L^{2}6GC: Evolving the Low-Latency Core for Future Cellular Networks

Shixiong Qi and K. K. Ramakrishnan, *University of California, Riverside, CA, 92521, USA*
Jyh-Cheng Chen, *National Yang Ming Chiao Tung University, Hsinchu, 30010, Taiwan*

With the deployment of higher speed radio (e.g., millimeter-wave) channels, emerging cellular use cases will place a heightened emphasis on performance of the cellular core (we call it "6GC"). Based on our earlier work on designing a low-latency, high-throughput 5G cellular core, we outline a design for L^{2}6GC, a 6G cellular core preserving the current trend to disaggregate core network functions (NFs) while ensuring low-latency communication between them. L^{2}6GC leverages shared memory processing for low-latency control plane communication, providing an equivalent service to the 3rd Generation Partnership Project service-based interface and supports information exchange between NFs for each user session. L^{2}6GC offloads data plane processing to Smart Network Interface Cards, exploiting scalable, flow-level packet classifiers to expedite forwarding-rule lookups when dealing with multiple user sessions. Overall, L^{2}6GC optimizes the cellular core, improving latency, scalability, and the user's quality of experience.

The integration of network function virtualization (NFV) into cellular networks, adopted by operators, vendors, and standards bodies (e.g., 3GPP), has resulted in the fundamental transformation of the 5G cellular core (5GC) toward a more flexible microservice-based architecture. The cellular core, in essence, serves as the central anchor point between user equipment (UE) and the data network (DN) (e.g., Internet), managing crucial control and data plane functions, such as user authentication and mobility management, user packet routing, and classification.

Despite architectural advancements, latency is likely to be a critical constraint for emerging applications using 6G networks. We identify three components of latency in the 3rd Generation Partnership Project (3GPP)-specified cellular core:

- *Data plane latency*: Caused by processing delays within the data plane, e.g., packet classification.
- *Control plane latency*: The time to complete control plane procedures, such as UE registration and session establishment.
- *Latency for interaction between the data plane and control plane*: Caused by interactions between the data plane and control plane functions. An example is when the control plane installs packet processing rules in the data plane. While much attention has been devoted to data plane latency, control plane latency and the latency for the interaction between the cellular data plane and control plane can have a significant impact on these applications.

Several factors contribute to the increasing significance of these contributors to latency in 6G networks:

- *Emerging application needs*: New 6G applications [e.g., real-time augmented reality (AR)/virtual reality, Internet of Things (IoT) applications, and cloud gaming] will demand low-latency access to Internet services. We anticipate increasing mobility of UE, including connected vehicles and unmanned aerial vehicles. Along with the deployment of smaller cells that have much smaller coverage areas with higher speed wireless links, higher mobility will result in frequent UE handovers, necessitating more frequent interactions between the 6GC control plane and data plane. Several of these use cases will also involve an increasing number of packet detection rules (PDRs).[1,2] This further emphasizes the need for low-latency control-plane-to-data-plane interactions.

1089-7801 © 2024 IEEE
Digital Object Identifier 10.1109/MIC.2024.3376655
Date of publication 18 March 2024; date of current version 16 April 2024.

Further, having the user plane function (UPF) entirely in software fails to meet stringent latency requirements and consumes excessive CPU resources.[1]

- *Architectural evolution of the cellular core*: The adoption of decoupled, microservice-based designs in the cellular core's control plane aims to optimize operational costs by leveraging cloud elasticity. However, this shift will result in increased communication [using the relatively slow HTTP/Representational State Transfer (REST)-based service-based interface (SBI)] between control plane functions, leading to added control plane delays.[1,2]

As we look ahead to the potential of future 6G wireless environments, it is imperative to address these limitations. We thoroughly examine these constraints and propose L^{2}6GC, an approach for a more efficient low-latency cellular core design. Our contributions encompass the following:

- *Evolving to a high-performance SBI*: L^{2}6GC utilizes shared memory processing to offer a low-latency data path for exchanging control plane messages. L^{2}6GC adds *synchronous* input–output (I/O) support on top of *asynchronous* shared memory I/O designed primarily for layer 2 NFs, ensuring our high-performance SBI is still compliant with 3GPP standards.
- *Enhancing PDR lookups*: We explore innovative packet classification mechanisms to expedite PDR lookups and update operations in the UPF, going beyond the linear search approach recommended by 3GPP.
- *Smart Network Interface Cards (SmartNIC)-based UPF*: We develop a SmartNIC-based UPF that offloads data packet processing, significantly increasing throughput and reducing packet processing latency.

We have an open source 5GC implementation that incorporates some of these proposed L^{2}6GC components at https://github.com/nycu-ucr/L25GC-plus.git.

BACKGROUND AND MOTIVATION

Evolving the Architecture of the Cellular Core

A cellular network's packet processing functions are typically divided between the radio access network (RAN) and the core network. The RAN manages wireless channel resources and processing at the cellular base stations to connect UEs and the cellular core. The cellular core forms the heart of an operator's cellular network. Figure 1 exemplifies the typical architecture of the 3GPP-suggested 5GC.[1] A 5GC is further split into the data and control planes. The UPF in the data plane connects UEs to the DN to access data (e.g., Internet) services.

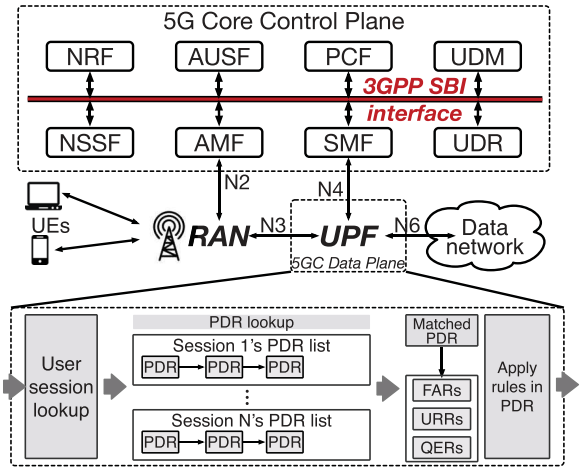

FIGURE 1. The architecture of 5GC control and data planes. The 5G UPF performs PDR lookup (based on the linked list) to enable data packet forwarding between the RAN and data network. 3GPP: 3rd Generation Partnership Project; 5GC: 5G cellular core; AMF: mobility management function; AUSF: authentification server function; FAR: forwarding action rule; NRF: network repository function; NSSF: network slice selection function; PCF: policy control function; PDR: packet detection rule; QER: quality of service enforcement rule; SBI: service-based interface; SMF: service management function; UDM: unified data management; UDR: unified data repository; UE: user equipment; UPF: user plane function.

Instead of purpose-built hardware appliances of the past, recent generations of the cellular core have been primarily built as a set of disaggregated microservice-based NFs. Several control plane NFs play important roles, including the access and mobility function and service management function (SMF). However, this disaggregated design requires the 3GPP SBI to interconnect the NFs, following a cloud-native, service-based architecture.

Emerging 6G Application Needs

Emerging applications, such as AR and autonomous vehicles, can enhance the perception of the real world with virtual holograms.[3] These applications need ubiquity and high bandwidth, and they have stringent

latency constraints.[4] It is, therefore, critical that the cellular core supports the bandwidth needs of all of the connected UEs without being a data plane bottleneck and that it does not introduce significant latency in either the data plane or control plane.[5]

Multiplayer games often have many application flows (throughput and latency sensitive) that need differentiated treatment. A game streamer may also be sending each of her control data, video, and voice streams separately. Messaging video, audio, and text between users (usually mediated by a server) may also need to be treated differently and even routed differently through the network. All of these suggest the need for more fine-grained packet detection and handling of each distinct flow, using appropriate Differentiated Services Code Point markings, rather than treating all flows of a UE in the same way.

In the future, with the widespread use of cellular networks as replacements for traditional fixed-wireline infrastructure, we anticipate challenges arising from 6G cellular home gateways that connect multiple mobile devices behaving as a single "virtual" UE. A cellular home gateway will likely increase the number of PDRs for a single user session. A 6G home gateway may have multiple concurrently active Internet Protocol (IP) flows with different security and QoS levels, requiring different PDRs in the UPF to support their processing. In addition, having the UPF support feature-rich firewall rules to filter specific traffic, while desirable, can lead to increasing demand for PDRs for a single user session.

Limitations of the 3GPP Cellular Core

Slow 3GPP SBI

The industry direction has been toward the disaggregation of the cellular core network components. For example, AT&T's mobile core network spans more than 60 cloud-native virtual NFs from 15 different vendors.[6] Several distinguishing characteristics of the cellular environment present challenges for such a disaggregation: 1) tight coupling and timing constraints between the control and data planes and 2) frequent and increasing control plane activities that impact data plane handling, driven by emerging use cases, such as IoT and frequent handovers, especially with small cells. Further, using the SBI for communication involves using JSON as the serialization format, which makes control actions [e.g., General Packet Radio Service Tunneling Protocol (GTP) tunnel setup and state synchronization] very expensive. To provide low-latency access and scalability, it is critical to avoid the overheads of such an interface, including data copying, context switching, and notification overheads.

Slow UPF

The UPF is responsible for packet buffering, classification, and forwarding between the UE and the DN. A software-based UPF may become a bottleneck if it is not implemented properly, resulting in limited data plane throughput and high latency. Using programmable hardware (e.g., SmartNICs and programmable switches) can accelerate the data plane's performance. However, the added Peripheral Component Interconnect Express (PCIe) and link delays increase the latency for interaction between the data plane (programmable hardware) and control plane on the host. Often overlooked in the past, these can become a bottleneck when we require frequent rule updates (e.g., PDRs) between the control plane NFs and the UPF. This is likely in 6G due to smaller cells and the increased mobility of UEs, leading to more frequent handovers. The limited memory in programmable hardware for storing packet processing rules [implemented as programming protocol-independent packet processors (P4) tables] may slow down data packet processing, especially with frequent rule misses in the P4 tables. This requires us to carefully apply hardware acceleration on the UPF without compromising the control and data plane performance.

Poor Scalability—PDR Rule Lookup

As shown in Figure 1, following the 3GPP specification, the UPF organizes the registered PDRs in a linked list,[1] in descending order of priority. A PDR list in the UPF defines the packet processing criteria (e.g., forwarding and buffering) of flows in a user session. The UPF traverses the PDR list for that session to classify each packet. The matching PDR points to subsequent rules, e.g., forwarding action rules and QoS enforcement rules, which together determine the action to be taken on the packet.

The complexity of the linked list is minimal for common 5G use cases, where each user session has a few PDRs. However, more advanced 6G use cases may significantly increase the number of PDRs to tens or even hundreds of entries for a single user session, leading to long PDR search delays, thus increasing data plane latency.

DESIGN OF L^{2}6GC

3GPP SBI Using Shared Memory Communication

We exploit shared memory I/O for low-latency data exchange between control plane NFs based on OpenNetVM,[7] a high-performance NFV framework. Shared

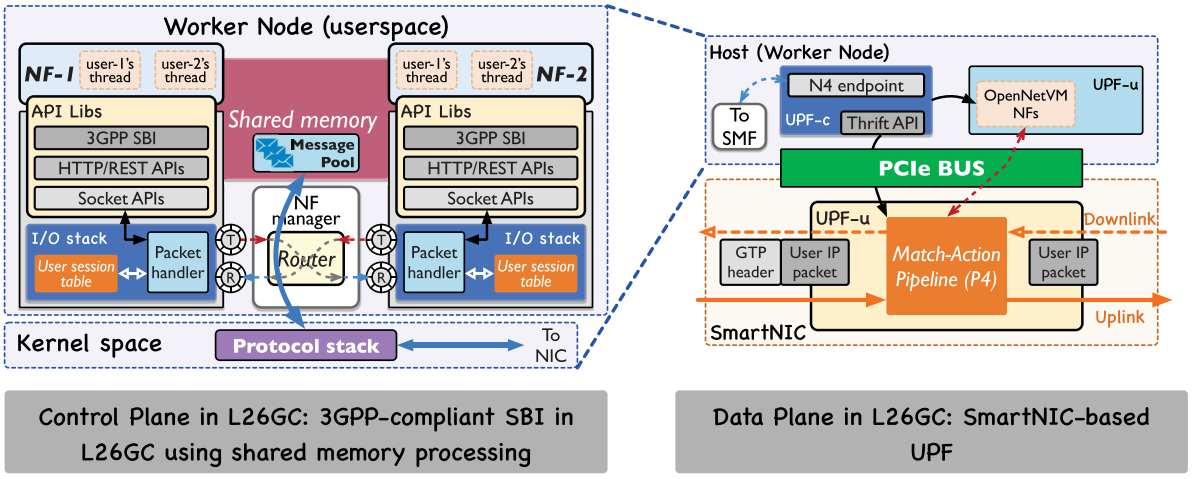

FIGURE 2. The design of control and data planes in L^{2}6GC: a 3GPP-compliant SBI in L^{2}6GC using shared memory processing and SmartNIC-based UPF. API: application programming interface; I/O: input–output; IP: Internet Protocol; Libs: libraries; REST: Representational State Transfer; SMF: service management function.

memory processing greatly reduces the latency caused by data copies and protocol processing.

However, OpenNetVM is mainly designed for layer 2 NFs using stateless *nonblocking, asynchronous* shared memory I/O. With asynchronous I/O, the caller (the source NF) does not wait for the response from the callee (the destination NF), but the 3GPP SBI supports request–response interactions between the caller and callee. Further, the cellular core control plane relies on connection primitives to distinguish between user sessions and handle their events in parallel. Therefore, L^{2}6GC introduces synchronous shared memory I/O and concurrent connections between control plane NFs, as depicted in Figure 2. Key components include a per-node shared memory pool to provide a shareable back end for data exchange between NFs. Zero-copy shared memory processing is aided by descriptor delivery between NFs, with the descriptor pointing to the message payload residing in the shared memory.

Synchronous I/O and Application Programming Interface (API) Libraries

L^{2}6GC implements the synchronous SBI by using the X-IO framework.[8] X-IO enhances the nonblocking, OpenNetVM-style shared memory I/O with blocking primitives, using condition variables, a commonly used synchronization primitive in Linux. This enables the request–response communication pattern required by 3GPP SBI. A set of HTTP/REST APIs is layered on top of the socket interface, using the condition variable from X-IO to block shared memory I/O when a response is expected. Refer to Liu et al.[9] for implementation details. Compared to PEPC,[10] which reduces control plane latency by having the user state shared by Evolved Packet Core functions (as threads) in a monolithic process, our design retains the disaggregated microservice-based design philosophy while enabling low-latency communication in the control plane.

Concurrent Connection Management

To distinguish between user sessions during control plane communication between NFs, a specific connection is bound to each user session (handled by a dedicated thread). The shared memory I/O stack in each NF maintains a local connection table, mapping the connection (user session) to a local thread. A packet handler demultiplexes the received descriptors to backend threads in the NF after consulting the connection table. Further details are elaborated in Qi et al.[8]

Offloading UPF Functions to SmartNICs

Figure 2 also shows the SmartNIC-based UPF in L^{2}6GC. We split the UPF into UPF-C and UPF-U. The UPF-C remains on the host as an endpoint to interact with the SMF in the control plane. The UPF-U is offloaded to a SmartNIC for high-performance packet processing.

We implement the UPF-U with a P4 match-action pipeline on the SmartNIC. When a downlink packet arrives, the UPF-U encapsulates the packet by adding a GTP header. If the destination IP of a packet matches

a specific UE IP, we encapsulate the packet and configure the tunnel endpoint identifier associated with that UE's IP. Similarly, for uplink packets, the UPF-U decapsulates the GTP header.

Due to the limited capacity of the P4 table, certain packets may not be fully processed on the SmartNIC. These packets must be processed on a host-based UPF-U. We choose OpenNetVM to implement the host-based UPF-U for its relatively high performance compared to other software-based solutions, e.g., kernel-based processing.

Feasibility of a P4-Switch-Based UPF

There are efforts to implement UPF on programmable switches, such as P4Switches.[11] The switch data path offers high throughput. However, this has to be balanced against the limited switch buffering, increasing the probability of packet loss and retransmissions[12] during handovers, if buffering is performed at the UPF as in Jain et al.[1] The limited P4Switch memory may also pose a challenge for maintaining a sufficient number of PDR rules. Requiring frequent PDR updates from the host also adds latency in the data plane, especially for control plane state transitions (e.g., paging operations). We choose a SmartNIC for its larger memory capacity [e.g., more than 8 GB of memory on a Netronome Agilio LX 2×40-Gigabit Ethernet (GbE) sNIC] compared to a typical programmable switch (e.g., on the order of 100 MB). SmartNICs also avoid the additional communication delay to the host compared to programmable switches.

Fast PDR Lookup in UPF

A simplistic linked-list-based packet classifier has $O(n)$ search complexity, resulting in high PDR lookup delays for use cases requiring many PDRs. To speed up, we adopt more sophisticated packet classification techniques.

Packet classifiers fall into two main categories: partitioning based and decision tree based; both offer less search complexity than $O(n)$. Partitioning-based classifiers reduce search complexity by dividing PDRs into subtables based on various partitioning criteria. For instance, tuple space search[13] (TSS) partitions PDRs into subtables based on tuples, i.e., PDR information elements. However, partitioning-based classifiers suffer from fluctuating search performance due to variability in the number of partitioned subtables.[14] On the other hand, decision-tree-based classifiers leverage the idea of multidimensional binary search to reduce search complexity, resulting in less variability in search performance. However, decision-tree-based classifiers have long PDR update times.[2]

L^{2}6GC takes advantage of a hybrid mechanism that combines partitioning-based and decision-tree-based classifiers, capitalizing on the strengths of both methods. We implement PartitionSort[14] in our UPF for fast PDR lookups.

State Management and Resiliency in L^{2}6GC

Our current implementation of L^{2}6GC adopts a stateful design with the NF maintaining the state (e.g., the UE context) in the local cache. We believe this has far less overhead compared to stateless cellular core designs,[15] which decouple core NFs from state management by using an external unstructured data storage function (UDSF). For resiliency, L^{2}6GC uses the principles of *external synchrony*[1] to replicate the state to a standby, *asynchronously* as in.[16] Failover to the standby replica is initiated by the NF management system, avoiding having the user re-establish the session after a failure, as suggested by 3GPP's procedures. Our improved strategy reduces the recovery delay by 3× compared to the 3GPP-based recovery procedure and maintains data plane goodput throughout the failover, unlike the 3GPP-based recovery procedure, whose goodput drops to zero. Additional implementation and evaluation details of L^{2}6GC's resiliency and failover are in our previous work.[1]

EVALUATION

We evaluate L^{2}6GC across multiple dimensions, including the completion time of representative control plane events, throughput and latency for SmartNIC-offloaded UPFs, and PDR lookup delays.

Low-Latency Control Plane SBI

We compare L^{2}6GC's shared memory SBI with the kernel-based SBI from free5GC[17] (a 3GPP-compliant 5GC implementation adopted as a core by multiple groups). We tested the following control plane events: UE registration, packet data unit session establishment, and paging (UE transitions from an idle to active state). Multiple simulated UEs concurrently generate control plane events, using the UE&RAN simulator from Jain et al.[1]

Results in Figure 3 show that L^{2}6GC's shared memory SBI consistently achieves a 1.5× to 2.2× reduction in completion time, demonstrating the benefit of using shared memory processing. We also evaluated L^{2}6GC in a commercial testbed with five real UEs. L^{2}6GC achieves a similar latency reduction of up to 2× for several control plane events.

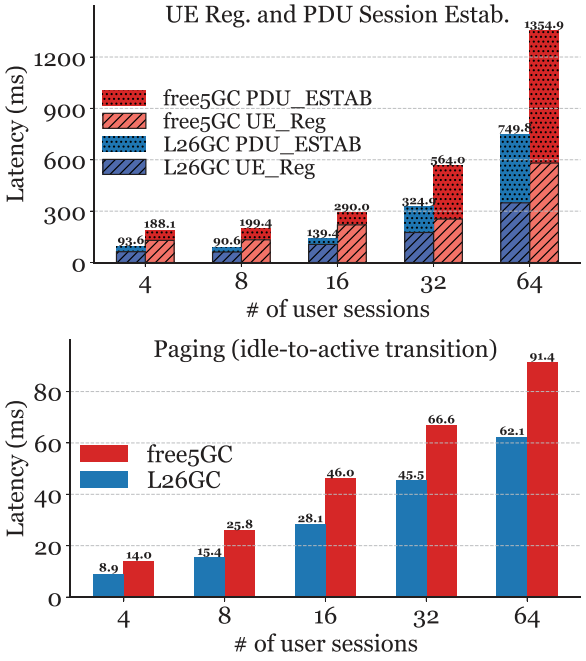

FIGURE 3. Completion time of various control plane events with concurrent user sessions (UEs). The values are averages of all UEs that were concurrently involved. PDU: packet data unit; Reg.: registration; Estab.: establishment.

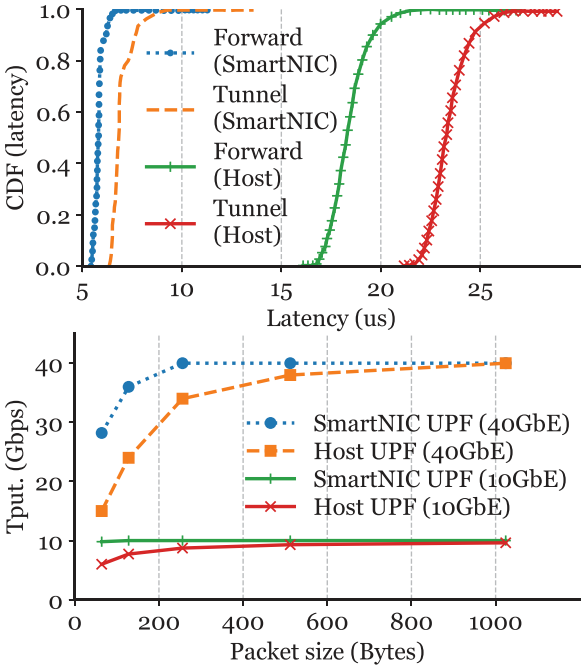

FIGURE 4. Performance of SmartNIC-based UPF in L^{2}6GC. CDF: cumulative distribution function; Gbps: gigabytes per second.

High-Performance UPF

Accelerating UPF With SmartNIC Offloading

We compare a SmartNIC-based UPF-U with a host-based UPF-U implemented as an OpenNetVM NF. We implement the SmartNIC-based UPF-U on the Netronome Agilio CX with 10-GbE and 40-GbE links.

Figure 4 shows packet processing latency for certain UPF operations: forwarding and GTP tunneling. Forwarding involves switching the packets from an input port to an output port after PDR lookup, while GTP tunneling additionally involves GTP header encapsulation. The SmartNIC UPF-U achieves up to 3.75× latency reduction due to the hardware acceleration for PDR lookup and GTP encapsulation. Processing packets on the SmartNIC eliminates the need to traverse the PCIe bus and host memory, thus avoiding the additional latency.

The SmartNIC-based UPF-U (40 GbE) also outperforms the host-based UPF-U (with a 40-GbE simple NIC and four CPU cores dedicated for the UPF-U), achieving 1.88× throughput improvement in handling 68-byte packets. Using our SmartNIC-based UPF-U leaves more CPU cores for the other colocated NFs. A 10-GbE SmartNIC-based UPF-U can achieve the line rate for all packet sizes, unlike a host-based UPF-U, which achieves the line rate only for larger packet sizes.

Fast, Scalable PDR Lookups

We compare the PDR classifier in L^{2}6GC's UPF (using PartitionSort[14]) with two alternatives: a linked-list-based classifier (LL) and a TSS-based classifier (TSS). We increase the number of PDRs inserted in the UPF to evaluate scalability. L^{2}6GC consistently achieves the lowest latency and highest throughput of all. Its performance shows negligible change with increased PDRs, while LL suffers from increasing latency and throughput loss as the number of PDRs increases. In addition, TSS shows considerable performance variability, impacted by the number of subtables partitioned from input PDRs. For more comprehensive results, refer to Jian et al.[1]

FUTURE DIRECTIONS

High-Performance Cross-Node Communication

Shared memory processing restricts the deployment of control plane NFs, requiring them to share the same memory pool on the same physical node. When control plane NFs are deployed on different nodes, our current implementation uses the kernel protocol stack for inter-node communication, resulting in increased latency. It is important to develop low-latency internode

communication methods, e.g., remote direct access memory (RDMA), to reduce this performance impact.

AI-Based Packet Classification

There are other intriguing packet classification approaches that use reinforcement learning (RL) to improve the speed of packet classification.[18] RL seeks to maximize long-term rewards to construct more efficient multidimensional classification trees compared to heuristic-based methods, thus minimizing PDR lookup delays.

CONCLUSION

L^{2}6GC aims to address the limitations of 3GPP 5GC in both the control and data planes, providing guidance for a future 6GC design. This includes a low-latency 3GPP SBI using shared memory processing, a SmartNIC-based UPF to enable high-performance data packet processing, and fast, scalable PDR lookups to support emerging 6G use cases. Future directions include an RDMA-based SBI to allow 6GC NFs to be deployed across multiple nodes, a low-latency software-based RAN, and AI-driven packet classification.

ACKNOWLEDGMENTS

This work was supported in part by U.S. NSF Grant CRI-1823270 and the National Science and Technology Council of Taiwan Grant 112-2218-E-A49-021, Grant 112-2218-E-A49-023, and Grant 111-2221-E-A49-093-MY3. We thank Han-Sing Tsai, Yu-Sheng Liu, Poyi Lin, and Sourav Panda for their help with implementation.

REFERENCES

1. V. Jain et al., "L25GC: A low latency 5G core network based on high-performance NFV platforms," in *Proc. ACM SIGCOMM 2022 Conf.*, New York, NY, USA: ACM, 2022, pp. 143–157, doi: 10.1145/3544216.3544267.
2. V. Jain, S. Panda, S. Qi, and K. K. Ramakrishnan, "Evolving to 6G: Improving the Cellular Core to lower control and data plane latency," in *Proc. 1st Int. Conf. 6G Netw. (6GNet)*, Piscataway, NJ, USA: IEEE, 2022, pp. 1–8, doi: 10.1109/6GNet54646.2022.9830519.
3. J. Chen, K. Ramakrishnan, A. Dhakal, and X. Ran, "Networked architectures for localization-based multi-user augmented reality," *IEEE Commun. Mag.*, vol. 61, no. 12, pp. 104–110, Dec. 2023, doi: 10.1109/MCOM.003.2300275.
4. M. Ghoshal et al., "Performance of cellular networks on the wheels," in *Proc. ACM Internet Meas. Conf.*, New York, NY, USA: ACM, 2023, pp. 678–695, doi: 10.1145/3618257.3624814.
5. K. Apicharttrisorn et al., "Characterization of multi-user augmented reality over cellular networks," in *Proc. 17th Annu. IEEE Int. Conf. Sens., Commun., Netw. (SECON)*, 2020, pp. 1–9, doi: 10.1109/SECON48991.2020.9158434.
6. S. Hakl. "*Improving the cloud for telcos: Updates of Microsoft's acquisition of AT&T's network cloud.*" Microsoft. Accessed: Mar. 1, 2024. [Online]. Available: https://azure.microsoft.com/en-us/blog/improving-the-cloud-for-telcos-updates-of-microsoft-s-acquisition-of-att-s-network-cloud/
7. W. Zhang et al., "OpenNetVM: A platform for high performance network service chains," in *Proc. Workshop Hot Topics Middleboxes Netw. Function Virtualization*, New York, NY, USA: ACM, 2016, pp. 26–31, doi: 10.1145/2940147.2940155.
8. S. Qi, H.-S. Tsai, Y.-S. Liu, K. Ramakrishnan, and J.-C. Chen, "X-IO: A high-performance unified I/O interface using lock-free shared memory processing," in *Proc. IEEE 9th Int. Conf. Netw. Softwarization (NetSoft)*, 2023, pp. 107–115, doi: 10.1109/NetSoft57336.2023.10175428.
9. Y.-S. Liu, S. Qi, P.-Y. Lin, H.-S. Tsai, K. K. Ramakrishnan, and J.-C. Chen, "L25GC+: An improved, 3GPP-compliant 5G core for low-latency control plane operations," in *Proc. IEEE 12th Int. Conf. Cloud Netw. (CloudNet)*, 2023, pp. 1–9.
10. Z. A. Qazi, M. Walls, A. Panda, V. Sekar, S. Ratnasamy, and S. Shenker, "A high performance packet core for next generation cellular networks," in *Proc. ACM SIGCOMM Conf.*, New York, NY, USA: ACM, 2017, pp. 348–361, doi: 10.1145/3098822.3098848.
11. R. MacDavid et al., "A P4-based 5G user plane function," in *Proc. ACM SIGCOMM Symp. SDN Res. (SOSR)*, New York, NY, USA: ACM, 2021, pp. 162–168, doi: 10.1145/3482898.3483358.
12. S. Panda, K. K. Ramakrishnan, and L. N. Bhuyan, "Synergy: A SmartNIC accelerated 5G dataplane and monitor for mobility prediction," in *Proc. IEEE 30th Int. Conf. Netw. Protocols (ICNP)*, 2022, pp. 1–12, doi: 10.1109/ICNP55882.2022.9940261.
13. V. Srinivasan, S. Suri, and G. Varghese, "Packet classification using tuple space search," in *Proc. Conf. Appl., Technol., Archit., Protocols Comput. Commun.*, Aug. 1999, pp. 135–146, doi: 10.1145/316188.316216.
14. S. Yingchareonthawornchai, J. Daly, A. X. Liu, and E. Torng, "A sorted-partitioning approach to fast and scalable dynamic packet classification," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1907–1920, Aug. 2018, doi: 10.1109/TNET.2018.2852710.
15. J. Larrea, A. E. Ferguson, and M. K. Marina, "CoreKube: An efficient, autoscaling and resilient mobile core system," in *Proc. ACM MobiCom Conf.*, New York, NY, USA: ACM, 2023, pp. 1–15, doi: 10.1145/3570361.3592522.

16. S. Kulkarni et al., "REINFORCE: Achieving efficient failure resiliency for network function virtualization based services," in *Proc. ACM CoNEXT Conf.*, New York, NY, USA: ACM, 2018, pp. 41–53, doi: 10.1145/3281411.3281441.
17. "free5GC." GitHub. Accessed: Mar. 1, 2024. [Online]. Available: https://github.com/free5gc/free5gc
18. E. Liang, H. Zhu, X. Jin, and I. Stoica, "Neural packet classification," in *Proc. ACM ACM Special Interest Group Data Commun.*, New York, NY, USA: ACM, 2019, pp. 256–269, doi: 10.1145/3341302.3342221.

SHIXIONG QI is a Ph.D. student in the Department of Computer Science and Engineering, University of California, Riverside, Riverside, CA, 92521, USA. His research interests include cloud computing, 5G, and network function virtualization. Qi received his M.Sc. degree in communication and information systems from Xidian University. Contact him at sqi009@ucr.edu.

K. K. RAMAKRISHNAN is a distinguished professor at the University of California, Riverside, Riverside, CA, 92521, USA. His research interests include software-based networks, cloud computing, cellular networks, and information centricity. Ramakrishnan received his Ph.D. degree from the University of Maryland, College Park. He is a Life Fellow of IEEE, an Association of Computing Machinery fellow, and an AT&T fellow. Contact him at kk@cs.ucr.edu.

JYH-CHENG CHEN is the chair professor with the Department of Computer Science and dean of the College of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, 30010, Taiwan. His research interests include wireless networks, mobile computing, and cellular networks. Chen received his Ph.D. degree from the State University of New York at Buffalo. He is a Fellow of IEEE and a distinguished member of the Association for Computing Machinery. Contact him at jcc@nycu.edu.tw.