WHITE PAPER

Communications Service Providers
User Plane Function

Towards Achieving High Performance in 5G Mobile Packet Core's User Plane Function

This paper presents the architecture for a user plane function (UPF) in the mobile packet core (MPC) targeting 5G deployments.

Authors

DongJin Lee
SK Telecom

JongHan Park
SK Telecom

Chetan Hiremath
Intel Corporation

John Mangan
Intel Corporation

Michael Lynch
Intel Corporation

Contents

Abstract

This paper presents architecture for a user plane function (UPF) in the mobile packet core (MPC) targeting 5G deployments. This paper's discussion begins by analyzing the various functions required in UPF by modeling packet streams, architectural partitioning of functionalities between software and hardware, and how various packet sizes affect the UPF's overall performance.

From the analyses, several functions are identified in which packet processing can be efficiently optimized by using various software and hardware designs. The functions are integrated to achieve a high-performance UPF architecture using standard rackmount servers and leveraging network functions virtualization (NFV) technology.

In particular, using a single unit of rackmount server, Intel and SK Telecom:

- Decomposed the user plane pipeline into smaller functional modules
- Analyzed how different partitions and pipelines affect performance of packet processing
- Iteratively developed and aggregated modules for software and hardware
- Estimated an eventual greater than 200 Gbps throughput

The implemented UPF servers are designed to be highly scalable and flexible, and can be deployed in a variety of physical scenarios, such as in core and edge telco infrastructure using NFV infrastructure (NFVI). Based on current results, telecom equipment manufacturers (TEMs) and independent software vendors (ISVs) can leverage learnings from this prototyping work to optimize and further develop solutions for high-performance commercial grade implementations.

Introduction

Exponential growth in mobile subscribers' use of wireless networks for daily activities is driven by a broad variety of services, including voice and video calling, video streaming, web browsing, P2P, SMSs, and so on. Along these lines, the insatiable desire of mobile subscribers to enrich their user experiences over time is exponentially transforming the amount of data transferred over wireless networks.

Upcoming 3GPP 5G technology is slated to be a critical enabler to dramatically enhance the variety of existing and new use cases, such as ultra-reliable and low-latency communication (URLLC), enhanced mobile broadband (eMBB), and massive machine type communications (mMTC). Some use cases drive bandwidth use (such as mobile broadband), while others are sensitive to latencies (such as gaming) or represent a large number of devices potentially periodically transferring small amounts of data (such as IoT).

In an LTE-based mobile packet core, the evolved packet core (EPC) provides functions for user equipment (UE) mobility, quality of service (QoS), billing/charging, and policy controls. EPC also provides IP-based packet data services and bridges the radio access network (RAN) with the rest of the converged voice (for example, VoLTE) and data (such as video streaming) services in the network. As such, the exploding growth in mobile bandwidth uses driven by packet data has resulted in significantly higher bandwidths that need to be delivered by the EPC. Furthermore, the bandwidth use requirements differ during various times of day, which necessitates the ability to modulate capacity as required to improve operational efficiencies of the network. Within various functions in the EPC system, Serving Gateway (SGW) and PDN Gateway (PGW) handle mobile subscribers' user packet data, known as the user plane function (UPF).

Recently, the study of Control and User Plane Separation (CUPS) architecture in 3GPP has been completed in the TR 23.714, and the specifications are finalized in TS 23.214/23.244. CUPS enables scaling of the user plane function capability by architectural separation of control and user plane functions in the SGW/PGW using an Sx interface. This separation allows the UPF to be distributed and deployed independently from the centralized control plane. For example, multiple UPF instances can be scaled flexibly, based on their workloads, interface status, and/or subscribers' capacity demands. This approach allows the UPF to be physically placed closer to the edge of the network or to be co-located with an eNodeB aggregation site within RAN.

5G mobile packet core, currently under 3GPP standardization (TS 23.501/TS 23.502), discusses UPF following a similar concept to the CUPS architecture. While specification details are a work in progress, the roles of UPF are clear—they must be able to provide a packet-based routing/forwarding, header manipulations, QoS, billing/charging, and policy controls. The descriptions of functionalities and architectural partitioning provide evidence that a highly efficient, performance optimized, and scalable UPF is critical in 5G. Further, new 5G-driven use cases targeting URLLC, eMBB, and mMTC require varying characteristics from the overall wireless network, including separation/isolation of functions to ensure performance, QoS, latency metrics.

These use cases drive the need to use a network function implementation to ensure scalability with efficient CapEx and OpEx. NFV-based virtual network functions (VNFs) are designed to enable these use cases that enable TEMs and operators to reuse software-based VNFs for deployment at various physical and logical locations. However, to realize this approach, solutions must scale and provide high performance efficiently.

To this end, this paper's primary focus is on high-performance user plane scalability, which can be a building block to enable use cases and services on top of commodity-off-the-shelf (COTS) IT servers. SKT and Intel have analyzed various functions required in UPF by modeling packet streams, resource consumptions by different software and hardware architectures, and how different sizes of packet sizes affect the UPF's overall performance. The analysis identifies several partitions and pipelines where packet processing can be efficiently optimized by using various software and hardware designs. The implementation in this work, using a single rackmount server, has achieved more than 200 Gbps throughput, approximately 70 micro-second average latency, and roughly 60 percent CPU usage.[1]

The remainder of this white paper discusses System Architectural aspects, including:

- Key tenets of a performance optimized UPF
- UPF pipeline implementation options
- Traffic profiles used to test the implementation
- Recommended partitioning between software and hardware targeting 5G UPF
- Test results and key takeaways

System Architectural Aspects

Figure 1 illustrates three network slices (eMBB, URLLC, and mMTC) driven from use cases in a 5G network. These use cases break the barriers between traditional access and core network, as specific network functions can be deployed to enable specific characteristics of the overall network use cases. The ability to deploy network functions at various

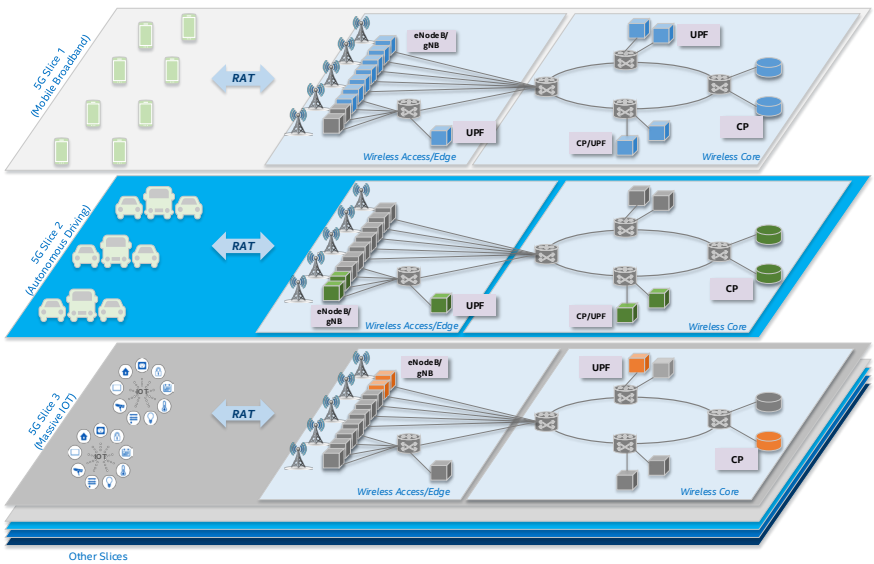

Figure 1. Network Slicing Concepts Driven by Specific Use Cases

locations as a matter of choice driven by use cases drives the need to reuse network function implementation, while at the same time enables scalability, with efficient CapEx and OpEx.

Figure 2 shows the 5G system architecture (TS 23.501/502) currently under standardization. For the UPF, 5G Core Network system architecture leverages concepts defined with CUPS in Rel-14 of the LTE specification and defines a dedicated UPF to handle all aspects of user data packet transfers. As mentioned, allowing UPF to be deployed on demand for processing various types of services without being tightly-coupled from the control plane is one of the key architecture benefits for 5G use cases. Specifically, UPF functionalities include:

- Packet routing and forwarding
- Branching point for multi-homed PDU sessions (also known as multi-PDN connectivity in LTE terminology)
- Uplink traffic validation
- External PDU session interconnect to data network
- Uplink classifier to support routing traffic flows to data network
- QoS handling for user plane
- Transport level packet marking in UL and DL
- Downlink data buffering
- Data notification triggers
- Other functionalities

In an NFVI deployment, a virtualized mobile packet core system is built around COTS IT servers. Typically, a virtualized mobile packet core has a front-end load distribution/ balancing server that terminates the interface from RAN (for example, S1-U), and distributes the traffic to one or more servers for either control plane or user plane processing. The server terminating S1-U interface provides the entry point for a packet core security domain by terminating IPSec traffic between one or more eNodeBs. After decrypting IP-Sec packets, typical methods used to distribute traffic to multiple user plane servers are based on round-robin or 5-tuple-based hash on the clear IP packet, and post IPSec decryption.

S1-U traffic entering into the mobile packet core domain from RAN is structured as shown in Figure 3. The UE IP packet is encapsulated in GTP-U protocol, which is then layered over the UDP/IP packet. The outer IP packet addresses are the eNodeB and the SGW. Since hundreds of UEs connect to a single eNodeB, a single eNodeB to SGW communication carries all the traffic from the UEs connected to the eNodeB. This results in a situation where load distribution based on outer IP header information has low entropy, since the total number of eNodeBs connecting to the SGW may be in the thousands, while the actual number of UEs communicating to the SGW could be in the millions.

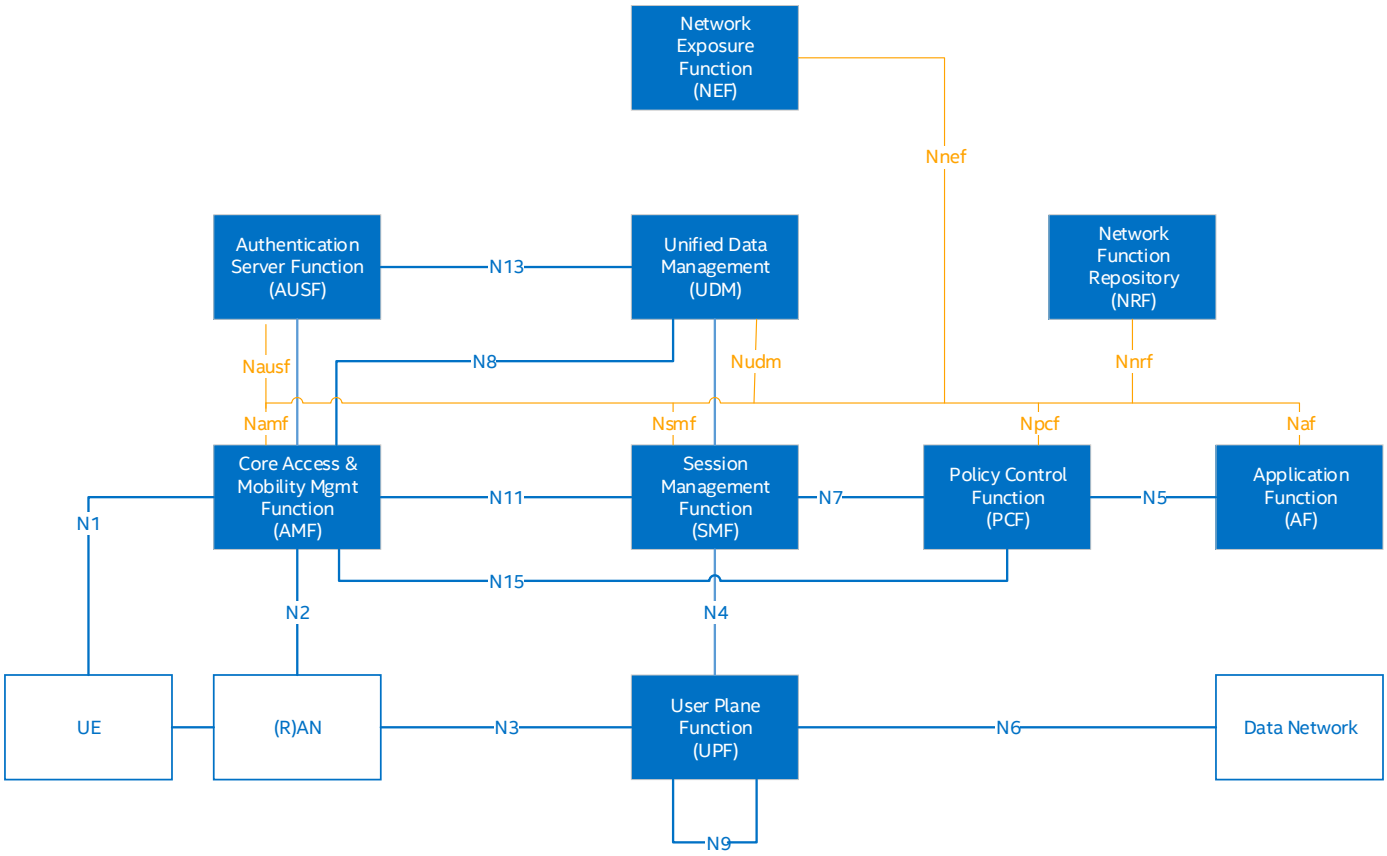

Figure 2. 5G System Architecture with Independent User Plane Function

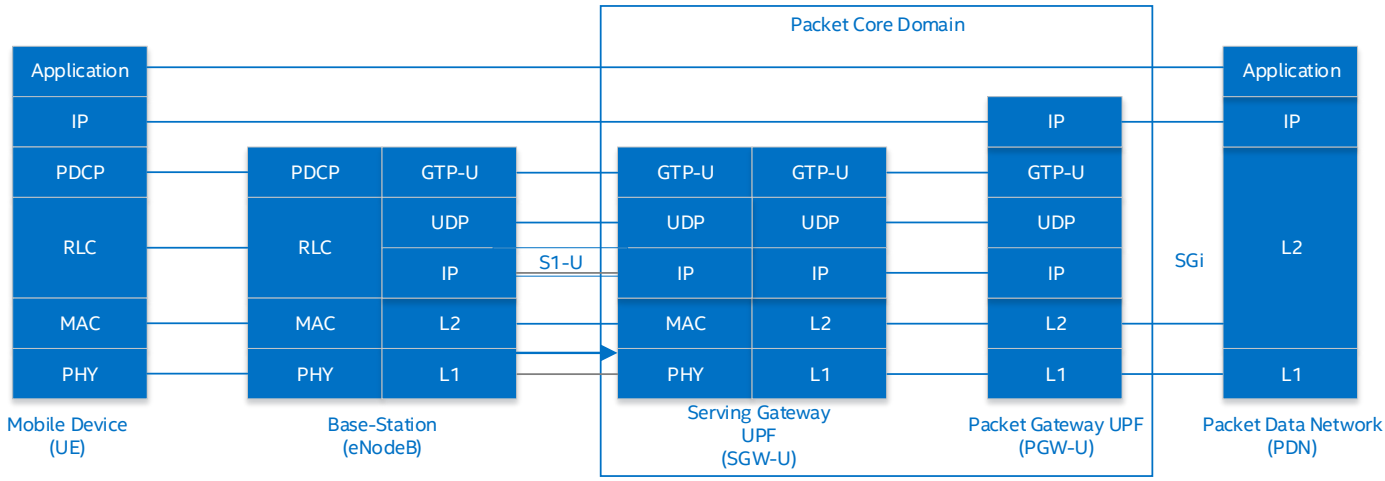

Figure 3. Protocol Structure in LTE Network

Problem Statement

User plane servers typically run one or more virtual machines (VMs), or containers. In this way, the load-distributed traffic is evenly distributed across servers. This can result in situations where UE traffic might end up on different VMs or containers on a server, or worse, on different servers, as illustrated in Figure 4.

Here, each user plane VM must determine whether the ingress packet belongs to the set of UEs it is capable of processing (based on UE context information). If the UE context information is not available, the user plane VM must:

- Identify the correct VM where the UE context already resides, and forward the packet to that VM, or
- Bring in UE context information from the control plane to process the packet successfully

An implication of the first option is the unnecessary use of IO bandwidth to forward packets to the correct user plane VM. Because VMs may be on the same physical server or on a different server, independent of where the target VM physically resides, overall system-level processing efficiency is compromised due to packets bouncing in the packet core domain. Such loss in efficiency is typically measured in terms of end-to-end packet latency increases and IO bottlenecks.

An implication of the second option is that the UE context is distributed across multiple VMs. In this case, delays are incurred when reading in-context information from the control plane. In addition, ensuring the data is correct requires cross-VM boundary-locking mechanisms to process the packets. Common context elements that would be updated across a VM boundary in this scenario includes statistics for QoS, charging, and so forth. This can result in performance losses (for example, reduced number of packets per seconds processed, increased end-to-end packet latency), due to the locking of shared resources. A larger problem with this approach is that ordering of packets in the packet core domain is not maintained, which can result in downstream impacts due to retransmissions in the overall network.

Solution

To optimally process traffic in the packet core domain, SKT and Intel propose that the packet must have a single hop to the user plane VM. As such, the server load balancing S1-U ingress traffic into the packet core domain must include two key elements. First, it must have the intelligence to include attributes from a certain number of eNodeBs. Second, information from inner IP packets must be included to determine load distribution and balancing decisions at runtime.

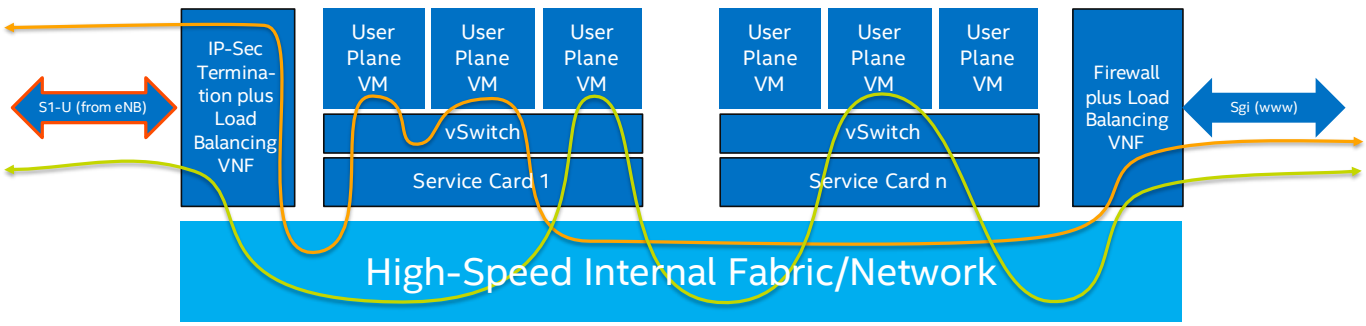

Figure 4. Typical Load Distribution in Virtual EPC System

The intent of the load distribution is to direct all traffic of a UE communicating via a given base-station to a known VM and, within the VM, to a known processing core (also called worker core/worker thread). Essentially, this pins UEs to cores, which enables optimal performance at the system level by reducing the amount of data shared across VMs or cores in a VM, and maximizes the use of on-CPU caches. This avoids packet bouncing in the packet core domain, which results in the lowest latency, less jitter, and the highest degree of packet throughput. This method is illustrated in Figure 5.

Along with optimal load distribution at the system level, packets coming into the physical interface of the user plane server also need to be appropriately steered toward the correct VM and worker core in the VM, as shown in Figure 6. This is achieved by intelligent steering of packets in hardware by classifying the packets as GTP-U, picking correct fields from inner IP packet (primarily the Src-IP address that identifies the UE), and hashing on it to identify the correct worker core. This method ensures that all traffic for a UE (with a given IP address) is always steered to the same worker core. As part of the prototyping work, the implementation was initially done in software and then moved to hardware.

This method of steering packets to cores is implemented for ingress traffic on both S1-U and SGi interfaces, resulting in optimal data flow in the packet core domain. This enables the maintenance of packet ordering in the packet core domain while maximizing user plane processing, with the goal of targeting lowest end-to-end latency and jitter, and highest packet processing throughput (and hence bit rate).

The resulting overall system architecture is one in which a set of eNodeBs are configured to use a given VLAN/VxLAN/MPLS tunnel to send/receive its S1-U traffic. Simultaneously, another VLAN/VxLAN/MPLS tunnel is used to send/receive S1-AP traffic, which is directed to the control plane "cluster." This mechanism is shown in Figure 7. Load balancing of a given set of eNodeB traffic is set to user plane VMs, while at the same time provides the ability to differentiate MVNO traffic from another set of user plane VMs.

User Plane Function

The user plane function handles the critical data plane processing of packets between radio access network (RAN) and the data network (for example, Internet). As such, the UPF provides functionalities such as:

- Access Control
- GTP-U tunnel encapsulation
- Decapsulation
- Bearer lookup
- Service data flow (SDF) mapping
- Per-flow QoS (such as handling of QCI performance characteristics)
- Guaranteed bit rate (GBR)
- Maximum bit rate (MBR)
- APN level aggregate Maximum Bit Rate (APN-AMBR)
- Charging (online/offline charging, enforcement of charging policies)
- Forwarding of packets to/from packet data network

While the 3GPP specification defines functionalities associated with the UPF, practical implementations also require flexibility to customize functionalities to comprehend operator-specific requirements driven by use cases or regulatory requirements. These additive requirements also tend to be geography-specific (for example, regulatory requirement driven charging customizations). Therefore, the UPF must support flexible implementations to be amenable for customizations.

NFVI deployments also require layer-2 overlays, including VLAN, or layer-3 overlays, such as VxLAN/GRE/MPLS. NFVI deployments must support virtualizations, and they require the ability for VNFs to communicate while providing isolation capabilities in a multi-tenant environment. These capabilities

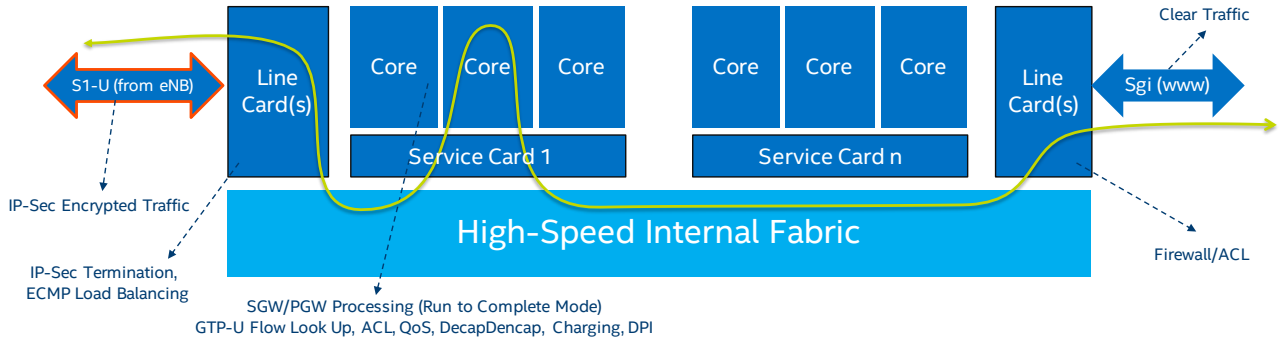

Figure 5. Packet Stream in Optimal System Architecture

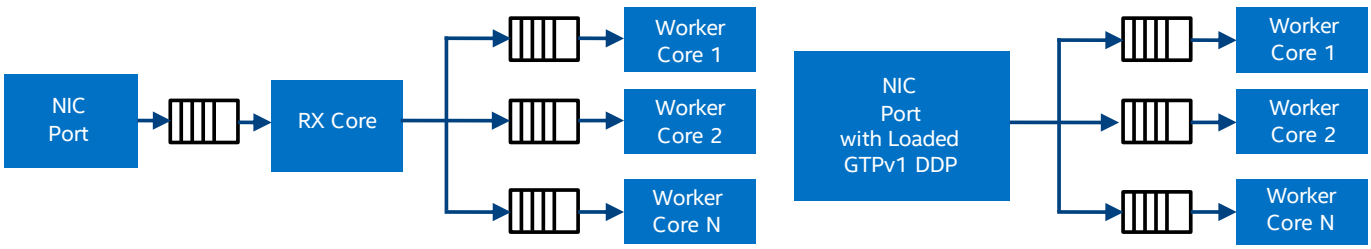

Figure 6. Steering of Packets to Worker Cores

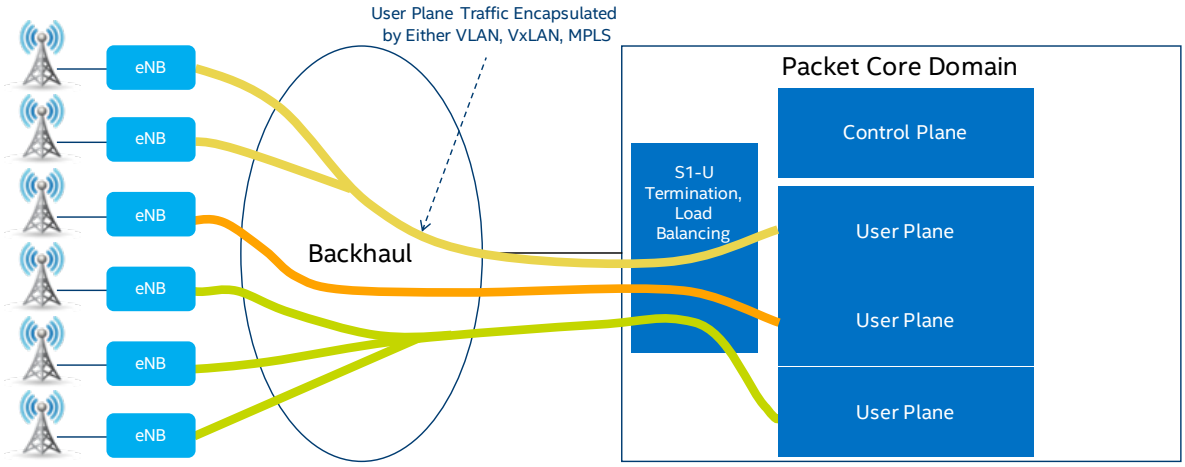

Figure 7. Overall Network Topology with Traffic Movement

enable uses such as MVNO or network slicing to isolate traffic between different slices driven by business requirements.

NFVI and UPF should be interworked together to produce a balanced architecture to increase performance and flexibility. The following list presents a few implementation options for a UPF pipeline:

- **Software-based with UPF pipelining across cores:** Each stage of the user plane pipeline (for example, bearer mapping, charging, QoS) individually executes on a worker core chained together with software-based queues to construct an end-to-end pipeline. User plane processing of a packet is partitioned per stage on each core.
- **Software-based with run-to-completion (RTC) UPF pipeline:** All stages of the user plane pipeline execute sequentially on the worker core. Hence, each packet's user plane processing is executed in entirety, followed by the next packet picked for processing.
- **Hardware-assisted UPF pipeline:** The user plane pipeline (or portions of it) is implemented in hardware (for example, ASIC, NPU, or FPGA).
- **Hardware-assisted with SDN switch UPF pipeline:** OpenFlow*/P4-based SDN enabled top-of-rack (ToR) switch could potentially implement portions of the pipeline (for example, forwarding, and so forth).

Implementing the Software-Based UPF Pipeline in Run-to-Completion (RTC) Mode

SKT and Intel chose to implement the software-based UPF pipeline in run-to-completion mode for the following architectural reasons:

- UPF inherently has a substantial number of subscribers and number of flows associated with each subscriber.
- Processing of packets from each subscriber is largely independent from other subscribers (for example, charging, QoS, bearer look ups, SDF matching, and so forth), which lends itself well to process packets in RTC without the need to partition processing into smaller stages that would then be spread across cores (as with pipelining across cores).
- The RTC case allows for localizing frequently used data structures (for example, subscriber context) to local caches associated with the cores. As such, minimal locks for shared structures are required across cores. This translates to overall lower per-packet processing latency and higher overall system level throughput.
- A software-based implementation provides the highest degree of flexibility to quickly add and/or customize operator or geo-specific functionalities with relative ease, compared to other options considered.
- A software-based implementation enables portability to deploy UPF at the location of choice using standard/ commercial off-the-shelf server infrastructure (for example, edge or core locations).
- The implementation provides the ability to scale out performance in a given form factor, based on the operators' deployment needs, because the software is built to be independent of underlying platform hardware configurations.

Implementing the LTE EPC Stack

The LTE EPC stack used for the prototype implementation detailed in this white paper is based on a commercial implementation from ASTRI. This software stack includes SGW, PGW, and MME functionalities. Refer to www.astri.org for more information.

Implementing Hardware

Along with software-based pipeline implementation on CPU cores, SKT and Intel leveraged a hardware-assisted NIC to distribute packets to various worker cores on the platform. This NIC also assisted the NFV infrastructural capabilities that are application-independent to benefit all categories of NFV applications deployed on a COTS platform, offloading functions including:

- **Virtual switching:** Use for intra-VM interactions when co-located on the physical server (for example, control plane and user plane VMs). Also, assists in deploying, orchestrating, and managing VMs on NFV platforms.
- **Tunnel endpoint termination for network overlays:** VLAN, VxLAN/GRE tunnels terminate in hardware as part of ingress/egress processing.

Based on the partitioning described, the overall hardware and software partitioning of the user plane pipeline on a COTS platform is illustrated in Figure 8.

This COTS-based UPF platform comprises network interfaces with multiple 25 GbE ports, which provide connectivity to servers that terminate S1-U or SGi interfaces. The hardware-assisted NIC offloads tunneling endpoint termination (for VLAN, VxLAN/GRE), virtual switching, and load distribution/balancing of ingress packets to user plane processing cores in one or more VMs or containers deployed on the CPU.

Implementing User Plane VMs

In the prototype implementation, the CPU subsystem comprises a Linux*-based host operating system running a KVM hypervisor on which user plane VMs are deployed. Each user plane VM is identical in functionality and configured to run on a specific set of cores on the platform. Each VM runs multiple user plane threads. The threads are tied to logical cores. Each UP thread executes a complete instance of the user plane pipeline, as shown in Figure 9.

To optimize for performance and latency, SKT and Intel implemented the following host and platform-level optimizations:

- Host housekeeping tasks affinitized to logical core0, IRQs affinity aligned to core0, and all other cores isolated from the kernel scheduler for application execution.
- Application execution is made predictable and jitter is reduced as much as possible:
  - CPU frequency governor set to Performance mode
  - Minimal services enabled on OS as needed
  - Watchdog and CPU stall detection threads located on core0
  - All existing tasks affinity set to core0, where possible
  - Unused cores powered down through system interface.
  - Sufficient 1G huge pages allocated per socket during boot time (for the VMs to use local memory per socket).

Further, each VM was optimized for performance/latency with the following:

- Per VM resources (including CPU, Memory, and PCIe NIC ports) localized to the socket to take advantage of CPU cache, local memory channel BW.
- Bind virtual-CPU threads to specific logical cores.
- Each VM's memory configured from pre-allocated 1G huge pages from host to guarantee physically contiguous memory for each VM for the huge pages used in DPDK-based UPF applications.
- VM housekeeping tasks bind to VCPU0. IRQs affinity forced to virtual-CPU0. Watchdog and CPU stall detect threads limited to virtual-CPU0.
- VMs started with minimal services and kernel configurations, as needed.

In addition, studies analyzed impacts of sharing VM housekeeping cores on a minimal set of logical and physical cores to reduce any packet loss, control plane/user plane interactions.

The overall mapping of virtual machines, VM housekeeping cores, host operating system on the dual socket COTS platform used in the performance and latency tests are shown in Figure 9.

Notice that socket-0 has four VMs, while socket-1 has six VMs. This method localizes the physical network interface port to the CPU, where its data is processed on its associated VM. Each VM is configured with a 1 x 25 GbE interface. As such, the COTS platform has 10 x 25 GbE ports on the system.

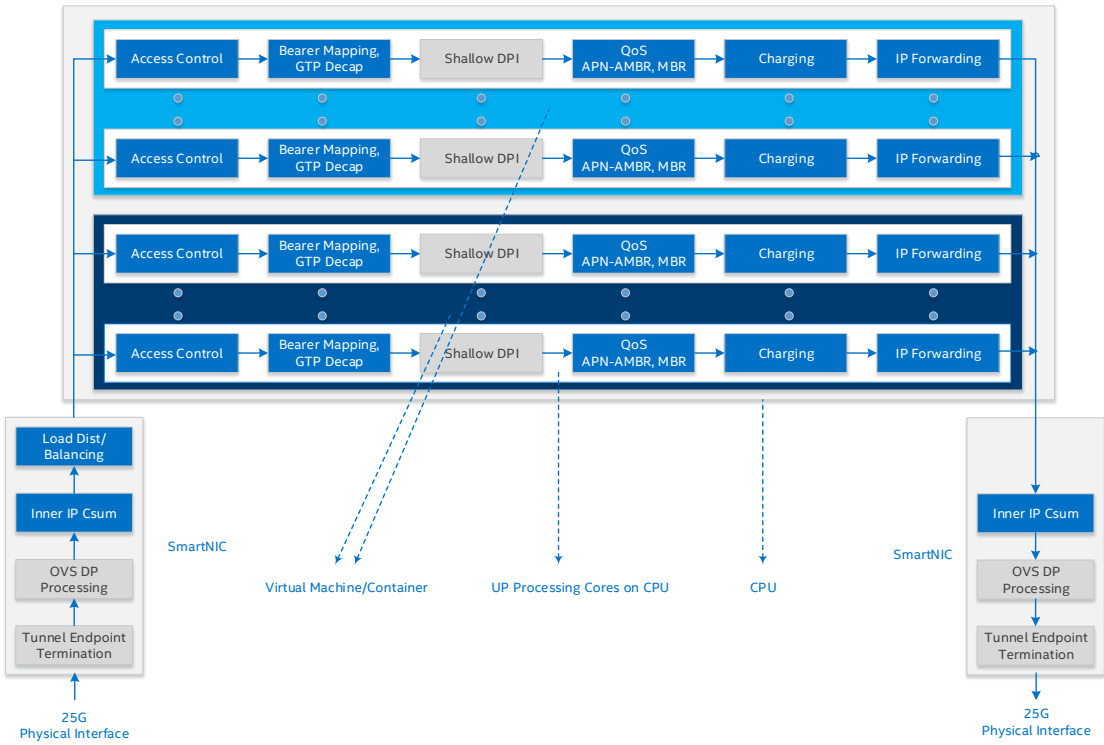

Figure 8. User Plane Processing Pipeline

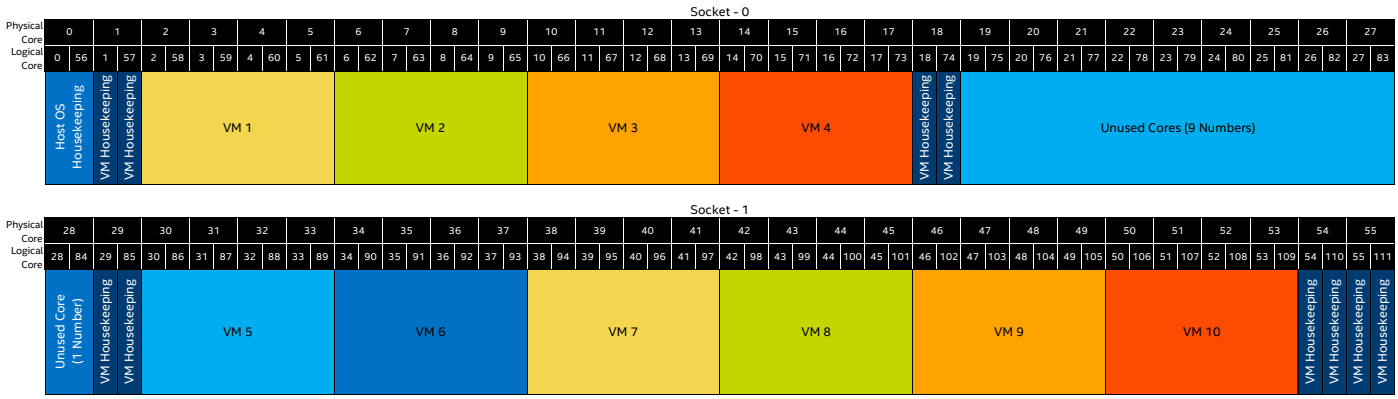

Figure 9. Mapping of Virtual Machines to Physical and Logical Cores

Traffic Profiles

Performance depends on packet sizes and their ratios. The UPF must receive realistic mobile traffic data to ensure the developed system is robust against variability of packet sizes.

The traffic dataset used in this analysis originated from one of SKT's commercial vPGW systems characterized over a period of 25 days. Initially, SKT measured the packet size distributions for both UL and DL traffic. SKT then modelled different sets of UL and DL ratios to account for different service types.

Figure 10 and Figure 11 show a sample of the user plane traffic from one of SKT's vPGW systems. With subtle difference between the user planes, overall traffic loads are distinguishable between Busy/Non-Busy hours. Generally, the UL:DL ratio for bytes came in around 1:9, while packets were 2.5:7.5. SKT also used distribution bins to categorize packet sizes from 64 bytes to 1500 bytes, and the model took the values to capture lower and upper bound to account for confidence bands, and finally scaled to target byte and packet rates.

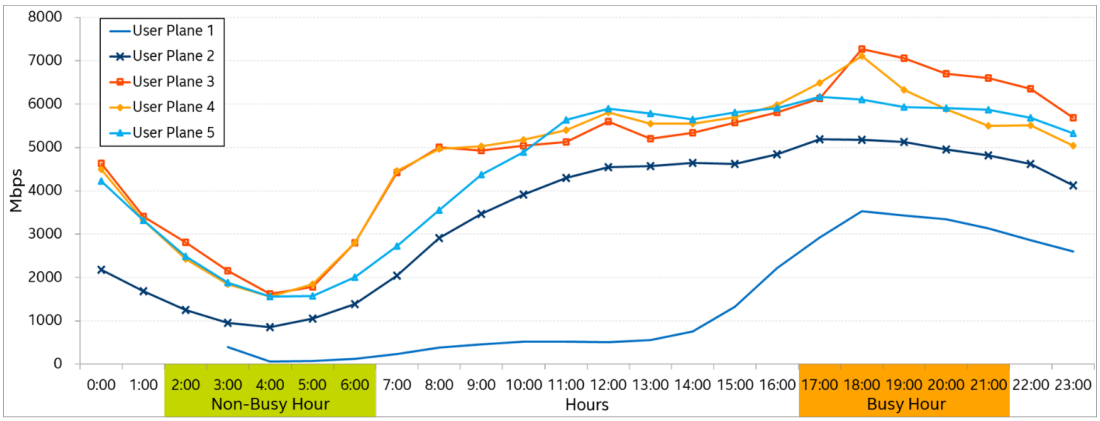

Figure 10. Sample User Plane Traffic Rates from vPGW System

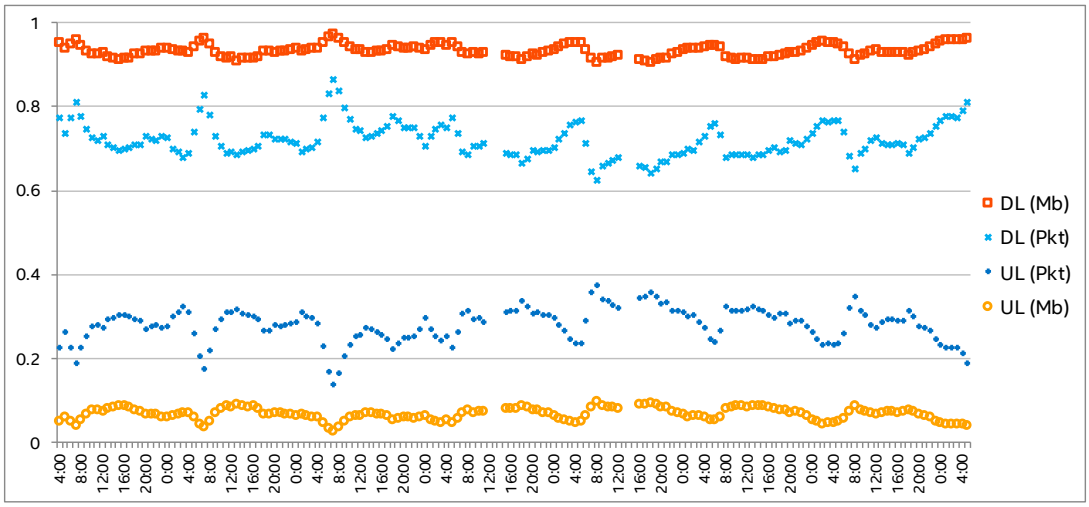

Figure 11. UL:DL Ratio for Bytes and Packets

Based on the measured packet distribution and throughput data, and looking forward to 5G requirements, the following four profiles were chosen for test purposes:

- **Profile 1: 4.2:5.7**—42.5 percent uplink throughput, 57.5 percent downlink throughput
- **Profile 2: 4:6**—40 percent uplink throughput, 60 percent downlink throughput
- **Profile 3: 2.5:7.5**—25 percent uplink throughput, 75 percent downlink throughput
- **Profile 4: 1:9**— 10 percent uplink throughput, 90 percent downlink throughput

The charts shown in Figure 12 and Figure 13 show the uplink, downlink packet, and throughput distribution.

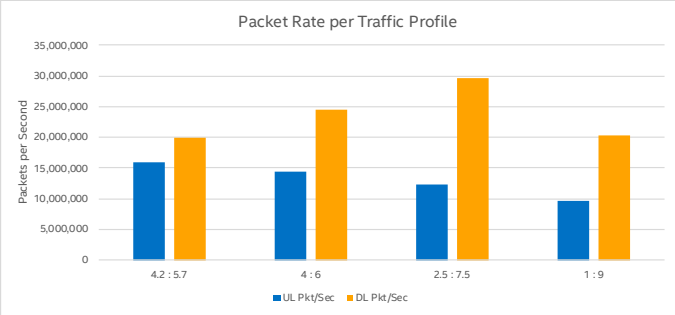

Figure 12. Packet Rate for Traffic Profiles Analyzed

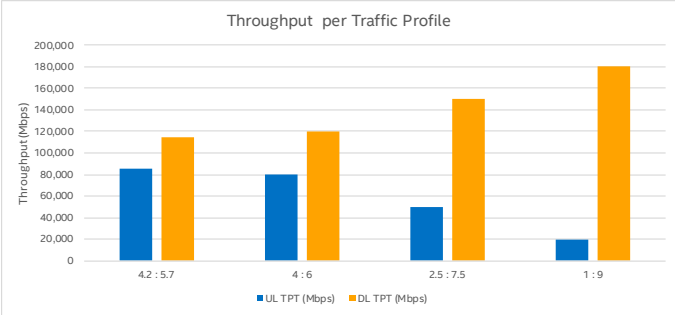

Figure 13. Bitrate Throughput for Each of the Traffic Profiles

For each of the four scenarios, the following configurations were used on the user plane:

- 100,000 mobile subscribers in connected state
- 2 Mbps throughput per subscriber, for aggregated throughput target of 200 Gbps
- Two bearers per mobile subscriber—dedicated bearer with guaranteed bit rate configuration, and default bearer with maximum bit rate configuration enforced in user plane processing
- Four flows per bearer, for a total of eight flows per mobile subscriber; resulting in 800,000 total flows (400K in uplink, and 400K in downlink)
- Access Control List (ACL), Charging included in user plane processing; all packets pass ACL/QoS/charging policies
- Packet size distribution with a mix of 64B, 128B, 256B, 512B, 1024B, and 1400B

The user plane pipeline implementation supports mobility, downlink data notifications, fragmentation and reassembly, NAT, event generation for charging data record (CDR) generation in control plane. For this white paper however, these functionalities were not exercised in the tests for throughput measurements. Shallow DPI is a functionality being considered for future studies.

Profile 1: 4.2:5.7

In Profile 1, 42.5 percent of the overall throughput is targeted in the uplink direction, while 57.5 percent of overall throughput is in the downlink direction.

The charts shown in Figure 14 and Figure 15 detail the packet distribution and throughput distribution by packet sizes in uplink and downlink direction exercised for this test.

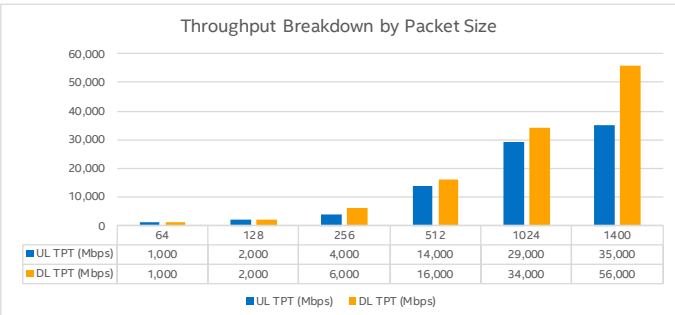

Figure 14. Bit Rate Throughput for 4.2:5.7 Profile

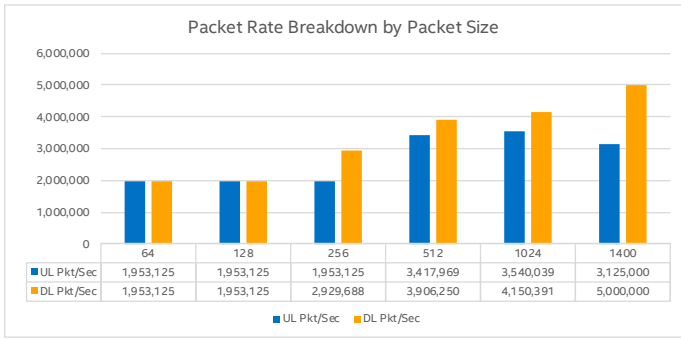

Figure 15. Packet Size Distribution for 4.2:5.7 Profile

Profile 2: 4:6

In Profile 2, 40 percent of the overall throughput is targeted in the uplink direction, while 60 percent of overall throughput is in the downlink direction. The charts shown in Figure 16 and Figure 17 detail the packet distribution and throughput distribution by packet sizes in uplink and downlink direction exercised for this test.

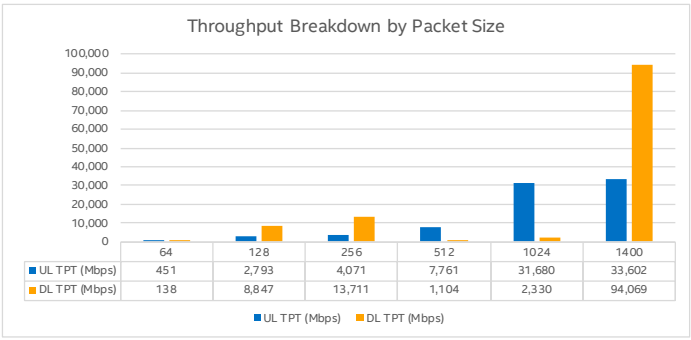

Figure 16. Bit Rate Throughput for 4:6 Profile

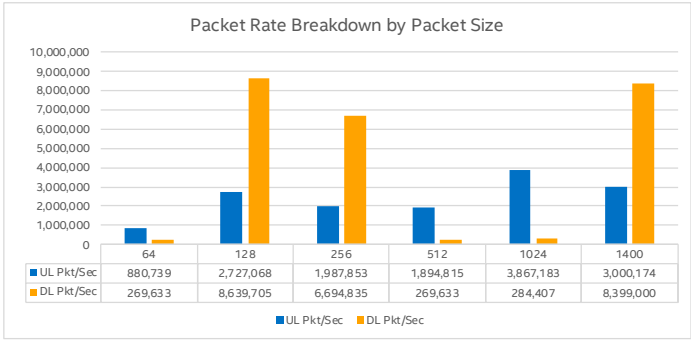

Figure 17. Packet Size Distribution for 4:6 Profile

Profile 3: 2.5:7.5

In Profile 3, 25 percent of the overall throughput is targeted in the uplink direction, while 75 percent of overall throughput is in the downlink direction. The charts shown in Figure 18 and Figure 19 detail the packet distribution and throughput distribution by packet sizes in uplink and downlink direction exercised for this test.

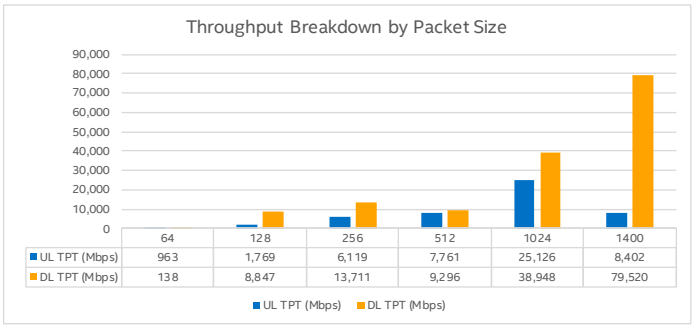

Figure 18. Bit Rate Throughput for 2.5:7.5 Profile

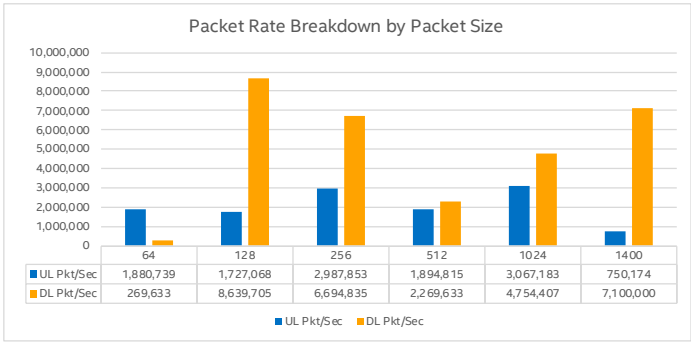

Figure 19. Packet Size Distribution for 2.5:7.5 Profile

Profile 4: 1:9

In Profile 4, 10 percent of the overall throughput is targeted in the uplink direction, while 90 percent of overall throughput is in the downlink direction. The charts shown in Figure 20 and Figure 21 detail the packet distribution and throughput distribution by packet sizes in uplink and downlink direction exercised for this test.

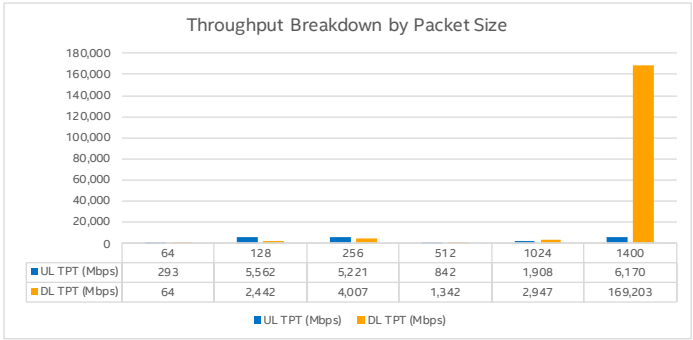

Figure 20. Bit Rate Throughput for 1:9 Profile

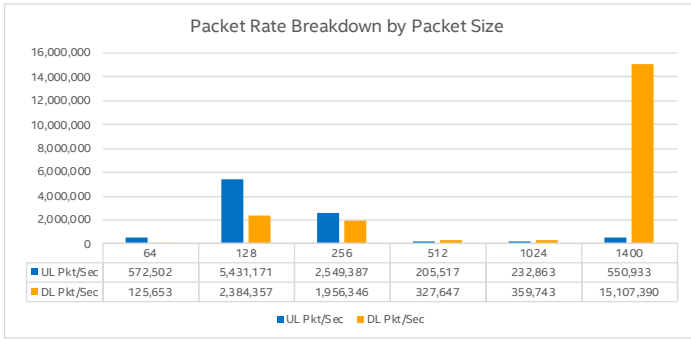

Figure 21. Packet Size Distribution for 1:9 Profile

Designing 5G Core Network User Plane Function

3GPP currently defines overall architecture and specifications for the 5G Core Network protocol. Figure 22 illustrates the protocol stack for the UPF. The N3 and N9 interfaces are expected to be based on GTPv1-U over UDP/IP to implement 5G UP encapsulation.

Given the criticality of deploying 5G services in the network infrastructure, NFV-based implementations that can be deployed in the Telco Cloud/Data Center enable optimal performance, while at the same time provide flexibility, scalability, and portability. Figure 23 shows the optimal partitioning of functionality between software and hardware.

SKT and Intel envision that a 5G user plane function pipeline is implemented as software executing on the CPU, while NFV infrastructural functionalities can be accelerated through a hardware-assisted NIC. This partitioning enables excellent performance. It separates the application from the infrastructure. For example, control plane functions like AMF and SMF can take advantage of this separation. Also, various operator-required features like DPI, content caching, and Gi-LAN services can take advantage of this. This provides a broader scale of homogenous hardware platforms to support multiple types of services.

5G User Plane Functions in Software

As discussed in this document, for the LTE user plane pipeline, the following 5G UPF capabilities can be implemented in software with the RTC model, enabling excellent performance, as follows:

- External PDU session point of interconnect to data network
- Packet routing and forwarding
- Packet inspection and user plane part of policy enforcement
- Lawful intercept
- Traffic use reporting, including customizations that may be geographic or subscriber specific
- Uplink classifier to support routing traffic flows to a data network
- Branching point to support multi-homed PDU session
- QoS handling for user plane
- Uplink traffic verification
- Transport level packet marking in the uplink and downlink
- Downlink packet buffering and downlink data notification triggering

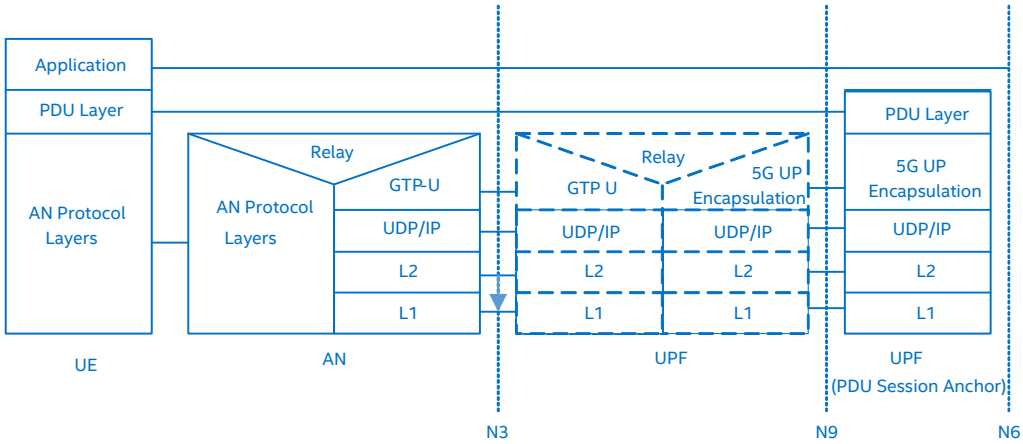

Figure 22. 5G UPF and User Plane Protocol Structure with Access Network (Source: 3GPP Specs)

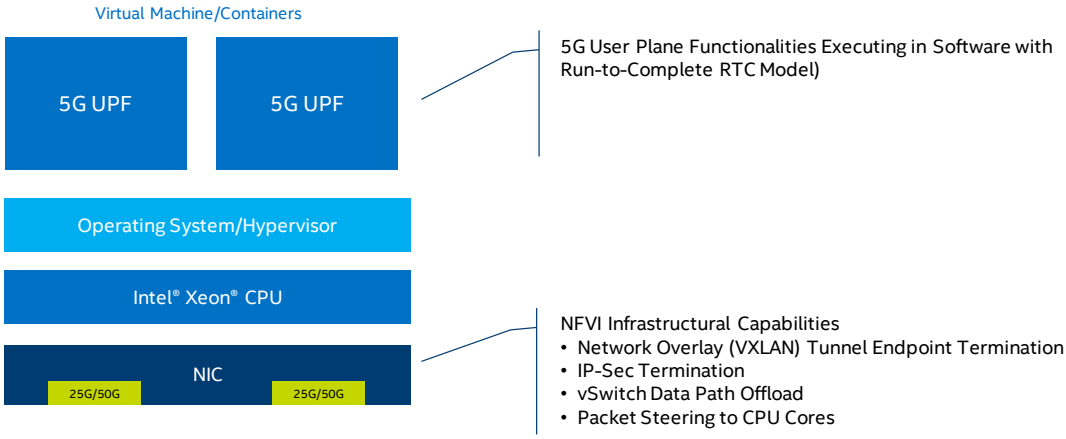

Figure 23. Partitioning of Functionalities between Hardware and Software to Implement 5G UPF

In addition, a 5G core network user plane function may also require functionalities such as DPI (either full or shallow), IP-Sec termination for the N3 interface, system level load distribution/balancing to steer N3/N6/N9 traffic to one or more UPF instances, which can be efficiently implemented leveraging VNF-based implementations.

Hardware Assists in Network Interface Controller

Hardware-assisted network interface controllers (NICs) are used to support the UPF software implementation. These provide a high-speed packet interface and a programmable hardware environment, for example, FPGA, to assist in scaling and optimizing the overall platform performance. Several features are required to maximize performance of the overall platform infrastructure. These features are flow-aware load balancing, overlay network termination, vSwitch acceleration, abstracted host interface, and IPSec termination, when used. The following section describes the motivation and design of these features based on an FPGA NIC reference implementation.

Load Balancer

A key component to attaining high performance of the UPF is the flow aware load balancing implemented on the FPGA NIC. To illustrate the requirement for a smart load balancer, Figure 24 shows the simplest form of network controller where the complete Ethernet flows are mapped to a single queue descriptor pair per physical port. This requires software to begin the processing of all packets of a complete Ethernet port from a single core. At a minimum, this core would need to determine which worker core would need to process the packet, maintain ordering of individual UE flows and possibly determine the destination VM itself. Even using poll mode drivers with DPDK, this risks becoming a bottleneck to the overall system, limiting scaling and adding latency.

Fortunately most NICs today support a mapping of Ethernet ports to multiple queue descriptor pairs, as shown in Figure 25. This allows software to process packets using multiple cores in parallel. A technique called receive side scaling (RSS) is used to choose which packet goes to which queue. It is based on calculating a hash from a subset of the packet fields and using the hash to choose which queue to use. Unfortunately, most NICs today don't have the capability to map 5G UPF flows to multiple queues that would be friendly to a scalable high-performance implementation and preserve correct flow ordering.

To better understand the challenge, Figure 26 shows the typical makeup of packets coming to the UPF from the eNodeB. An outer IP header provides the eNodeB to SGW connection, while the actual packet from the UE destined for the internet or other network is embedded in a GTP-U tunnel.

To get the necessary entropy to preserve ordering of flows it is necessary to perform RSS deep within the header and comprehend at least the TEID and UE IP Address in the hash calculation. Standard NICs and even OpenFlow based SmartNICs don't have the built in capability of recognizing and parsing within the GTP tunnel.

The FPGA based NIC has been configured to parse on both the outer header and inner GTP tunnel allowing full flexibility for RSS to load balance on individual UE flows, thus

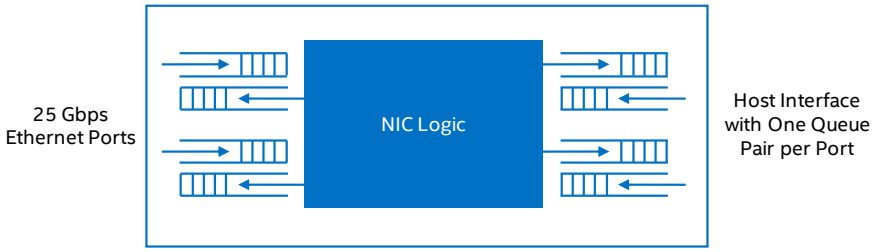

Figure 24. Simple Network Interface Controller

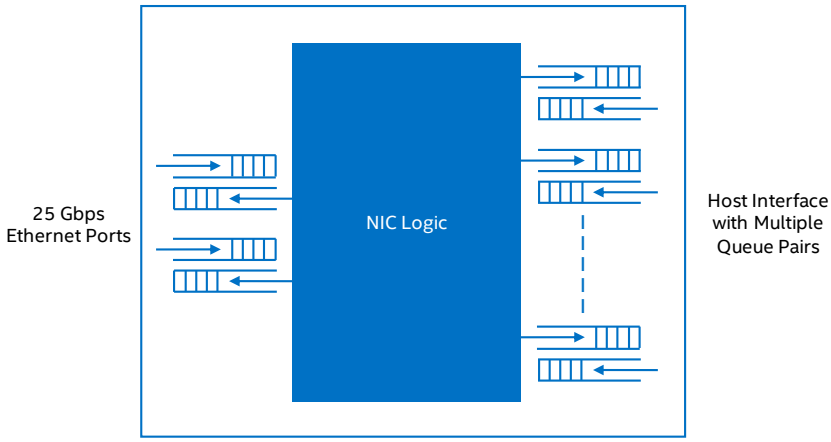

Figure 25. Network Interface Controller with Multiple Queue Pairs

SA(eNB)	DA(SGW)	Prot	UDP	GTP(TEID etc.)	SA(UE)	DA(Internet IP)	Prot	Rest of Payload

Figure 26. Basic User Plane Packet Format from eNodeB to Serving Gateway

preserving packet ordering. This eliminates any requirement for a SW balancer core and allow worker cores to scale linearly as capacity and performance requirements grow.

Network Overlays

With the advent of massive virtualization as part of the ever-growing Cloud and NFV deployments for telecommunication infrastructure, network overlays have become central to software-defined networking. Network overlays provide a means to create many virtual networks that can coexist on the same physical network. Virtual network endpoints do not require physical adjacency, and virtual networks can be completely reconfigured without impacting the physical network.

The concept is straightforward—a virtual network is assigned a virtual network ID (VNID), and any traffic from the virtual network is encapsulated with a new IP header containing the VNID and routed in a traditional L3 manner. Encapsulations with different VNIDs never see each other, and many virtual networks may coexist on the same infrastructure. These tunnels may be optionally encrypted to provide additional isolation and security. The actual tunneling protocol and function are not part of the virtualized networking application. Instead, the infrastructure provides the VNID mapping capability and the encapsulation/decapsulation function. In many cases, this role can be filled by hardware to minimize any impact on the application itself.

Two of the common network overlay tunneling protocols are VxLAN and NVGRE. Both work in a similar manner as described. For the test setup, the tunnel endpoint capability with complete encapsulation/decapsulation of the VxLAN overlay is implemented in the hardware assisted NIC (such as an FPGA-based NIC). Figure 27 shows the format of a VxLAN encapsulated packet. The outer header uses a standard L2/IP/UDP format. The standardized UDP destination port of 4789 indicates a VxLAN protocol, and the VxLAN header including the VNI follows. This then encapsulates the entire original packet, including the L2 information, effectively allowing fully distributed L2 connectivity across the virtual network.

In this implementation, a table is configured with the VxLAN header data including VNI and the mapping information to the load balancer associated with a particular virtual machine. For the test setup, each VM is considered part of its own virtual network. Other mapping schemes could be considered based on deployment and Virtual Network Function design.

Overall, the implementation demonstrates use of a VxLAN network overlay to support the scalable use case and not impact the overall system performance.

Virtual Switching

Virtual switching is the other function that can benefit from support by the FGPA-based NIC. Virtual switching is part of the network infrastructure that provides the capability of intelligent communication from VM to VM and from VM to physical ports. At its most basic form, it's a virtual L2 switch, but it can be much more sophisticated, working also in the L3/L4 domains and providing capabilities such as NAT, TEP, and QoS. In many cases, virtual switching is performed solely in software. For high performance, high throughput systems, such as the 5G CN user plane, the overhead of software virtual switching may be significant, depending on the actual virtual switching required. Accelerating some or all of the virtual switching in the NIC, for example, an FPGA-based NIC, can reduce software overhead, particularly in the domain of lookups and performing actions, such as encapsulation and address translation.

Host Interface

In an NFV Cloud environment, a full abstraction is required between the NFV infrastructure and the VNFs being hosted so no hardware dependencies exist between the two. This enables support for features such as live migration and the independent evolution of each. In this environment, the network interface abstraction is typically achieved using standardized para-virtualized drivers, such as Virtio* net. The principle drawback here is performance, due to a large overhead in the host OS to manage both the physical network interface, the para-virtualized backend interfaces, and the set of data and descriptor copies, translations, and interrupts. The traditional alternative is to use the SR-IOV model, in which a virtual function from the network device is passed directly to the VNF. This meets network IO performance but breaks the NFV Cloud model and can only be used in appliance-type deployments.

The solution proposed here leverages the flexibility of the FPGA-based NIC and uses a combination of hardware and software optimizations to achieve SR-IOV-like performance for the NFV Cloud deployment. The FPGA-based NIC supports acceleration of the backend of a standard para-virtualized driver and implements a host interface in which data and descriptors don't need to be translated and can be shared directly with the VM driver, with no copies required. The network physical function is not passed directly to the VM but still owned and controlled by the host OS. Only the memory is shared. The host OS is in control, and the host can attach and detach the physical device and support features, such as live migration between both the FPGA-based NICs and between the FPGA-based NIC and a software only para-virtualized backend. In all cases, the standard VNF driver remains the same.

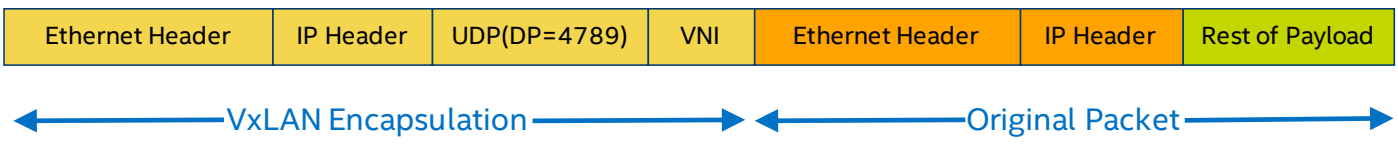

Figure 27. VxLAN Encapsulated Packet

IPSec Termination

Transport Security in the form of IPSec or equivalent is now commonplace in networking to ensure security-enabled isolation between virtual networks, particularly in a multi-tenant environment. Implementing IPSec in the FPGA-based NIC provides two benefits. First, it's a relatively cycle-intensive task on a CPU, and a hardware implementation can reduce the overall number of CPU cores required. Second, an IPSec implementation in the FPGA-based NIC allows inline packet processing of a decrypted packet and enables hardware-based load balancing/distribution capability on decrypted packets.

Performance Results

Device under Test Configuration

User plane performance measurements use a 1U COTS dual socket server. Table 1 shows key specifications of the COTS IT server used as DUT, and Figure 28 shows the photos of the server as a prototype.

Category		Description
Processor	Product	Intel® Xeon® Platinum 8180 processor
	Frequency	2.5 GHz
	Cores per processor	28 cores/56 hyperthreads
Memory	DIMM slots per processor	6 channels per processor
	Capacity	192 GB
	Memory speed	2667MHz, DDR4
Network	Number of ports	10 x 25 GbE
1U Server	Vendor	HPE* ProLiant DL360 Gen 10
Host OS	Vendor/version	Wind River* OVP6
KVM	Vendor/version	CentOS*-6.8 generic cloud image

Table 1. Specifications of the COTS IT Server Used

Figure 28. Photos of the COTS IT Server

Test Infrastructure

Figure 29 shows the overall structure of the setup used to exercise various traffic profiles to measure user plane performance and latency. The device under test (DUT) system running user plane VMs is interconnected with traffic generators (source, sink) and control plane functions (SGW-C/PGW-C, MME) via an Ethernet switch. Spirent Landslide* systems emulate eNodeB and mobile subscribers:

- The Spirent Landslide system emulates all 3GPP defined flows related to S1-U, S1-AP interface to setup PDU sessions from each of the emulated mobile subscribers, modification of PDU sessions to establish bearers, and initiation of uplink traffic.
- S1-U & SGi interfaces are terminated on the device under test (DUT) system, where vEPC user plane processing executes.
- Uplink traffic generated by Spirent Landslide system ingresses into the DUT is processed in user plane VMs as uplink traffic and egresses to UL sink machine, that acts as terminating point for uplink traffic, and is also the source for downlink traffic sent to the DUT, that is processed as downlink traffic in User Plane VMs and egresses towards DL sink machine which acts as terminating point for downlink traffic.

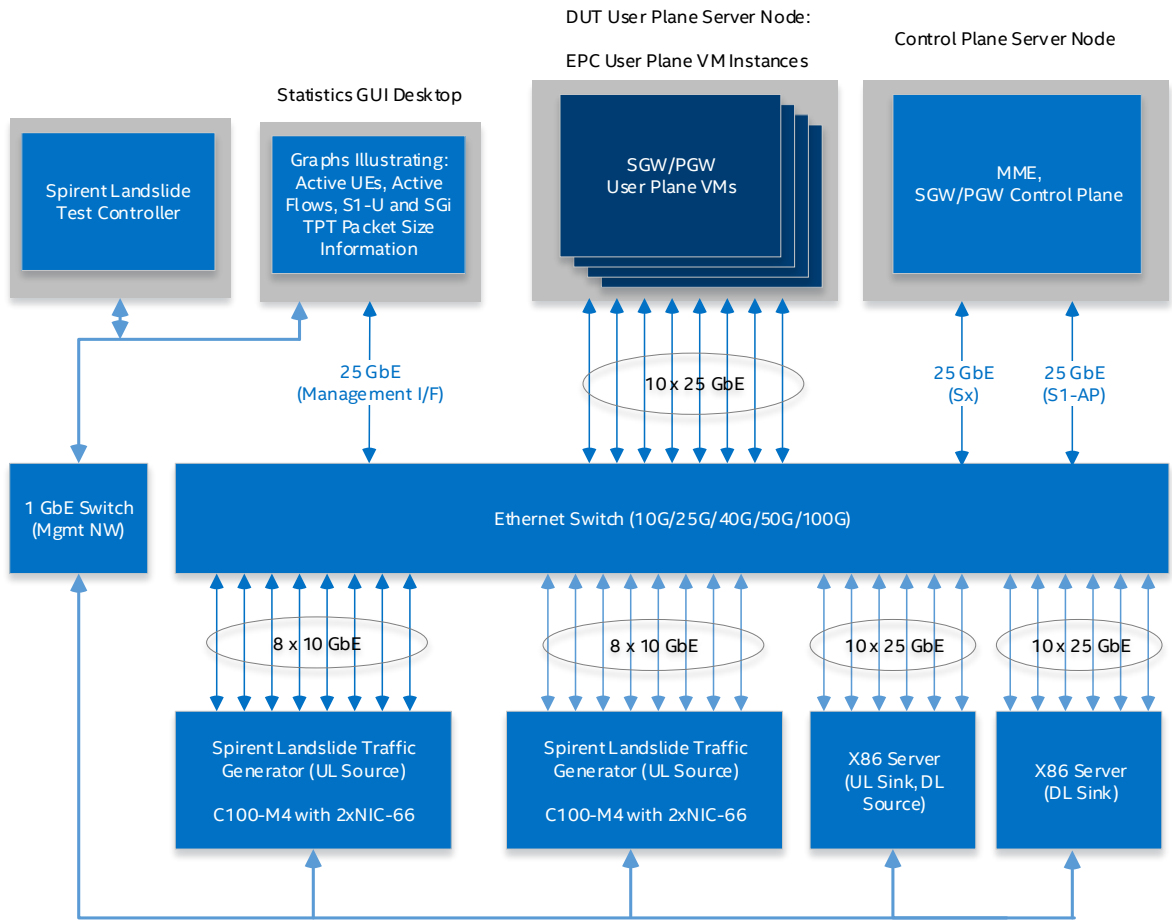

Figure 29. Test Infrastructure Block Diagram

Test Results

Intel and SKT measured packet throughput, overall bit rate, packet loss, and CPU usage metrics. The chart in Figure 30 shows the packet throughput and overall system-level throughput for each of the scenarios considered. As illustrated, 200+ Gbps throughput was consistently measured for packet rates between 36.9 and 42 million packets per second.[1]

The CPU usage on the user plane worker cores was between 52 and 66 percent. The DUT has a total of 56 cores, with 40 cores used for user plane worker cores. The mapping of cores for various purposes on the platform are described earlier in the document. The measurements shown in Figure 31 indicate a significant headroom (approximately 40 percent) on the user plane processing cores, while minimizing the use of the host OS and VM housekeeping cores. In addition, the 10 cores not allocated for user plane processing are entirely free. They can be repurposed for additional system functionality.

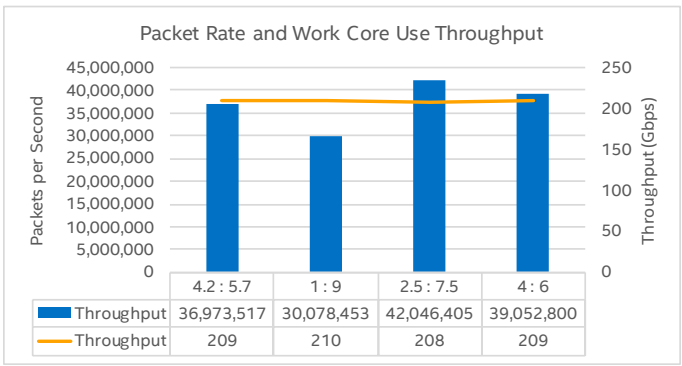

Figure 30. Measured Throughput for Each Traffic Profile

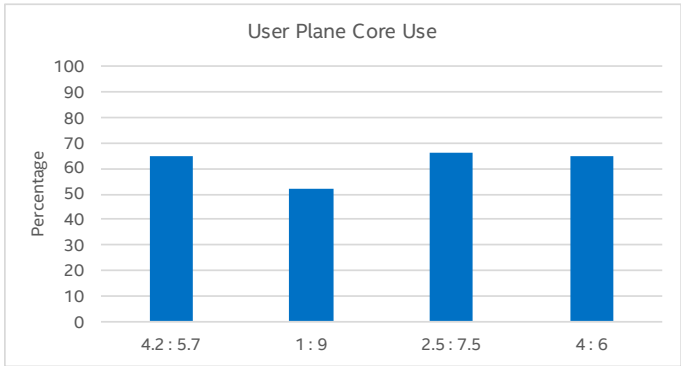

Figure 31. User Plane Worker Core Usages

Summary

As the industry is preparing for early 5G services that are more traffic demanding, the mobile packet core system expects to receive exponential amounts of user plane loads, and so it must be able to scale and perform efficiently for 5G use cases such as eMBB, URLLC, mMTC.

This white paper presents a new architecture for the user plane function (UPF) by analyzing various features required in UPF by modeling packet streams, resource consumption by software and hardware architectures, and how various packet sizes affect the UPF's overall performance. The analyses identify functions in which packet processing can be efficiently implemented by using optimal software and hardware partitioning. Integrating network interface adapters with infrastructural offload capabilities help achieve a high performance UPF architecture using standard commodity off-the-shelf (COTS) IT servers.

The results indicate that telecom equipment manufacturers (TEMs) and independent software vendors (ISVs) can leverage the recommended partitioning of functionalities between hardware and software. This recommended partition enables a pipeline implementation approach to develop commercial grade and deployable user plane functions for 5G core network infrastructures.

References and Resources

Evolved packet core (EPC) software used as a starting point for the prototype implementation used to characterize performance is based on commercial stack from Hong Kong Applied Science and Technology Research Institute* (ASTRI). Refer to www.astri.org for more information. This commercial stack includes SGW, PGW, and MME functionalities.

About the Authors

This white paper is jointly developed by SK Telecom and Intel Corporation.

SK Telecom—Network Technology R&D Center, ICT R&D Center

DongJin Lee received his Ph.D. in Computer Science (Internet Measurement) and completed Post-Doctoral research (Distributed System Optimization) from the University of Auckland in 2009 and 2011 respectively. Since then, he was part of the Samsung Electronics* (Network Business) Research and Development team for enhancing the LTE system, including advanced caching and machine learning. He joined SK Telecom in 2016, where he focuses on early 5G Core research and development activities, including NFV/SDN systems and performance analyses. Email: dongjin@sk.com

JongHan Park joined SK Telecom in 2014, where he currently focuses on 5G, NFV, SDN, and orchestration as the 5G core network architect in the overall mobile network context. Prior to joining SK Telecom, he spent a few years at AT&T Labs as a system architect, where he worked on various network topics, including BGP and network traffic data monitoring, network performance analysis, NFV, and SDN in the context of efficient and cost-effective CDN architecture. Dr. Park received his B.S. and M.S. degrees in Computer Science and Networked Systems from the University of California, San Diego, in 2000, and the University of California, Irvine, in 2005, respectively. He completed his Ph.D. in Computer Science from the University of California at Los Angeles in 2011. Email: jonghan.park@sk.com

Intel Corporation—Network Platforms Group, Data Center Group

Chetan Hiremath, Principal Engineer
chetan.hiremath@intel.com

John Mangan, Platform Solutions Architect
john.mangan@intel.com

Michael Lynch, Product Line Manager
michael.a.lynch@intel.com

Glossary of Terms

Term	Description
3GPP	Third Generation Partnership Project
5G	Fifth Generation Mobile Wireless Networks
5G CN	5G Core Network
ACL	Access Control List
ASIC	Application Specific Integrated Circuit
CapEx	Capital Expenditure
CDR	Charging Data Record
COTS	Commercial Off the shelf
CPU	Central Processing Unit
CUPS	Control Plane and User Plane Separation
DL	Downlink
DPI	Deep Packet Inspection
eMBB	Enhanced Mobile Broadband
eNodeB	Evolved NodeB
EPC	Evolved Packet Core
FPGA	Field Programmable Gate Array
gNB	Next Generation NodeB
GRE	Generic Routing Encapsulation
GTP-U	GPRS Tunneling Protocol User Plane
GTPv1-U	GPRS Tunneling Protocol User Plane (Version 1)
IMS	IP Multimedia Services
IoT	Internet of Things
IP	Internet Protocol
IPSec	Internet Protocol Security
ISV	Independent Software Vendor
LTE	Long Term Evolution
MME	Mobile Management Entity
mMTC	Massive Machine Type Communication
MPC	Mobile Packet Core
MVNO	Mobile Virtual Network Operator
NAT	Network Address Translation
NFV	Network Functions Virtualization

Term	Description
NFVI	Network Functions Virtualization Infrastructure
NIC	Network Interface Card
NPU	Network Processing Unit
NVGRE	Network Virtualization using Generic Routing Encapsulation
OpEx	Operational Expenditure
P2P	Peer-to-Peer
PDN	Packet Data Network
PDU	Protocol Data Unit
PGW	PDN Gateway
PGW-C	PDN Gateway Control Plane
QoS	Quality of Service
SAE	System Architecture Evolution
SGW	Serving Gateway
SGW-C	Serving Gateway Control Plane
SMS	Short Messaging Service
TEM	Telecom Equipment Manufacturer
TEP	Tunnel Endpoint
TOR	Top of Rack
UE	User Equipment. Synonymous with mobile device or mobile subscriber.
UL	Uplink
UPF	User Plane Function
URLLC	ultra-reliable and low-latency communication
VLAN	Virtual LAN
VM	Virtual Machine
VNF	Virtual Network Function
VNID	Virtual Network Identifier
VoLTE	Voice Over LTE
VxLAN	Virtual Extensible LAN

1. Testing conducted by SKT and Intel. See the Performance Results section for configurations and full test results.

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.

Benchmark results were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown." Implementation of these updates may make these results inapplicable to your device or system.

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice Revision #20110804

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software, or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at intel.com.

Printed in USA 0218/DO/PTI/PDF ♻ Please Recycle 337174-001US