Evolving to 6G: Improving the Cellular Core to lower control and data plane latency

Vivek Jain, Sourav Panda, Shixiong Qi, K. K. Ramakrishnan
University of California, Riverside

2022 1st International Conference on 6G Networking (6GNet) | 978-1-6654-6763-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/6GNet54646.2022.9830519

***Abstract*—With the commercialization and deployment of 5G, efforts are beginning to explore the design of the next generation of cellular networks, called 6G. New and constantly evolving use cases continue to place performance demands, especially for low latency communications, as these are still challenges for the 3GPP-specified 5G design, and will have to be met by the 6G design. Therefore, it is helpful to re-examine several aspects of the current cellular network's design and implementation.**

Based on our understanding of the 5G cellular network specifications, we explore different implementation options for a dis-aggregated 5G core and their performance implications. To improve the data plane performance, we consider advanced packet classification mechanisms to support fast packet processing in the User Plane Function (UPF), to improve the poor performance and scalability of the current design based on linked lists. Importantly, we implement the UPF function on a SmartNIC for forwarding and tunneling. The SmartNIC provides the fastpath for device traffic, while more complex functions of buffering and processing flows that suffer a miss on the SmartNIC P4 tables are processed by the host-based UPF. Compared to an efficient DPDK-based host UPF, the SmartNIC UPF increases the throughput for 64 Byte packets by almost 2×. Furthermore, we lower the packet forwarding latency by 3.75× by using the SmartNIC. In addition, we propose a novel context-level QoS mechanism that dynamically updates the Packet Detection Rule priority and resource allocation of a flow based on the user context. By combining our innovations, we can achieve low latency and high throughput that will help us evolve to the next generation 6G cellular networks.

***Index Terms*—Cellular core, Low latency, SmartNIC**

I. Introduction

5G cellular networks significantly improve performance, flexibility, and efficiency over the previous generation (*e.g.,* 4G LTE), driven by the improvements in radio access (*e.g.,* mmwave) and the softwarization of network resident functions.

Network function virtualization has been the driving force for the softwarization of the 5G core (5GC) network infrastructure. Fig. 1 shows the 5G architecture proposed in the 3rd Generation Partnership Project (3GPP) specification [1]. A mobile device, called User Equipment (UE), connects to a base station (gNB) using a wireless medium which are then typically connected a Metro Area Network backhaul, eventually connecting to the 5GC. The 5GC acts as an anchor point that connects the UEs to the data network and provides network resident services. It comprises control and data plane related network functions (NFs), such as the Access and Mobility Management Function (AMF) and User Plane Function (UPF). These NFs are responsible for various user operations, including authentication and authorization of the connecting user and taking care of user mobility to provide uninterrupted data service. The control plane NFs communicate over HTTP/REST, forming a service-based interface (SBI). The control plane configures one or more data path tunnels (using the GPRS Tunnelling Protocol (GTP)) for a user session to support flows with different levels of Quality of Service (QoS). These tunnels are also used to support user mobility and avoid connection disruption.

However, several entrenched limitations in both the control plane and data plane of the current 5GC design make it difficult to meet the promise of low latency for 5G. Introducing softwarization in the 5GC design was primarily to support ease of deployment and support multiple use-cases such as low-latency and high-throughput network access. However, the 5GC is still much of the base 3GPP control plane protocol to set up GPRS Tunnelling Protocol (GTP) tunnels and communicate over HTTP (i.e., the Service-Based Interface (SBI)) between 5GC NFs. The complex network I/O (*e.g.,* protocol processing, serialization/deserialization) makes the control plane protocol more 'heavyweight'.

The 5G data plane (UPF) performs packet routing between multiple UEs and the data network, requiring packet classification at the UPF. The 5G UPF employs packet detection rules (PDRs) to classify arriving packets. The current UPF design recommended by 3GPP uses a linked list to organize PDRs, which does not scale well and may impact data plane performance when the number of PDRs increases. Emerging use cases for 5G & beyond, such as multi-player games and flow-specific firewalls, are increasingly popular, leading to a much

larger number of PDRs in a single UPF for more fine-grained, flow-specific classification.

The 6th Generation Wireless Systems (6G), as an evolution of the current 5G system, is currently being designed [2]. We expect the 6G system to overcome any perceived 5G limitations, which we study in this work and discuss their potential impact on future 6G systems with new usage scenarios considered. For each limitation we identify, we propose an optimization in the context of 6G. The contribution of this work includes:

(1) Since 5GC design has adopted the software-based microservice design, we study the impact of different SBI implementations for communication between NFs, and the need for state synchronization when setting up data paths (*e.g.,* tunnel setup and buffering during handover).

(2) We explore several packet classification mechanisms that may be applied to speed up PDR lookup and update operations in the 5G UPF, compared to the linear search recommended by 3GPP.

(3) We propose a novel QoS mechanism (we call it "context-level") that goes beyond 5G's flow-level QoS. The context-level QoS dynamically adjusts the resource allocation based on the current user activity (*e.g.,* playing online games).

(4) Importantly, we develop a SmartNIC-based UPF that provides a fastpath for UPF processing, yielding up to 2× higher throughput and 3.75× lower packet processing latency.

II. Background and Motivation

A. Impact of NF placement and Service Based Interface on Tunnel setup

Industry direction has been the disaggregation of the components in the cellular environment, especially in the core network. NFs are separated, followed by the software-based microservice design pattern, moving away from purpose-built dedicated hardware component functions. For example, AT&T mobile core network spans more than 60 cloud-native virtual network functions from 15 different vendors [3]. This has the potential for better scaling and deployment flexibility with emerging cloud-native tools such as Kubernetes. However, several distinguishing characteristics of the cellular environment present challenges for the disaggregation of the cellular core: (1) tight coupling and timing constraints between the control and data plane, (2) increasing control plane activities that impact data plane handling, driven by emerging use cases such as IoT and the frequent handovers, especially with small cells.

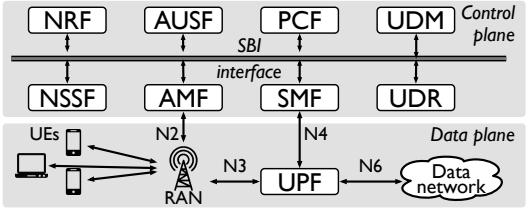


Figure 1. 5G architecture

Simply implementing the 5GC NFs as regular cloud-native microservices may not be enough. To provide low-latency access, it is crucial to accelerate the control plane procedures by minimizing communication and state coordination among different NFs. For instance, the cellular network carries data packets over what is commonly known as GTP tunnels to support mobility. This requires several message exchanges to set the tunnels up and keep the state consistent across NFs before the data transfer can begin. Even if the GTP tunnels are replaced with the more straightforward IP-in-IP tunnels, the overhead incurred due to message exchanges and state synchronization remains. Thus, the design needs to consider the tight coupling and affinity of 5GC NFs during their deployment. The placement of these microservices can significantly influence the control plane latency, especially if they are placed on different nodes requiring communication across multiple nodes. Secondly, 5G's SBI uses the heavyweight JSON as the serialization format. While JSON works well for machine-to-human communication due to its readability, it may not be a good match for machine-to-machine (NF-to-NF) communication. The serialization time and size of the message structure make the tunnel setup and state synchronization very expensive.

B. Packet Detection Rule lookup in the 5G UPF

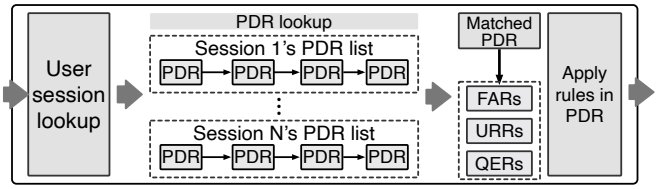


Figure 2. PDR lookup in the 5G UPF (recommended by 3GPP [4])

As shown in Fig. 2, following the 3GPP specification, the 5G UPF organizes the registered PDRs in a linked list according to their priority [4], in descending order. For each user session in the UPF, a PDR list defines the packet processing criteria (*e.g.,* classification, forwarding, buffering, etc.) for different flows in the user session. When a packet is received, the UPF first identifies the user session to which the packet belongs. The UPF traverses the PDR list for that session to perform

packet classification until a matched PDR is found. The matching PDR provides pointers to subsequent rules, *e.g.,* FARs (Forwarding Action Rules) and QERs (QoS Enforcement Rules), which together determine the action to be taken on the packet.

The current design may be adequate for the common 5G use cases being considered nowadays. Each user session typically has only a few PDRs, where the complexity of the PDR search is minimal so that PDR lookups being performed in a linked list are not that detrimental to performance. However, as cellular networks evolve from 5G to 6G, more advanced use cases may significantly increase the number of PDRs for a single user session (maybe grow to tens or even hundreds of entries). Typical advanced use cases include multi-player online gaming, flow-specific firewalls, or home gateways.

Multi-player games often have many application flows that need differentiated treatment. Within the gaming application flow, we may have latency-sensitive subflows that need to be delivered from a cloud server to UEs on a timely basis. At the same time, other subflows have different performance requirements (*e.g.,* flexible timing requirements), allowing the cellular network to take advantage of differentiated treatment. A game streamer may be sending her control data stream, video stream, and voice stream separately. Messaging video, audio, and text between users (that is usually mediated by the server) would also need to be treated differently, possibly also routed differently through the network. Having more fine-grained PDR rules on a per-flow basis, rather than for each UE, can support such differentiation. With the appropriate DSCP marking provided by the cloud server, the cellular network could efficiently handle even encrypted traffic. However, this inevitably increases the number of PDRs registered in the UPF for each user session, increasing search complexity based on the linked-list-based PDR lookup.

A single UE (or home gateway) may have multiple active IP flows operating concurrently, which is a use case needing a flow-specific firewall. A variety of Information Elements (IEs) is supported by PDRs, such as IP 5 tuples, and additional fields such as Type-of-Service and Security Parameter Index [4]. As firewalls move into the 5GC, having an efficient but full-featured firewall is essential. Having the UPF support feature-rich firewall rules to handle packet filtering for specific traffic, while desirable, can lead to increasing demand for PDRs for a single user session. Using "home gateways" is also likely to further increase the number of PDRs for a single user

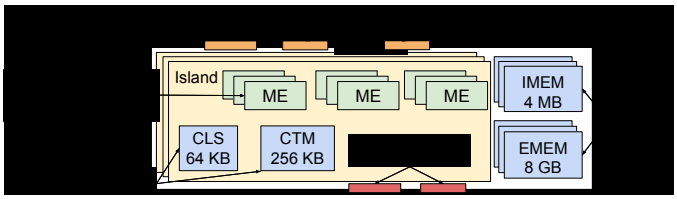


Figure 3. SmartNIC Architecture

session. Instead of treating each mobile device as a single UE, an aggregated "virtual" UE could be used as a 6G home gateway, connecting multiple mobile devices in a single home. The "virtual" UE can register many PDRs for its user sessions since it has many concurrent flows initiated by the backend mobile devices it connects to. Different flows have different levels of security and QoS, requiring different PDRs in the UPF to support packet processing for specific flows.

Thus, examining several likely advanced use cases, we notice the potential growth of PDRs for individual user sessions maintained in UPF. The complexity of finding PDRs in a linked list is *O(n)*, leading to long search delays when there are many PDRs in the list. This will impact data plane latency. We seek to optimize the PDR lookup by replacing the poorly scalable linked list with faster PDR lookup mechanisms.

C. Static priority in PDR and QER limits flow-level QoS

In the current 3GPP specification, the Session Management Function (SMF) assigns a static priority to PDRs as well as statically guaranteed bit-rate (GBR) and maximum bit-rate (MBR) in QERs. Even though these rules provide flow-level QoS, they do not account for the current user context, such as users playing games or streaming videos, because the MBR is static. The static prioritization does not consider the multitasking capability of more capable end-user devices, which can run multiple applications simultaneously. For example, when users start a multi-player gaming application, it is desirable to prioritize and allocate more resources for gaming flows instead of keeping resources reserved for VoIP and video streaming applications.

With advancements in applications and device capabilities, the 5G's "flow-level" QoS is insufficient to meet the desired flexibility. One of the goals of 6G would be to develop and support mechanisms for "context-level" QoS with a highly efficient PDR lookup which also requires dynamic PDR and QER updates based on ongoing traffic.

D. SmartNIC Architecture

SmartNICs have been extensively used to offload some protocol processing functions on hosts [5]–[8]. Offloading the UPF processing to programmable SmartNIC

can significantly improve overall performance. Packet forwarding by the UPF requires looking up the flow state and the PDR. The UPF has to be aware if the UE is idle or in the process of going through a handover from one base station to another, so that packets are buffered in the 5GC during this time. Since the UPF has to carry out flow-state dependent processing, the acceleration provided by the SmartNIC can help achieve high throughput and low latency processing. SmartNICs with significant processing and memory enable state tracking [5], [8], and with multiple high speed ports can act as a switch to directly forward packets without going in and out of the host. Their tight coupling with the host allows for control-plane events to be processed with very low latency. Co-location of the control plane NFs and the SmartNIC-based UPF on the same node ensures rules can be programmed rapidly, determining how packets of sessions are forwarded, tunneled, and prioritized.

The 5GC's dataplane (*i.e.,* UPF) is mainly involved in the forwarding of packets. Functions such as tunneling and buffering are managed by the rest of the 5GC, for example when the UE is idle or is going through a handover. Implementing the UPF on a SmartNIC requires programmability and configurability of the SmartNIC. We use the Netronome Agilio LX 2×40 GbE sNICs (depicted in Fig. 3), which have 8GB DDR3 DRAM memory and 96 Micro-Engines (MEs), which are 32-bit RISC processor cores with 8 thread contexts [5], [8]. Our SmartNIC-based UPF uses up to 80 MEs distributed across 7 islands, taking advantage of the highly-threaded flow processing given by SmartNIC. The threads in a ME are scheduled in a run-to-completion manner. Threads responsible for packet processing have access to large, shared global memories (*e.g.,* 8 GB capacity) and small, local memories (*e.g.,* 4 KB capacity) that are fast [8]. The SmartNIC offers extensive programmability by leveraging domain specific languages such as P4 [5]. By defining P4 match-action rules and populating them into a SmartNIC-based UPF, we can accelerate flow-state dependent processing in dataplane.

III. Design Overview

This section discusses the techniques that can accelerate the control and data plane processing capabilities of the 6G cellular core.

A. Accelerating tunneling setup using intelligent NF placement and Service Based Interface

Even though it is desirable to have the NFs of the 5GC disaggregated, it is equally important to consider their placement and the implementation of SBI used for state synchronization. We examine various implementations of the SBI between NFs that can speed up control plane operations, including that of state management to accelerate the tunnel setup procedures. A straightforward and intuitive way to optimize the SBI is to replace heavy JSON with lightweight serialization structures such as Protobuf and Flatbuffers.

There have been several research proposals to improve the performance of the SBI using advanced serialization structures. Neutrino [9] replaces ASN.1 encoding with Flatbuffers. Similarly, Buyakar *et al.* [10] used Protobuf instead of a JSON. While these different serialization structures reduce the serialization cost, they still incur costs due to: (1) wasteful messaging for state synchronization across individual NFs, and (2) communication delays. A clean-slate approach, CleanG [11], consolidates the control plane NFs to avoid state synchronization messaging between 5GC NFs. While this simplifies and reduces the number of control plane messages, scaling decisions are more coarse-grained, as the set of control-plane NFs needs to be scaled as a consolidated unit rather than individual overloaded microservices. It also limits the potential pipelining of control plane tasks that we could have if the NFs are organized as microservices, by leveraging the popular microservice chain paradigm.

Our ongoing work [12] optimizes 5G's SBI by examining the overheads for different serialization structures such as FlatBuffers, Protobuf, and JSON. By leveraging the shared memory optimizations and avoiding costs incurred due to serialization and complex HTTP/TCP processing, individual message processing overhead reduces by 13×. This allowed us to accelerate the control plane task (for a single UE case) by at least 2× for paging and handover compared to free5GC [13], avoiding spurious timeouts and retransmissions of data packets.

Further, CleanG [11] achieved up to 5× improvement for the completion time of control plane tasks over the equivalent implementation of the LTE core by combining two improvements: maintaining the state for the control and data planes in shared memory and simplifying the complex control plane protocol processing. Based on the learning from CleanG and building services using multiple NFs while utilizing shared memory in OpenNetVM [14], we ensure that the key NFs for the control plane are deployed within the same server. Keeping the state in a shared memory region, which individual NFs can access, reduces the need to exchange messages for state synchronization.

B. Fast PDR lookup & update

The PDR processing in the UPF is a packet classification problem. In order to achieve faster PDR lookup & updates for new cellular network use cases, we consider advanced packet classification mechanisms as alternatives to the linked list approach. We can extend existing packet classification mechanisms designed for routers to classify packets based on PDR Information Elements (IEs) without being limited to IP 5 tuples. We follow the same packet processing pipeline suggested by 3GPP (Fig. 2) but replace the linked-list-based PDR lookup step with advanced packet classification mechanisms.

Typical packet classification mechanisms can be broadly classified into partitioning-based and decision-tree-based classifiers. Both achieve better performance than a linked-list-based approach since their search complexity is less than $O(n)$. Partitioning-based classifiers partition a set of PDRs into multiple sub-tables based on specific criteria, *e.g.,* a typical partitioning-based classifier, Tuple Space Search (TSS [15]), partitions PDRs with the same number of prefix bits in each IE field by combining them into the same sub-table. Therefore, the search complexity can be reduced. However, the classification time of partitioning-based classifiers may not be optimal because the number of partitioned sub-tables shows randomness. Given n PDRs, partitioned sub-tables ranges from 1 to n. In the worst case, n sub-tables need to be searched before finding a matched PDR, which results in the same search complexity as the linked list approach. In addition, partitioning-based classifiers (*e.g.,* TSS) typically organize each sub-table into a key-value hash table. Performing a lookup in a sub-table requires software hashing, which often takes more time than a lookup in a linked list. In the worst case, partitioning-based classifiers have a much higher classification latency than the linked list due to the overhead of software hashing.

Unlike partitioning-based classifiers, decision-tree-based classifiers show less variability in packet classification performance. The key for such decision-tree-based classifiers is to generate multidimensional classification trees, given a set of PDRs. Various greedy heuristics have been explored to generate efficient multidimensional classification trees (with logarithmic search time) to support high-speed packet classification [16], [17]. However, decision-tree-based classifiers suffer from long rule update latency [18]. When constructing a multidimensional classification tree, the classifier cuts nodes starting from the 'root', which contains all the given PDRs. The classifier recursively splits the PDRs in each node until the number of PDRs within each leaf node is under a predefined threshold. Since different PDRs may intersect in some dimensions, splitting intersecting PDRs into different nodes is infeasible, resulting in a PDR being copied and added to multiple nodes. Multiple updates must be performed on each copy to ensure consistency among multiple copies of the same PDR, resulting in long delays for PDR updates. This vulnerability of decision-tree-based classifiers is avoidable in partitioning-based classifiers, as partitioning PDRs into multiple sub-tables does not lead to multiple copies of a PDR, thus achieving faster PDR updates.

For the 6G use case, a packet classification mechanism that enables fast PDR lookup and updates is desirable. Fast updates are also needed to support the dynamic prioritization of PDRs that we discussed earlier. On the other hand, it is important to minimize the PDR lookup latency to guarantee low data plane latency. Efforts have been made to operate packet classification as a hybrid of partitioning-based classifiers and decision-tree-based classifiers that exploit both approaches' advantages [18]–[20]. The low latency PDR lookup and fast updates of these approaches can ideally match our requirements. Other approaches include using reinforcement learning (RL) to achieve fast packet classification [21]. By constructing multidimensional classification trees with RL instead of greedy heuristics, [21] can achieve minimum PDR lookup delay since the RL seeks to maximize long-term rewards and can construct more efficient decision trees. While the performance of rule updates of [21] has not been reported and remains a concern, it is still worth exploring the use of machine learning to enhance PDR processing as we move toward AI-powered 6G services. Our current work [12] examines the effectiveness of these approaches.

C. "Context-level" QoS using dynamic prioritization

As discussed in §II-A, a straightforward implementation of context-level prioritization involves the SMF iteratively provisioning a new set of PDR rules with updated priorities. This approach has a significant drawback of going back and forth between the control and data planes. Since the control plane has no visibility into the application actions, it is desirable to design an intelligent scheme in the data plane to dynamically adapt to user context without involving other control plane functions. With intelligent monitoring capability and deep packet inspection (DPI) in the UPF, "context-level" QoS can be achieved without frequent coordination with

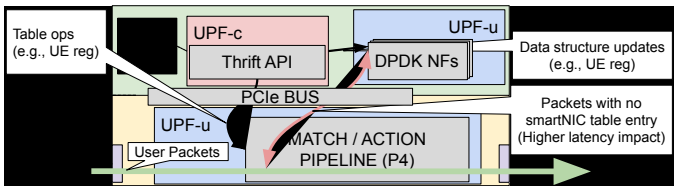


Figure 4. SmartNIC 5G Architecture

the control plane NFs in the 5GC. We can leverage DPI and machine-learning-based classification [22], [23] to detect flow information by identifying traffic through its application-level signature. We could potentially dynamically adjust the resources for an application flow rather than basing it on a static flow level QoS. This dynamic, context-sensitive prioritization can be provided by setting PDR rules based on the Policy Control Function (PCF) to comply with the operator's policies.

D. SmartNIC Offload

As a packet is processed, its flow state must be quickly located within the UPF, updated, and actions carried out based on the flow state. Prior work, such as DeepMatch [8] and SmartWatch [5], focus on flow state tracking within the SmartNIC for traffic monitoring. However, their designs are not directly applicable for use in an UPF. DeepMatch only supports a limited number of flows, while SmartWatch stores flow state in DRAM and is slower. Therefore, in our ongoing work, we explore combining the use of SRAM and DRAM on the SmartNIC for flow management to assure scalability and performance. Since the SmartNIC processes packets in a run-to-completion manner, every action on the SmartNIC influences the SmartNIC ME's time in the thread processing the packet. Careful design of the UPF processing on the SmartNIC is needed to perform all the UPF functions and maintain small packet latency as it is forwarded by the SmartNIC.

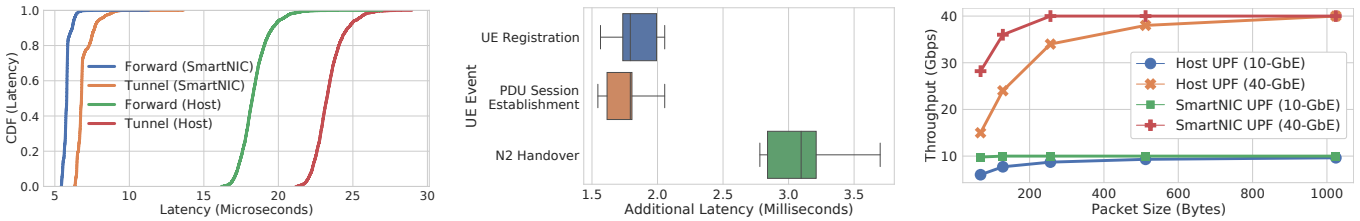
Fig. 4 shows the SmartNIC UPF architecture. On arrival of a downlink packet, we encapsulate the packet with a GTP header and determine the TEID to set in the GTP header. This is done by matching against a P4 table wherein if the destination IP of a packet (matching key) equals a specific UE IP (table entry), we encapsulate the packet (action) and set the TEID associated with the UE IP (action data). On the other hand, for uplink packets, we decapsulate the packet after segregating the uplink traffic from the downlink traffic in the parsing phase of the P4 pipeline. Doing this entirely in the SmartNIC helps reduce the packet processing latency significantly, bypassing the host interaction. However, some packets cannot be fully processed on the SmartNIC because of the limited P4 table capacity. These packets will have to be processed on the host UPF. For a UE session establishment, both uplink and downlink rules are inserted in the P4 match action tables of the SmartNIC UPF. This corresponds to what would be performed on a host UPF. However, unlike the SmartNIC-based UPF, where P4 tables are updated, a host-based UPF would update the PDR tables stored in the host's memory.

In our current prototype, the SmartNIC forwards packets to the host for buffering. However, buffering within the SmartNIC has promise, leading to higher packet processing throughput. When a packet is buffered in the SmartNIC, we only have to DMA the packet from one memory region to another within the SmartNIC [8]. On the other hand, when buffering packets on the host, packets will have to be DMA'd from the SmartNIC to the host. PCIe transactions between the SmartNIC and host can take up to 2μs [24], reducing data path throughput and increasing latency [25]. To provide context-level QoS, the SmartNIC's P4 pipeline can support the dynamic setting of the MBR based on arbitrary packet header attributes [26].

IV. Evaluation

We start by measuring the data plane acceleration achieved by processing user packets in the SmartNIC as opposed to the host. We use a MoonGen [27] traffic generator to replay a pcap consisting of a single flow (thus avoiding the PDR lookup overhead contribution). We then evaluate two different features on the host and SmartNIC UPF, namely forwarding and tunneling. In simple forwarding, the packet is matched against a table and then sent out of another port. In tunneling, after matching on the table, downlink packets are encapsulated and forwarded out of another port.

End-to-end packet processing latency: As shown in Fig. 5(a), the tunneling operation on the SmartNIC is 3.75× faster than on the host. This is because the SmartNIC UPF completely avoids the overhead of going in and out of the host memory and PCIe bus [5]. Compared to simple packet forwarding without using a tunnel (baseline), the SmartNIC UPF spends 19% more time tunneling. On the other hand, a host would spend 30.2% additional time (or a larger base value) on tunneling. The lower cost of tunneling on the SmartNIC can be attributed to the hardware acceleration for PDR lookup and packet encapsulation. Lastly, by offloading the UPF processing to the SmartNIC, we leave more CPU cores for the other 5GC NFs.



(a) CDF end-to-end latency (b) Thrift API additional latency (c) UPF Throughput

Figure 5. SmartNIC-based UPF overheads and performance

Control plane latencies: Next, we evaluate the control plane latencies to update the rules on the UPF. Due to the Thrift API, the SmartNIC UPF takes slightly longer than the host UPF to update the rules. Fig. 5(b) shows the additional latency spent on the Thrift API to update the rules. A N2Handover takes longer with the Thrift API than UE Registration and PDU Session Establishment because of more Thrift API-based communications. The number of P4 match-action table rules is limited to 64K [28]. This means flows will potentially miss on the P4 match-action table and forward the packet to the host, resulting in higher packet processing latency (Fig. 5(a)). As the P4 table in the SmartNIC will only contain the active set of flows, when flows miss on this table, the host has to perform the intended action. This is followed by pushing a rule to the SmartNIC. Using the Thrift API, we can push rules at the rate of 8K rules per second based on our experiments, which is 6.6× more than the rule insertion rate seen on programmable switches [29]. Therefore, using the SmartNIC, we attain network acceleration without dealing with a low rule insertion rate with P4 capable network switches.

UPF throughput: Fig. 5(c) shows the throughput achieved with the SmartNIC and host UPF for 10G and 40G link rates. In the SmartNIC, our 40GbE UPF implementation achieves 28.81 Gbps for 68-byte packets. This is 1.88× the throughput achieved with a 4 CPU core host UPF (and a corresponding number of Rx and Tx threads using OpenNetVM [14]) due to the acceleration provided by the SmartNIC. The throughput achieved with the SmartNIC UPF of 28.81 Gbps is consistent with the throughput observed earlier [5], [30], where it is speculated that the bottleneck lies in the scatter-gather operation for packets. Observe that with the 10GbE SmartNIC UPF, implemented on a Netronome CX 2x10GbE [31], we achieve the 10 Gbps line rate for all packet sizes.

Table. I shows the percentage increase in the completion time of control plane tasks due to having the SmartNIC as opposed to the host for the UPF. We observe that the completion time of control plane tasks (including SMF and AMF communication) increases by at most 2.33%, which is tolerable. For example, on average, the N2Handover events take 135.25ms with the SmartNIC UPF as opposed to 132.11ms with the host UPF. Similarly, UE registration takes 137.82ms as opposed to 136.02ms with the host UPF.

Table I
CONTROL PLANE TASK COMPLETION TIME: PERCENTAGE INCREASE DUE TO SMARTNIC

UE Registration	PDU Session Establishment	N2 Handover
1.33%	2.29%	2.33%

V. RELATED WORK

A significant amount of work and study has been done to analyze and improve the performance of the 5G and previous cellular core. Learning from these studies is likely to drive the improvements and building of the next-generation cellular core. Xu et al. [32] perform a comprehensive analysis of the operational 5G network. They study different parameters such as coverage, performance, and energy consumption on the UE. Their study shows that while 5G shows significant improvement in throughput and latency (due to improvements in radio technology), it suffers from a significant throughput reduction in 5G due to large handover latency, which interrupts the ongoing TCP transmission. A similar study [33] has shown performance reduction in LTE networks with different protocols such as TCP and SPDY, implying that the cellular core still possesses the latency issue, and there is a critical need for redesigning it.

Even though the 6G is relatively new, a few recent research proposals highlight the vision for new wireless for emerging use cases, such as Connected Autonomous Vehicles and Multisensory XR Applications. For instance, Saad et al. [34] discuss new services such as Human-Centric Services, and Energy Services in addition to enhanced Massive URLLC (mURLLC) and Mobile Broadband Reliable Low Latency Communication for Haptics and AR/VR/XR applications.

VI. Conclusion

This paper proposed several optimizations for the cellular core to improve performance while still being compliant with the 3GPP specification. We explored different ways to minimize the penalties caused by tunnel setup and message processing in the 5GC. By carefully placing NFs and leveraging a fast SBI, 6G can significantly accelerate control plane operations. We proposed a novel "context level" QoS that requires dynamic prioritization of PDRs and QERs. With advanced packet classification mechanisms in the UPF, PDR lookup and update latency can be minimized, allowing the UPF to prioritize the treatment of multiple flows for a single UE. The data plane of UPF is further enhanced by offloading the data plane processing to the SmartNIC. Our preliminary experimental results show that offloading the UPF functions to a SmartNIC increases the data path throughput for 64-byte packets by almost 2× and reduces data packet processing latency by 3.75× compared to a host-based UPF implementation using DPDK.

Acknowledgment

We thank US National Science Foundation for generous support with grants CRI-1823270 and CSR-1763929.

References

[1] "System architecture for the 5G System (5GS)," https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/16.06.00_60/ts_123501v160600p.pdf.

[2] Saad et al, "A vision of 6g wireless systems: Applications, trends, technologies, and open research problems," *IEEE Network*, vol. 34, no. 3, 2020.

[3] Hakl, "Improving the cloud for telcos: Updates of Microsoft's acquisition of AT&T's network cloud." [Online]. Available: https://azure.microsoft.com/en-us/blog/improving-the-cloud-for-telcos-updates-of-microsoft-s-acquisition-of-att-s-network-cloud/

[4] 3GPP, "LTE; 5G; Interface between the Control Plane and the User Plane nodes," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 29.244, 09 2019, version 15.7.0.

[5] Panda et al., *SmartWatch: Accurate Traffic Analysis and Flow-State Tracking for Intrusion Prevention Using SmartNICs*. New York, NY, USA: Association for Computing Machinery, 2021.

[6] Firestone et al., "Azure accelerated networking: Smartnics in the public cloud," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, April 2018.

[7] Zhipeng et al., "Achieving 100gbps intrusion prevention on a single server," in *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. USENIX Association, Nov. 2020, pp. 1083–1100.

[8] Hypolite et al., *DeepMatch: Practical Deep Packet Inspection in the Data Plane Using Network Processors*. New York, NY, USA: Association for Computing Machinery, 2020.

[9] Ahmad et al., "A Low Latency and Consistent Cellular Control Plane," in *ACM SIGCOMM 2020 Conference*.

[10] Buyakar et al., "Prototyping and load balancing the service based architecture of 5g core using nfv," in *2019 IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2019.

[11] Mohammadkhan et al., "Cleang—improving the architecture and protocols for future cellular networks with nfv," *IEEE/ACM Transactions on Networking*, vol. 28, no. 6, 2020.

[12] Jain et al., "L25GC: A Low Latency 5G Core Network based on High-Performance NFV Platforms," in *Proceedings of the ACM SIGCOMM Conference, Aug. 2022 (to appear)*.

[13] free5GC. [Online]. Available: https://www.free5gc.org/

[14] Zhang et al., "Opennetvm: A platform for high performance network service chains," in *Workshop on Hot topics in Middleboxes and Network Function Virtualization*, 2016.

[15] Srinivasan et al., "Packet classification using tuple space search," in *ACM SIGCOMM 1999 Conference*.

[16] Singh et al., "Packet classification using multidimensional cutting," in *Proceedings of the ACM SIGCOMM 2003 Conference*.

[17] Qi et al., "Packet classification algorithms: From theory to practice," in *IEEE INFOCOM 2009*, 2009.

[18] Yingchareonthawornchai et al., "A sorted partitioning approach to high-speed and fast-update openflow classification," in *2016 IEEE 24th ICNP*, 2016.

[19] Vamanan et al., "Efficuts: Optimizing packet classification for memory and throughput," in *ACM SIGCOMM 2010*.

[20] He et al., "Meta-algorithms for software-based packet classification," in *2014 IEEE 22nd ICNP*, 2014.

[21] Liang et al., "Neural packet classification," in *Proceedings of the 2019 ACM SIGCOMM Conference*.

[22] Jin et al., "A modular machine learning system for flow-level traffic classification in large networks," vol. 6, 2012.

[23] Sen et al., "Accurate, scalable in-network identification of p2p traffic using application signatures," in *Proceedings of the 13th international conference on World Wide Web*, 2004.

[24] Neugebauer et al., "Understanding pcie performance for end host networking," in *ACM SIGCOMM 2018 Conference*.

[25] Bose et al., "Leveraging programmable dataplanes for a high performance 5g user plane function," in *5th Asia-Pacific Workshop on Networking (APNet 2021)*.

[26] "MeteringImplementation." [Online]. Available: https://github.com/open-nfpsw/p4_basic_lb_metering_nic

[27] Paul et al., "Moongen: A scriptable high-speed packet generator," in *Proceedings of 2015 Internet Measurement Conference*.

[28] Harkous et al., "Performance study of p4 programmable devices: Flow scalability and rule update responsiveness," in *2021 IFIP Networking Conference (IFIP Networking)*.

[29] Jiarong et al., "NetWarden: Mitigating network covert channels while preserving performance," in *29th USENIX Security Symposium*.

[30] YoungGyoun et al., "AccelTCP: Accelerating network applications with stateful TCP offloading," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. Santa Clara, CA: USENIX Association, Feb. 2020.

[31] "Netronome CX 2x10GbE Whitepaper." [Online]. Available: https://www.netronome.com/media/documents/PB_Agilio_CX_2x10GbE-7-20.pdf

[32] Xu et al., "Understanding operational 5g: A first measurement study on its coverage, performance and energy consumption," in *ACM SIGCOMM 2020 Conference*.

[33] Erman et. al, "Towards a spdy'ier mobile web?" *IEEE/ACM Transactions on Networking*, vol. 23, 2015.

[34] Saad et al., "A vision of 6g wireless systems: Applications, trends, technologies, and open research problems," *IEEE network*, vol. 34, no. 3, 2019.