Deciphering the Performance of Satellite UPF: Measurement and Enhancement

Yupeng Fu[*‡1], Jianing Si[*‡2], Xiao Ma[‡*3], Yuanzhe Li[†4], Yiran Zhang[*5]
[*]*Beijing University of Posts and Telecommunications,* [†]*Tsinghua University*, Beijing, China
[‡]Beiyou Shenzhen Institute, Shenzhen, China
[1]fyp@bupt.edu.cn, [2]sijianing@bupt.edu.cn, [3]maxiao18@bupt.edu.cn
[4]liyuanzhe@air.tsinghua.edu.cn, [5]yiranzhang@bupt.edu.cn

***Abstract*—User Plane Function (UPF), with the functions of local data dispatching, traffic management, and session anchoring, is a key facilitating component in core network and a critical enabling component in integration of communication and computation for future space computing power network (Space-CPN). With the rapid evolution of communications technology and Low Earth Orbit (LEO) satellites, both industry and academia are devoting efforts to exploring the necessity and feasibility of deploying 5G core network functions on LEO satellites. However, no prior work has studied in depth the performance of UPF in the space environment, especially the deployment of UPF on resource-constrained LEO satellites. This work presents the first measurement study on onboard UPF, revealing the main factor influencing the performance of onboard UPF and identifying the underlying bottleneck restricting throughput. Inspired by the key findings, this work further designs a high performance 5G UPF architecture for LEO satellites. Evaluation results have validated the effectiveness of the proposed design in improving the performance of onboard UPF.**

***Index Terms*—5G, LEO satellite, UPF, Measurement**

I. Introduction

As 5G technology progresses towards commercialization, there is a surging demand for Enhanced Mobile Broadband (eMBB) and Ultra Reliable Low Latency Communications (URLLC) applications, such as 4K/8K video streaming [1] and cloud gaming [2]. These applications require networks to achieve extremely high bandwidth and low latency, necessitating User Plane Functions (UPFs) to enforce considerably high performance in data handling and transmission. In 5G networks, UPF performs the functions of local data dispatch, traffic management, and session anchoring, making it a key facilitating component of the core network and enabling the integration of communication and computation (e.g. edge computing) in 5G [3]. Current 5G infrastructure is primarily deployed in densely populated terrestrial regions, leaving suburban, remote, and other non-terrestrial areas such as maritime and aerial zones not sufficiently covered. The forthcoming 6G paradigm is envisioned to integrate terrestrial, aerial, and space networks (forming a space-air-terrestrial integrated architecture), and to support integration of communication and computation capabilities in space networks (i.e., space computing power network, Space-CPN). To realize the visions of the 6G paradigm, deploying UPF on satellites and enhancing the performance of satellite UPF is one of the key researching topics.

Deploying UPF on satellites is beneficial for Space-CPN for two reasons. First, as the anchor for user plane data forwarding (i.e., PDU Session Anchor), deploying UPF on satellites is conducive to directly provision computing services on satellites without experiencing satellite-ground data transmission, thus lowering the delay of space computing. Secondly, deploying UPF on satellite can distribute traffic between the satellite and the terrestrial network (UPF with UP-link Classifier), the space-native data traffic can be dispatched to satellite computing devices, which can significantly help relieve the burden of the costly and resource-scarce satellite-terrestrial links. Yet, deploying UPF on satellites and enhancing the performance of satellite UPF is far from well investigated.

Industry and academia are actively promoting and investigating the necessaries and feasibility of deploying 5G core network functions on LEO satellites. The 3rd generation partnership project (3GPP) is extending 5G (and beyond) to LEO constellations and includes the scenario of UPF deployment on satellites in standardizations, such as 3GPP R18 [4]–[6]. The National Aeronautics and Space Administration (NASA) has deployed a spaceborne core network to validate the potential of 5G capabilities on satellites [7]. Operators and infrastructure vendors have conducted experiments by hosting core network functions on LEO satellites [8], [9]. Researchers have also implemented experiments and emulations to study the feasibility of deploying 5G core network on satellites [10] and demonstrate the advantages of satellites as part of transmission networks [11]. The problem of signaling storms in the satellite core network has been pointed out and a stateless design for the satellite core network prototype has been developed [12].

Despite of the extensive efforts devoted to the integration of 5G core network with satellites, they mostly did not study in depth the performance of satellite core network quantitatively, especially for satellite UPF which has been under standardization. As indicated by the measurement study on satellite computing, the capability of onboard computing is significantly influenced by factors including temperature, energy, and resource capacities [13]. These findings suggest the importance of conducting a similar study on onboard UPF performance. To this end, this study presents the first-of-its-kind measurement study on onboard UPF. A simulation platform of satellite core network has been constructed on a one-to-one scale, with two servers simulating user access network and control plane,

and four Raspberry Pis performing the functions of UPF and data network. The results demonstrate that temperature notably surpasses energy and resource limitations as the primary determinant of onboard UPF performance. Additionally, it is also identified that Vanilla UPF (Free5GC) implementations with kernel data handling are bottlenecked by frequent CPU interruptions and extensive memory operations when deployed on lightweight satellite computing equipment (e.g., Raspberry Pi), which considerably undermines network throughput.

Motivated by the above findings, we further propose a xdp-based UPF design for Raspberry Pi (i.e., XupfPI) to enhance the performance of satellite UPF. The measurement results indicate that enhancing satellite UPF performance necessitates minimizing transitions between user and kernel modes. Inspired by this finding, we propose an onboard adaptation of UPF functionalities designated for a satellite payload environment. To adapt to resource-restricted environments, we redesign the network address translation functionalities of the protocol stack. In specific, we employ direct addressing techniques for optimizing the data packet processing trajectory and leverage pointer offset matching for the segmented retrieval of crucial information, facilitating routing decisions. This design can accelerate the modification of field positions and eliminate the need for data packet duplication, which thereby can well support high network throughput. Experimental validations confirm that our proposed solution significantly enhances the network performance of onboard UPF systems.

Our main contributions can be summarized as follows:

- This work presents the first-of-its-kind measurement study on onboard UPF. The results demonstrate that temperature is the main factor limiting the performance of onboard UPF. Additionally, it is also identified that frequent CPU interruptions and extensive memory operations considerably undermines network throughput.
- Inspired by the key findings of the measurement, this work designs a high performance 5G UPF for the LEO satellite. The ability to directly process network card packets within the kernel expedites packet handling, minimizing the overhead of kernel-user mode transitions.
- The evaluation results have demonstrated the effectiveness of the proposed design in improving the performance of onboard UPF.

II. RELATED WORK

Satellite Performance Measurement. The measurement study on onboard computing [13] highlighted the significant influence of temperature, power consumption, and resource constraints on computational efficiency. The work [14] measured the performance of the Starlink system from the perspective of ordinary end users, revealing that the service quality of Starlink users is more susceptible to environmental conditions. Moreover, many research initiatives have been dedicated to the empirical examination of satellite systems, aiming to find out the determinants influencing satellite performance [15].

UPF Acceleration Research. UPF is identified as a pivotal component within the user plane, where its capability to deliver high throughput and minimal latency services is contingent upon its performance. Strategies to enhance UPF performance are diverse, which are mainly summarized into the following three categories: (1) using P4 programmable hardware to achieve performance acceleration [16]; (2) using DPDK acceleration framework to improve packet processing [17], [18]; (3) using XDP technology to achieve performance acceleration [19].

III. BACKGROUND

5G Core Network. The 5G core network architecture employs a separation of the control plane and the user plane [20], as is depicted in Fig. 1. UPF is essential for achieving low latency and high bandwidth, as discussed by [16]. Fig. 1 illustrates the UPF's functionality as an Intermediate UPF (I-UPF) when situated amongst multiple Protocol Data Unit (PDU) session anchors. In this capacity, the UPF serves as a Uplink Classifier (UL-CL) and a branching point to facilitate multi-homed PDU sessions, directing traffic to specific Data Networks (DNs) based on traffic matching filters. UPF is a critical component in the routing and forwarding of user plane data within the 5G core network system. The operational workflow of UPF data packet forwarding predominantly involves the Packet Forwarding Control Protocol (PFCP). Initially, the User Equipment (UE) requests PDU session establishment, prompting the Access and Mobility Management Function (AMF) to designate an appropriate SMF for session management, based on the specifics of the request. For a UE that seeks to build a session to the data network, its packets should be transmitted through a UPF which is selected by the SMF. Communication between SMF and UPF is facilitated via the PFCP through the N4 interface, where the SMF directs UPF packet forwarding operations by issuing a PFCP Session Establishment request. Upon successful establishment, a PFCP Session Context is generated within the UPF which stores the UE's identity, Packet Detection Rules (PDR), Forwarding Action Rules (FAR), and other information. The forwarding of user plane data packets is executed in accordance with this context.

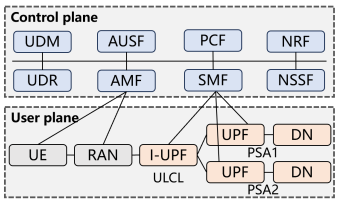


Fig. 1: 5G system architecture.

Temperature and Power on Satellites. The complex and harsh satellite environment poses severe challenges to the stable operation of Commercial Off-The-Shelf (COTS) equipment. The study [13] presented the first measurement study on the thermal control, power management, and performance of COTS devices on satellites. The findings emphasize the critical role of temperature and power consumption as

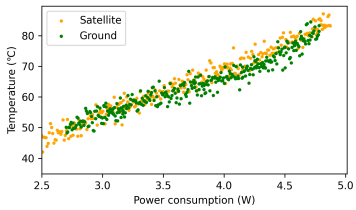


Fig. 2: Power-Temperature Curve between satellite platform and simulation platform.

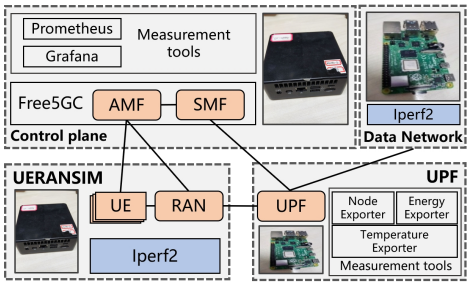


Fig. 3: Overview of the simulation platform.

primary constraints that markedly restrict the capacity of the computational workload and the operational duration of onboard COTS devices. Specifically, it was demonstrated that sustained operation of the Atlas 200 DK under maximum load causes adverse effects on the satellite's surface temperature and battery longevity. Conversely, Raspberry Pi with low power consumption causes a comparatively minimal impact on satellite systems, highlighting the importance of power efficiency in the selection of COTS devices for space missions.

IV. Performance Measurement

This section studies the performance constraints of satellite core networks, highlighting the complexities introduced by the space environment.

A. Setup and Methodology

We constructed a terrestrial simulation platform (see Fig.3) that mirrors the architecture of COTS equipment within satellite frameworks on a one-to-one scale. Using empirical data from the satellite platform [13], we emulated the environmental conditions and computational capabilities characteristic of the onboard systems. As illustrated in Fig.2, we accurately simulated the thermal dissipation and power performance of low-power devices in the true space environment through a series of advanced physical thermal isolation methods. The instrumentation comprises two Intel NUC 11th Generation Intel® Core™ i5-1135G7 @ 2.40GHz servers operating Ubuntu 22.04 with kernel version 5.15.0-67-generic. One server is dedicated to hosting the network functions of the Free5GC control plane, while the other facilitates UERANSIM [21], an open-source User Equipment and Radio Access Network simulator. Additionally, the setup includes four Raspberry Pi 4Bs, each with 8GiB RAM. Three serve as UPF and one as a Data Network. We conducted three parallel sets of identical experiments to ensure the generalizability of the experimental results. Measures were taken to ensure that the initial temperature discrepancy of each experimental iteration remained below 2°C. The room and Raspberry Pi standby temperatures for these experiments were maintained at 27°C and 50°C, respectively.

Our methodology primarily employs Prometheus Exporter and iPerf2 (UDP) for the acquisition of UPF metrics, encompassing temperature, power consumption, CPU utilization, and packet loss rate. To ensure reliability and stability, each set of

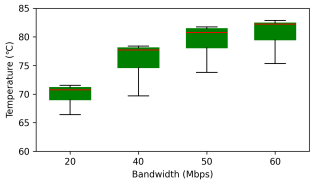


Fig. 4: Correlation between bandwidth and temperature.

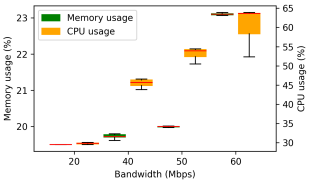


Fig. 5: Correlation between bandwidth and resource.

experiments was repeated for 10 times and the measurement duration was uniform.

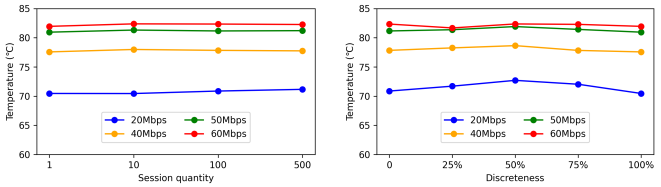
B. Key Findings

1) Temperature is the Main Factor Limiting the Performance of Onboard UPF: Experiments show that the bottleneck of the Raspberry Pi onboard is temperature rather than power consumption. The Raspberry Pi consumes about 5W at full load, and there is little difference between ground and space [13]. The total capacity of the satellite battery is 230Wh. This is significantly more than the needs of the Raspberry Pi. Fig.4 illustrates that the high transmission bandwidth may cause the Raspberry Pi to overheat, thus causing irreversible hardware damage.

We examined CPU and memory utilization to determine the cause of temperature elevation. Fig.5 reveals a proportional increase in both CPU and memory utilization with increasing bandwidth, with CPU utilization demonstrably more affected. This pattern indicates that a higher data transfer bandwidth precipitates increased CPU utilization, which then increases temperature levels. Upon surpassing a predefined temperature threshold, the Raspberry Pi engages the Dynamic Voltage and Frequency Scaling (DVFS) mechanism, decrementing operational frequency to mitigate heat accumulation, potentially impairing network performance.

Further exploration reveals a critical insight: the total transmission bandwidth significantly impacts system temperature, overshadowing the effects of session quantity or bandwidth allocation. This conclusion is supported by two principal observations:

- **Session quantity:** Investigating a scenario with a 20Mbps transmission bandwidth evenly distributed across varying numbers of sessions, from a solitary session to as many as 500, preliminary data (illustrated in Fig.6a)



(a) Session quantity (b) Bandwidth allocation

Fig. 6: Impact of session quantity and bandwidth allocation on temperature.

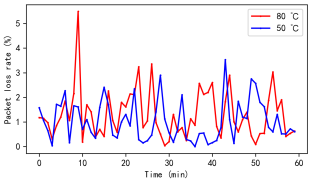


Fig. 8: Correlation between temperature and packet loss rate.

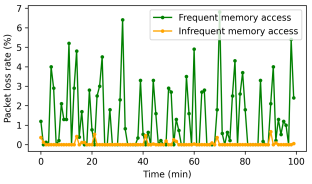
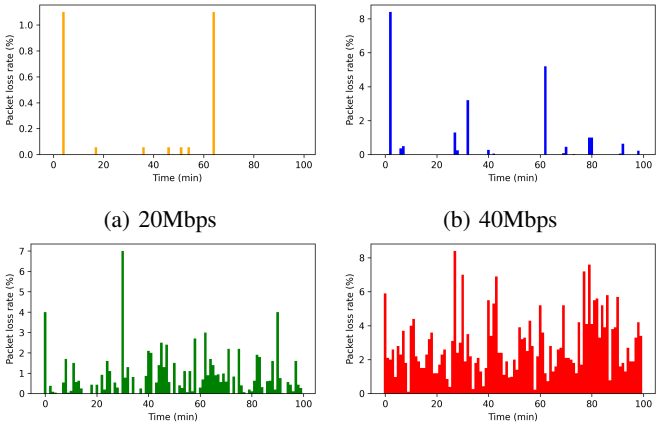


Fig. 9: The impact of frequent memory access on packet loss rate.



(a) 20Mbps (b) 40Mbps

(c) 50Mbps (d) 60Mbps

Fig. 7: Correlation between packet loss rate and bandwidth in Vanilla UPF.

indicate that the system's temperature remains stable across different session counts.

- **Bandwidth allocation:** In examining the impact of bandwidth distribution within the 20Mbps transmission bandwidth, experiments were carried out adjusting the bandwidth allocation from 100% (single session using the entire bandwidth) to 0% (bandwidth uniformly distributed between 100 sessions). We can see from Fig.6b that minimal temperature variations irrespective of the bandwidth distribution strategy, given a constant total bandwidth.

2) High Load Induces Significant Packet Loss in Vanilla UPF: Experiments show that under the same measurement duration, the increase of data transmission bandwidth has less impact on the delay, but leads to higher packet loss. As shown in Fig.7, Vanilla UPF experiences significant packet loss at 60Mbps, adversely affecting core network functionality and service quality.

Initially, temperature was hypothesized to be a principal factor contributing to packet loss. However, experiments under various thermal conditions revealed minimal reduction in packet loss rates with a decrease in temperature, as illustrated in Fig.8. To find out the primary cause of packet loss, a sequence of controlled experiments was undertaken, including kernel adjustments, driver updates, and alterations to the operating system, yet these investigations yielded no significant discoveries. A fortuitous insight emerged through the use of Stress-ng, indicating that competition for memory access could impair packet processing efficiency, especially under high-frequency memory utilization conditions. Comparative analyses conducted between frequent versus sparse memory access scenarios, as shown in Fig.9, identified a pronounced increase in packet loss rate under frequent memory access conditions. This emphasizes the influence of memory access patterns on network performance. Subsequent validation through Sysbench confirmed memory bandwidth as the predominant factor impacting network performance disparities.

C. Key Insights

The results of our measurement emphasize a pivotal consideration for the design and optimization of UPF: data transmission mechanisms that rely on kernel intervention induce extra interrupts and memory copy overheads. Consequently, when designing UPF, the processing method of data packets should be fully considered, and the kernel should be avoided as much as possible.

V. DESIGN AND IMPLEMENTATION

A. Design

The system is designed to provide the best network quality of service with the normal operation of the satellite core network. As detailed in section IV, packets processed through the Linux kernel's network protocol stack will not only result in additional packet loss but also cause an increase in satellite temperature. When the core network is overloaded, performance is severely impaired. To address these challenges, this paper proposes a novel protocol stack independent data packet parsing mode suitable for satellite scenarios. Using centralized management concepts, we devised an efficient UPF routing component called XupfPI. This component is designed to intermittently capture essential information via pointer displacement, facilitating swift routing decisions that conserve satellite resources.

The workflow of XupfPI encompasses interactions between the kernel and user spaces. Given these considerations, XupfPI is segmented into two subsystems: the control subsystem of the user space and the data acceleration subsystem of the kernel space. The framework of XupfPI is shown in Fig.10.

The user space control subsystem in this design intercepts packets at the UPF network interface ingress, utilizing the

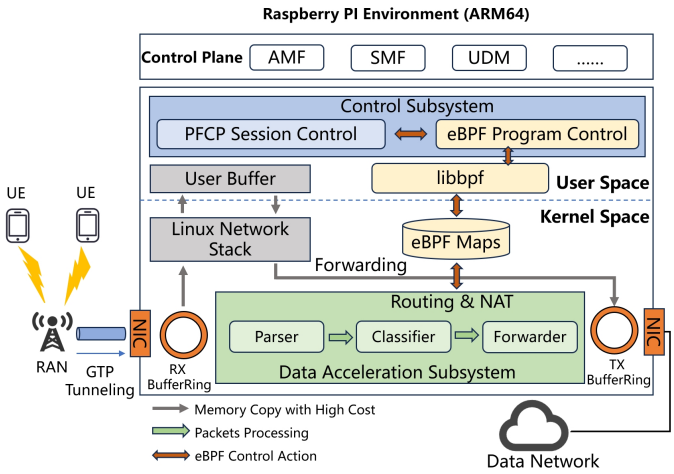


Fig. 10: Overview of XupfPI architecture.

mapping capabilities of the extended Berkeley Packet Filter (eBPF) to organize forwarding data within the kernel's forwarding and flow tables. This subsystem is structured around two main components: the PFCP session controller and the eBPF Program controller. These components are responsible for the lifecycle management of PFCP sessions and eBPF programs, respectively. PFCP session management encompasses several steps: upon a data packet's arrival at the user plane, XupfPI initially seeks a corresponding PFCP session via the PDU session (F-TEID). Upon successful identification, XupfPI evaluates all related Packet Detection Rules (PDRs), selecting the one with the highest priority. Then XupfPI identifies the Forwarding Action Rules (FARs), Buffering Action Rules (BARs), QoS Enforcement Rules (QERs), and Usage Reporting Rules (URRs) associated with the selected PDR. Subsequently, XupfPI applies the identified PDR association rule to the packet, forwarding it according to the specified FAR. eBPF lifecycle management involves the invocation of eBPF Programs during the PFCP session management to address specific events. Before calling the eBPF program, the bytecode of the program needs to be dynamically loaded into the kernel through the libbpf library. eBPF programs loaded into the kernel can access eBPF Maps, which store information about PDRs and FARs and are used to share data between kernel space and user space. Upon receiving network data structures, the PFCP session controller utilizes the libbpf library to decode these structures into eBPF map entries, subsequently updating the eBPF maps with these entries. This process enables the kernel's eBPF program to access the latest data, performing operations based on the PDRs and FARs.

The data acceleration subsystem in kernel space is intricately designed for efficient parsing of GPRS Tunneling Protocol (GTP) headers in UE packets. This enables the subsystem to adeptly process and route packets based on predefined forwarding rules, which utilize the Tunnel Endpoint Identifier (TEID) along with the packet's source and destination IP addresses for rapid decision-making. The subsystem is structured into three principal components: the Parser, the Classifier, and the Forwarder. The parser's role is to analyze incoming data packets, extracting essential information for further processing. Following this, the Classifier consults eBPF Maps to identify the PDR corresponding to the packet. Upon successful identification, the packet is relayed to the Forwarder. The Forwarder then utilizes the PDR data to locate the FAR, executing the necessary packet forwarding decision. Since NAT is a function provided by the Linux kernel network stack, and some data packets processed by XupfPI will cross the kernel network stack, we also implemented an additional external proxy forwarding function in Forwarder to translate private IP addresses into public ones, facilitating communication between internal network hosts and the external network.

B. Implementation

XupfPI is built on the open-source 5G core project Free5GC (version 3.3.0) and is deployed on Raspberry Pi which has important applications in LEO satellites [13]. However, limited by its hardware, the XupfPI was unable to reach its full potential. The system is mainly implemented in Go, but XupfPI is specially written in C. The XupfPI program can be roughly divided into the following three parts: parsing received data packets, reconstructing the data packets through address conversion, and forwarding the data packet according to the rules. Data packets are made up of IP, UDP, and GTP packets. Upon reception, XupfPI initially deconstructs a data packet into an IP packet, subsequently parsing this into a UDP packet. Only when the UDP port corresponds to 2152 is the packet identified as a GTP packet, requiring specialized processing. In contrast, packets not matching this criterion are processed via the Linux kernel network protocol stack. XupfPI further parses the data packet with port number 2152 into a GTP packet and obtains the original message information. Since we want to access the Data Network, the data packets processed by XupfPI do not pass through the Linux kernel network protocol stack, so the NAT function is not available. In order to have the ability to access the public network, we reimplemented the function and reconstructed the data packets. Ultimately, XupfPI forwards the restructured packet following the determined FAR.

VI. EVALUATION & ANALYSIS

In this section, we conducted a series of experiments to evaluate the performance of XupfPI under high-load scenarios, measuring its superiority in packet loss and temperature, among others, compared to Vanilla UPF which was inapplicable at 60Mbps bandwidth (refer to Section IV-B2) using Iperf.

A. Experiment Setup

As illustrated in Fig.3, consistent with previous experiments, the testing platform was similarly divided into four segments. Among these, the UPF segment, earmarked for critical testing, employed either XupfPI or Vanilla-UPF deployed on Raspberry Pi 4Bs with 8GiB RAM. The remaining components maintained alignment with the previous descriptions: UERANSIM and Free5GC control plane were individually deployed

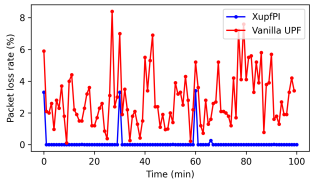


Fig. 11: Comparison of packet loss rate.

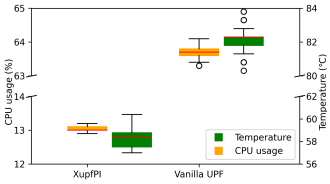


Fig. 12: Comparison of resource.

on two Intel NUC 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz servers, running on Ubuntu 22.04 with kernel 5.15.0-67-generic. The DN segment utilized the same hardware as the UPF segment.

B. Performance

We initially verified the packet loss rates of XupfPI and Vanilla UPF based on Free5GC under scenarios of high system load, by controlling a UE with Iperf to continuously transmit at a bandwidth of 60Mbps through the UPF to the DN. As depicted in Fig.11, XupfPI demonstrates a notable outperformance in packet loss rates, achieving a qualitative leap in service availability time. This advantage probably stems from the kernel-level packet processing techniques. Packets are handled prior to stack entry, thereby avoiding the necessity of constructing a sk_buff (used to manage and control receiving or sending data packets) structure that heavily consumes Linux memory. Moreover, circumventing the complex Linux network stack directly leads to reduced data packet processing times and eliminates the risk of processing delays.

As can be seen in Fig.12, there is a big difference in CPU overhead and temperature changes between Vanilla UPF and XupfPI under the same data transmission bandwidth of 60Mbps. Vanilla UPF implemented atop the kernel protocol stack, encounters high-frequency kernel interrupts when processing high-concurrency streaming data packets. This will lead to recurrent switching between user mode and kernel mode, incurring additional context switching overheads. Such overheads not only result in increased CPU usage and temperature rise, but also become more pronounced in satellite core network scenarios due to the extreme thermal conditions. Consequently, this exacerbates system strain, leading to a vicious cycle of escalating temperatures and overhead.

VII. Conclusions

User Plane Function (UPF) is a key facilitating component in core network and a critical enabling component in integration of communication and computation for future space computing power network (Space-CPN). In this paper, we first present the first-of-its-kind measurement study on satellite UPF and identify critical factors that affect the performance. The results show that temperature and frequent CPU interruptions lead to performance degradation. To address such issues, we designed and implemented XupfPI, targeting spaceborne low-power devices, achieving fast packet processing. The experimental results compellingly demonstrate that our XupfPI, significantly reduces the packet loss rate and enhances network service quality.

Acknowledgements

This research was supported by the Science and Technology Foundation of Shenzhen (No. KJZD20230928112759002) and the National Natural Science Foundation of China (No. 62372061 and 62032003). Xiao Ma is the corresponding author of this work.

References

[1] Ghufran Baig, Jian He, Mubashir Adnan Qureshi, Lili Qiu, Guohai Chen, Peng Chen, and Yinliang Hu. Jigsaw: Robust live 4k video streaming. In *MobiCom*, pages 1–16, 2019.

[2] Chun-Ying Huang, Kuan-Ta Chen, De-Yu Chen, Hwai-Jung Hsu, and Cheng-Hsin Hsu. Gaminganywhere: The first open source cloud gaming system. *TOMM*, 10(1s):1–25, 2014.

[3] Stefan Rommer, Peter Hedman, Magnus Olsson, Lars Frid, Shabnam Sultana, and Catherine Mulligan. *5G Core Networks: Powering Digitalization*. Academic Press, 2019.

[4] 3GPP. TR38.811: Study on New Radio (NR) to support non-terrestrial networks, 2020.

[5] 3GPP. TR38.821: Solutions for NR to support non-terrestrial networks (NTN), 2020.

[6] 3GPP. TS23.501: System architecture for the 5G System (5GS), 2020.

[7] Expanding network capability on the ISS with a 5G core prototype. https://www.hpe.com/us/en/newsroom/press-release/2022/04/hewlett-packard-enterprise-drives-innovation-at-the-extreme-edge-on-the-international-space-station-with-24-completed-experiments.html, 2022.

[8] 5G-Advanced core experiments in the Baoyun LEO Satellite by China Mobile. https://www.c114.com.cn/news/118/a1183668.html, 2021.

[9] AST SpaceMobile. https://ast-science.com/spacemobile/, 2021.

[10] Ruolin Xing, Xiao Ma, Ao Zhou, Schahram Dustdar, and Shangguang Wang. From earth to space: A first deployment of 5g core network on satellite. *China Communications*, 20(4):315–325, 2023.

[11] Armir Bujari, Michele Luglio, Claudio E Palazzi, Mattia Quadrini, Cesare Roseti, and Francesco Zampognaro. A virtual pep for web optimization over a satellite-terrestrial backhaul. *IEEE Communications Magazine*, 58(10):42–48, 2020.

[12] Yuanjie Li, Hewu Li, Wei Liu, Lixin Liu, Yimei Chen, Jianping Wu, Qian Wu, Jun Liu, and Zeqi Lai. A case for stateless mobile core network functions in space. In *SIGCOMM*, pages 298–313, 2022.

[13] Ruolin Xing, Mengwei Xu, Ao Zhou, Qing Li, Yiran Zhang, Feng Qian, and Shangguang Wang. Deciphering the enigma of satellite computing with cots devices: Measurement and analysis. *arXiv*, 2024.

[14] Sami Ma, Yi Ching Chou, Haoyuan Zhao, Long Chen, Xiaoqiang Ma, and Jiangchuan Liu. Network characteristics of leo satellite constellations: A starlink-based measurement from end users. In *INFOCOM*, pages 1–10, 2023.

[15] Daniel Perdices, Gianluca Perna, Martino Trevisan, Danilo Giordano, and Marco Mellia. When satellite is all you have: watching the internet from 550 ms. In *IMC*, pages 137–150, 2022.

[16] Abhik Bose, Shailendra Kirtikar, Shivaji Chirumamilla, Rinku Shah, and Mythili Vutukuru. Accelupf: Accelerating the 5g user plane using programmable hardware. In *SOSR*, pages 1–15, 2022.

[17] Madhura Adeppady, Mohit Kumar Singh, and Bheemarjuna Reddy Tamma. Onvm-5g: a framework for realization of 5g core in a box using dpdk. *CSI Transactions on ICT*, 8(1):77–84, 2020.

[18] Haoran Zhang, Zikang Chen, and Yang Yuan. High-performance upf design based on dpdk. In *ICCT*, pages 349–354, 2021.

[19] Jianer Zhou, Zengxie Ma, Weijian Tu, Xinyi Qiu, Jingpu Duan, Zhenyu Li, Qing Li, Xinyi Zhang, and Weichao Li. Cable: A framework for accelerating 5g upf based on ebpf. *Computer Networks*, 222:109535, 2023.

[20] Vivek Jain, Hao-Tse Chu, Shixiong Qi, Chia-An Lee, Hung-Cheng Chang, Cheng-Ying Hsieh, KK Ramakrishnan, and Jyh-Cheng Chen. L25gc: A low latency 5g core network based on high-performance nfv platforms. In *SIGCOMM*, pages 143–157, 2022.

[21] UERANSIM: Open source 5G UE and RAN (gNodeB) implementation. https://github.com/aligungr/UERANSIM.