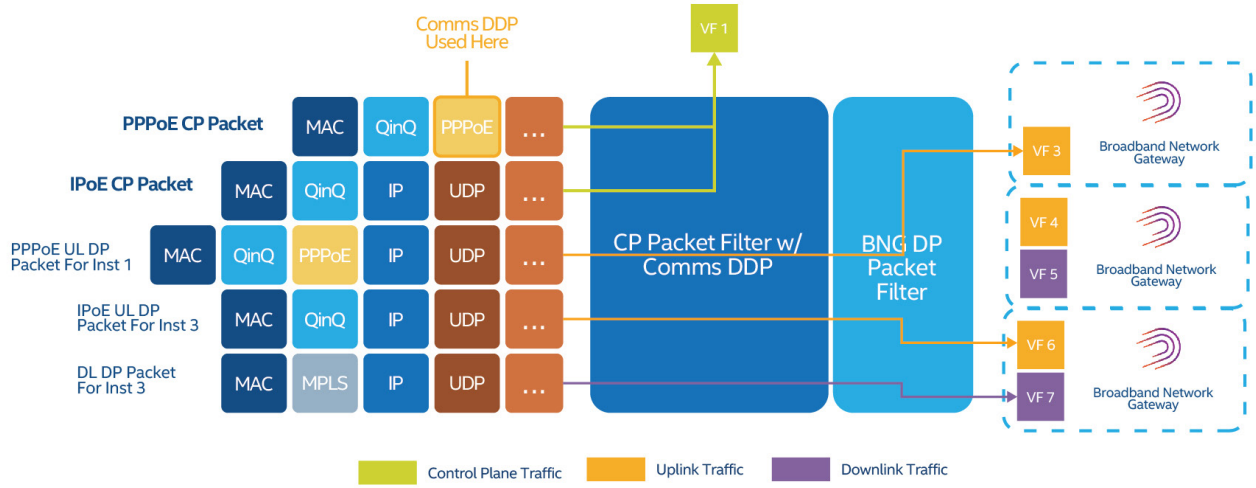

Figure 10. NIC Routes Packets to Virtual Functions based on PPPoE and DHCP Packet Headers

Forwarding Control Plane Packets

For control plane traffic, like PPPoE session setup or PPPoE link control packets, the BNG data plane must identify and forward these packets to the control plane for processing. In a traditional BNG, the control plane and the data plane are located in the same place and a local software queue is used to move packets between them. With the advent of CUPS, the control plane and the corresponding data plane is most likely located in different physical locations in the network. In this case, the BNG data plane needs to pass control packets to the control plane by generating a physical link to forward them. As can be seen in Figure 10, the Intel Ethernet Network Adapter E810 is able to identify these control packets using the Comms DDP package and forward them to a separate VFs and queues (I/O), relieving the data plane of this task. The Comms DDP package enables the Intel NIC to recognize these control plane packets and forward them the control plane. Figure 11 gives a higher level view of how this works with the BNG application deployment.

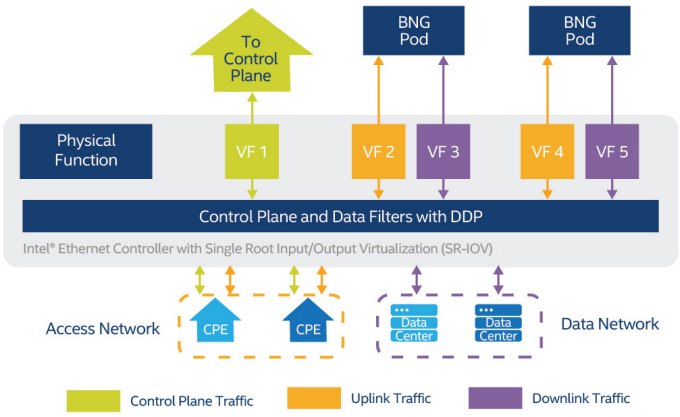

Figure 11. Control Plane Traffic Forwarding

Considering HQoS

Hierarchical QoS is a function that is implemented within the vBNG downlink pipeline. It ensures traffic priority is preserved when traffic coming from the core network is scheduled for transmission on the reduced bandwidth access network pipe to a subscriber, and the available bandwidth on a given port is shared efficiently across all users. The HQoS scheduler can either be implemented in NIC if support is available or as a software function in packet processing pipeline before the transmit function. As discussed previously, each downlink pipeline has a single virtual function connection for I/O. The following sections describe three models for implementing HQoS:

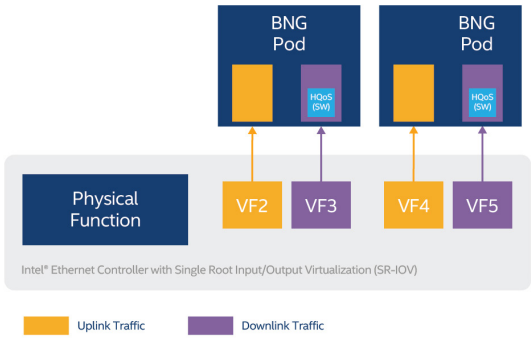

Figure 12. Software-Only HQOS Model

Software-Only Model

A software-only model fully implements HQoS schedular in software. As shown in Figure 12, 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 no hardware support is needed, and allows to scale the HQoS scheduler instances with downlink packet processing pipelines. The disadvantage of this method is unused bandwidth in one vBNG instance cannot be shared with another instance, which may lead to sub-optimal use of the port's bandwidth.

Hardware/Software Hybrid Model

A hybrid model can be used when the resources on the NIC (e.g. queues, scheduler/shaping nodes) are not sufficient to

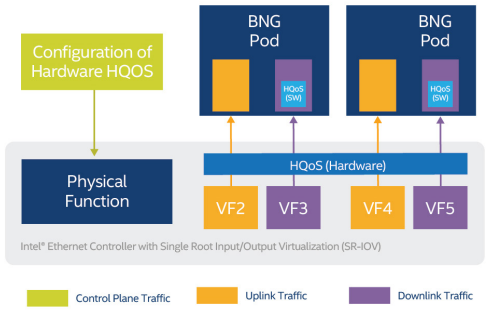

Figure 13. Hardware/Software Hybrid Model

fully implement the HQoS scheduler as shown in Figure 13. In this model, each BNG instance implements some scheduling/shaping levels of the hierarchy in software and remaining levels are implemented in NIC. Decision on dividing the hierarchy between software and hardware depends upon the number of NIC resources. As an advantage, this model allows unused bandwidth from one BNG instance to be shared among other instances.

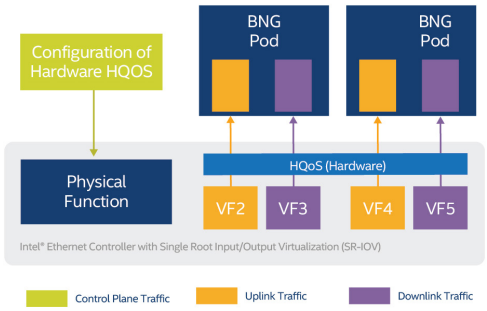

Figure 14. Full HQoS Offload Model

Full HQoS Offload Model

A full HQoS offload model requires a NIC that can support full offload of HQoS processing from all vBNG instances, as shown in Figure 14. A major advantage of this model is the CPU does not play a role in HQoS, freeing up CPU cycles for other pipeline blocks.

The proposed vBNG architecture supports all three of these models, depending on the capabilities of the underlying hardware.

Orchestration of the BNG

The vBNG architecture proposed in this paper does not limit how a vBNG instance is virtualized. For example, either full virtual machine (VM) virtualization or Linux containers can be used. When a vBNG comprises of two individual pipelines, deploying each in a separate container may/can be better as it is a light weight virtualization option, compared to deploying in virtual machines. Containers also allow for more rapid initialization and recovery of instances following the convention in network deployments of high availability. For the deployment of a vBNG instance in both the reference application and this paper, a pair of containers is launched by the orchestration engine Kubernetes. This section will further discuss how this is achieved under the umbrella convention of a Cloud-Native Network Function (CNF). The two biggest influences on the design and deployment of the vBNG is the CUPs architecture and Cloud-Native Networking. CUPS described earlier makes the control and user plane two separate entities that communicate over a set API. This section will mainly focus on how the vBNG achieves Cloud-Native Networking at high throughput rates. Also referenced in other Intel wireline papers are the following conventions that a telco application needs to abide by to consider itself a CNF:

- **Highly Performant** – The CNF must take advantage of Enhanced Platform Awareness (EPA) features to ensure low latency and high throughput.
- **Agile Placement** – The CNF must allow for flexible placement to be allowed to deploy on any EPA feature ready platform and must be generic to the EPA infrastructure provided beneath.
- **Lifecycle Management** – Using automatic telemetry aware controllers, the CNF must ensure that it can scale resources under increasing workloads and retract resources under decreasing workloads.
- **Highly Available** – The CNF must always present high availability to its required workload by engraining extreme fault tolerance into the component architecture to meet the Service Level Agreement of almost zero downtime. The CNF must also utilise the HA schema to maintain interfaces for quick and simple service upgrades without any effect on the service it provides.
- **Observability** – The CNF must ensure all network and performance metrics of workloads are exposed through an easy to consume platform allowing for rapid network debugging and modification.

Date Plane Deployment

The 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 15, the full Data Plane deployment (Not Control Plane) is separated into 3 components:

- **BNG DP Management**
  - In short, this section is seen as the interface between the Control Plane and the Data Plane in a full CUPS deployment
  - This section is responsible for:
    - Receiving Data Plane Configuration (PFCP Agent)
    - Setting and storing the Data Plane Configuration (etcd)
    - Retrieving telemetry data from the Data Plane instances
    - Managing the scale of vBNG Pods/Instances
- **BNG Data Plane**
  - Discussed previously this section is responsible for:
    - The vBNG forwarding pods Uplink and Downlink
    - The Telnet etcd agent (This agent is seen to be shared between the BNG DP Management and BNG Data Plane but physically it is deployed in the Data Plane)

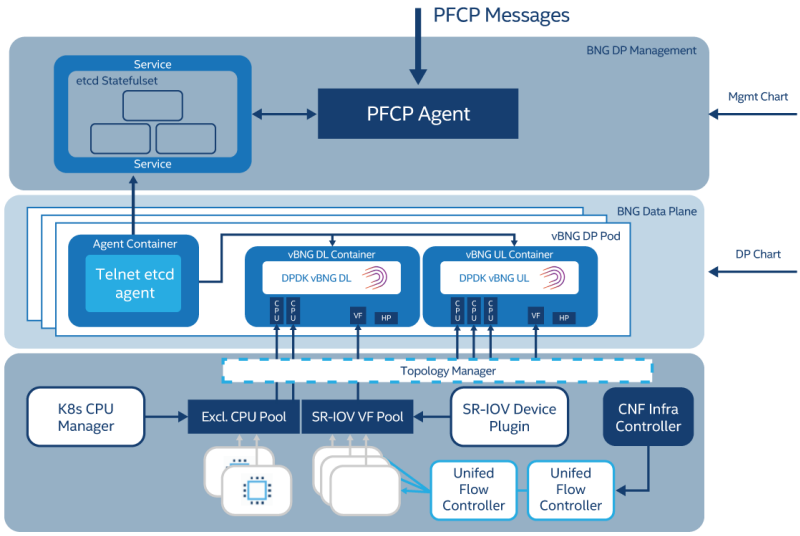

Figure 15. Full Micro Service Architecture of a BNG Deployment

- **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 K8s Kubelet (Manages container state of the Node)
    - The K8s CPU Manager(Supplies exclusive vCPUs to the vBNG Containers)
    - The SRIOV 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)
    - The Unified Flow Stack (Uses DCF and DDP to set SR-IOV Switch rules allowing scale based on subscriber header fields)

Control Plane Deployment

The Control Plane used in the BNG deployment is built by BiSDN. This follows the same micro service architecture as the other components in the BNG deployment in which each element is deployed as a container entity. For the BNG deployment, the BNG Control Plane may be deployed on the same cluster as the BNGDP Management and BNG Data Plane or in a separate remote cluster for commanding multiple BNG clusters. If deployed in the same cluster it utilizes the same container network interfaces (CNIs) as with other BNG components; see Figure 16. If the network operations engineer requires 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 Plane Cluster to ensure external BNG Control Plane access is regulated and load balanced for the DPPFCP Agent to receive and parse messages.

Deployment Overview

By combining all of the architecture proposals previously discussed, it is possible to build a scalable, orchestratable, and CUPS-enabled BNG solution that efficiently uses the I/O and compute resources of an Intel® processor-based server. This solution can help CoSPs address the need to deliver ever-increasing bandwidth at lower cost, as outlined at the beginning of the paper. Figure 17 provides a high level overview of a full CUPS deployment alongside all ingress and egress Broadband traffic.

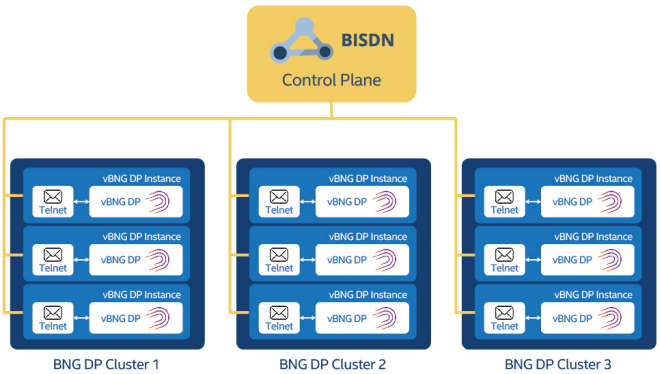

Figure 16. 1:N mapping of Control Plane to Data Plane on multiple clusters

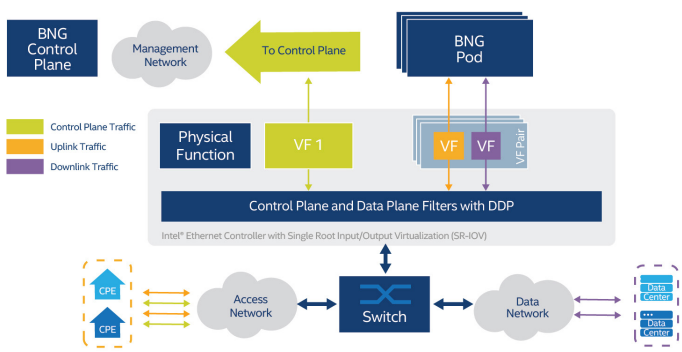

Figure 17. Scalable CUPS-Enabled BNG Architecture

Figure 18. Performance Testing Pipeline Blocks

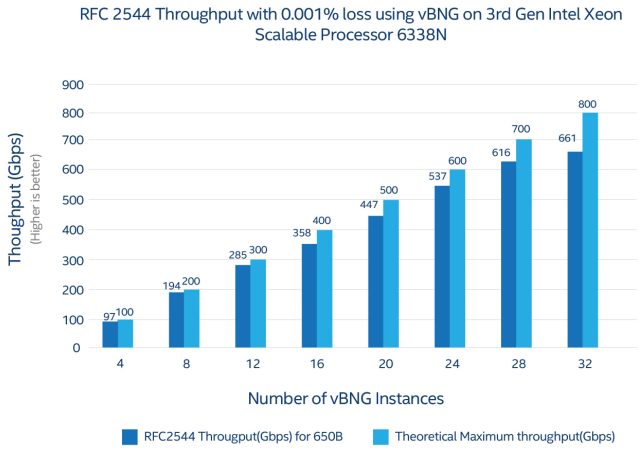

Figure 19. Intel® Xeon® Processor-Based Server (Dual Socket) Throughput Running Various vBNG Instances

Performance Benchmarking[7]

Performance measurements on the blue pipeline blocks shown in Figure 18 were taken on a dual socket server with Intel® Hyper-Threading Technology (Intel® HT Technology) enabled, Enhanced Intel SpeedStep® Technology and Intel® Turbo Boost Technology disabled. The same traffic profile was applied to all instances (4,000 flows, downlink/uplink packet size = 650B), and the cumulative throughput (downlink+uplink) across all instances was measured. The optional grey blocks were not enabled.

Figure 19 shows the throughput of an Intel® Xeon® processor-based server with two 3rd Gen Intel Xeon Scalable processors 6338N running vBNG container instances. The throughput scales very effectively as we deploy from four through thirty two vBNG instances with increment of four instances.

With thirty two instances deployed, the throughput is 661Gbps when using RFC2544 test methodology with 0.001% packet loss. This is achieved using 96 data processing cores (1.5 cores per instance for thirty two instances). All resources used by the BNG application are local to the socket. It is found to be I/O bound but not CPU bound.

Something to be noted with these results is that they were run in a "Docker Only" configuration whereby K8's was not used to scale up and down instances.

Summary

The future viability of NFV-based networking equipment running on general-purpose servers hinges on the ability to service ever-increasing traffic volume in a cost-effective manner. This paper presents architectural considerations and benchmarking data that demonstrate the huge potential for NFV-based packet processing. In addition, CoSPs can deliver network connectivity and new services from the edge of the network using a combination of BNG and service-edge solutions. By rethinking how virtualized network functions are created and deployed, new possibilities arise, such as:

1. Redefining the unit of performance from the number of VMs or containers to the number of cores that deliver a nearly linear increase in uplink and downlink throughput
2. Creating a model that is virtual network function (VNF) architecture agnostic (i.e., VM, container, or baremetal).
3. Generating a deterministic price per home model that stays predictable with the traffic CAGR.
4. Increasing network availability by converting the large monolith of systems to distributed systems, which enables CoSPs to better manage fault domains and contain affected areas, thus minimizing connection storms and outage times.
5. Creating a multifunction edge infrastructure that can address both NFV and services.

Raw BNG performance is not one single data point answer but a discussion on network location, subscriber density, average bandwidth per subscriber, and traffic CAGR over the deployment lifecycle. The vBNG architecture presented in this paper allows CoSPs to model uplink and downlink throughput and scale control and user plane function independently on general-purpose servers in a predictable and reliable way.

Appendix

vBNG Server	
Platform	Intel® Server System M50CYP Family
CPU	2x Intel® Xeon® Gold 6338N Processor, 2.2GHz, 32 Cores
Memory	16x32 GB DDR4
Hard Drive	Intel® SSD D3-S4510 Series (480G)
Network interface Card	4x Intel® Ethernet Network Adapter E810-2CQDA2 (aka Chapman Beach)
Software	
Host OS	Red Hat Enterprise Linux 8.2 (Ootpa)
vBNG	vBNG 20.11
Linux Container	Docker version 20.10.5, build 55c4c88
DPDK	DPDK-v20.11*
BIOS Settings	Bios Version: SE5C6200.86B.0020.P24.2104020811 uCode: 0xd0002c1 P-state Disabled, HT ON, C-States Disabled,Turbo Boost Disabled,SRIOV and Vtd enabled

Application Configuration per Instance	
Uplink	
Frame Size: 128B*; % of Overall Traffic: 11; 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*; % of Overall Traffic: 89;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)

*All vfs are bound to igb_uio module for application use.

**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)

1. “Cisco Visual Networking Index: Forecast and Trends, 2017–2022 White Paper,” February 27, 2019, pg.,

https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html#_Toc484813985

2. 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 only 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.

3. “Cisco Visual Networking Index: Forecast and Trends, 2017–2022 White Paper,” February 27, 2019, pg. 2.

4. “Architecture for Control and User Plane Separation on BNG"

https://datatracker.ietf.org/doc/draft-wadhwa-rtgwg-bng-cups/

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-for-telecommunications-workloads-technology-guide.pdf

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 only 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.

Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex. 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. Your costs and results may vary. Intel technologies may require enabled hardware, software or service activation.

Printed in USA 0821/BB/ICMCEL/PDF ♻ Please Recycle 340031-002US