Power Efficiency of a Hybrid 5G gNB Data Plane Combining SmartNICs and Commodity Servers

Mohsen Memarian*, Andreas Kassler*†, Karl-Johan Grinnemo*, Sándor Laki‡, Gergely Pongracz§, Johan Forsman¶
*Computer Science Department, Karlstad University, Karlstad, Sweden
†Institute of Applied Computer Science, Deggendorf Institute of Technology, Deggendorf, Germany
‡Faculty of Informatics, ELTE Eötvös Loránd University, Budapest, Hungary
§Ericsson Research, Budapest, Hungary
¶TietoEvry, Umeå, Sweden
Email: *mohsen.memarian@kau.se, andreas.kassler@kau.se, karlgrin@kau.se †andreas.kassler@th-deg.de,
‡lakis@inf.elte.hu, §gergely.pongracz@ericsson.com, ¶johan.forsman@tietoevry.com

***Abstract*—Power efficiency is crucial in softwarized 5G gNBs due to rising energy costs and CO_2 emissions from the networking infrastructure. SmartNICs have the potential to accelerate packet processing tasks. However, their limited resources and functionalities make handling complex tasks like retransmissions challenging. Offloading operations to server-class processors can improve scalability but also increase power consumption, warranting careful management. This study examines the performance and power consumption of splitting the processing of the 5G gNB between SmartNICs and DPDK-enabled servers. We implement an adaptive CPU power conserving strategy in the DPDK data plane that dynamically adjusts CPU power states using P-states and C-states based on traffic load. Our evaluation shows that when using five SmartNICs, we can achieve 95 MPPS, demonstrating that offloading can boost throughput by up to 40% for gNB header processing. Our adaptive power management limits CPU power consumption growth to a maximum of 11%, compared to 52% without such management, balancing performance and power efficiency effectively.**

***Index Terms*—5G gNB, P4, Data Plane, SmartNIC**

I. Introduction

5G networks face unprecedented data volumes and require ultra-low latency. The power efficiency of packet processing is a key factor, as the network infrastructure significantly contributes to the cost and CO_2 footprint of the operator. The ultra-lean design of 5G NR improves energy usage by reducing mandatory signaling. However, gNB functionalities can be run on commodity servers with general-purpose CPUs, which are not optimized for low power consumption. SmartNICs can take on network-centric tasks like packet forwarding, and utilizing dedicated hardware to lower overall power consumption. Yet, not all gNB pipeline tasks can be effectively accelerated on SmartNICs due to their limited resources and packet processing functionalities. For example, complex functions such as buffering in the Radio Link Control (RLC) layer or retransmitting lost packets are challenging to implement on SmartNICs and are typically offloaded to the host.

Recent research has explored SmartNICs for power efficiency. [1] compares offloading microservices to SmartNICs with performing them on the host DPDK, but overlooks DPDK power-aware packet polling. [2] introduce an RSS-based load balancer for SmartNIC-to-host paths with dynamic voltage and frequency scaling but rely on simulations. [3] propose a load-balancing framework with DPDK on SmartNICs to dynamically distribute packet processing, though they don't tackle host CPU power management. [4] focuses on DPDK power optimizations without integrating SmartNICs. [5] analyzes 5G UPF power savings through DPDK but misses the combined effects of SmartNICs and host power optimizations.

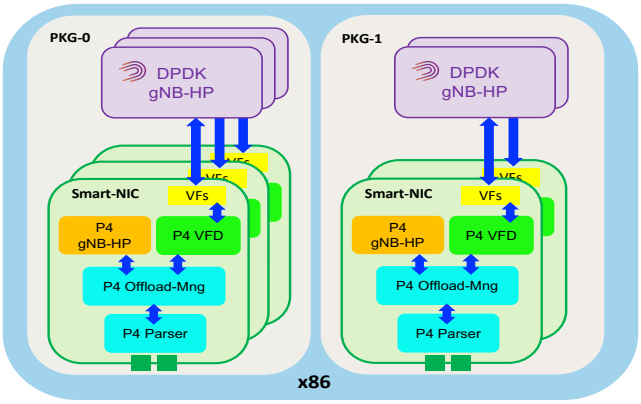

Fig. 1: 5G gNB header processing design uses SmartNICs and x86.

In our previous work [6], we investigated the additional throughput gained by offloading 5G gNB tasks from a single SmartNIC to the host. Building on that foundation, this paper expands the setup to five SmartNICs, achieving 95 MPPS, and examines whether the additional offloading remains advantageous regarding power efficiency, especially given the dependency on power-hungry general-purpose CPUs. We explore techniques to enhance CPU power efficiency while maintaining the high performance required by 5G gNB. Our investigation compares traditional DPDK busy-wait packet polling with a power-aware approach that employs a heuristic algorithm to adjust CPU power states dynamically based on traffic load. This method utilizes active states (P states) for frequency scaling, increasing CPU frequency when RX queue

occupancy surpasses certain thresholds, and incorporates idle states (C states) by transitioning cores into low-power modes using pause and sleep mechanisms when repeated polling cycles yield no packets. Our experimental results demonstrate that offloading can boost throughput by up to 40% for gNB header processing. Our adaptive power management limits the total power consumption increase to a maximum of 11%, rather than the potential 52%, effectively balancing performance and power efficiency.

II. Design and Implementation

Figure 1 illustrates the architecture of our 5G gNB packet processing pipeline. Three SmartNICs are connected to CPU Package 0 (PKG-0), while the remaining two are connected to CPU Package 1 (PKG-1). The P4 Parser first extracts packet headers. The P4 Offload Manager then categorizes packets into two groups based on a marker in the IPv4 Identification field, indicating their intended gNB processing target. Packets in the first group are processed by a dedicated P4 program for gNB header processing that is compiled and deployed on each SmartNIC, while packets in the second group pass through the P4-VFD (P4 Virtual Function Distributor), which evenly distributes them among the VFs which are bound to the host DPDK applications. Building upon our previous work [6], both the SmartNIC and the host adhere to a specified packet header processing path. A 64K-entry Match-Action Table (MAT) or hash table identifies the corresponding Data Radio Bearers (DRBs) based on flow attributes, such as the GTP Tunnel Endpoint Identifier (TEID). In the downlink path, packet processing begins with GTP decapsulation and adds Service Data Adaptation Protocol (SDAP), Packet Data Convergence Protocol (PDCP), and RLC headers.

Our heuristic (see Algorithm 1) dynamically adjusts the active states, known as P states, and idle states, referred to as C states, within the DPDK data plane to achieve a balance between performance and power efficiency. It monitors incoming packet loads and scales to higher active states, utilizing 13 distinct levels that range from P0 (above 2 GHz), to P12 (800 MHz), when additional processing power is required. Simultaneously, the algorithm evaluates idle conditions to determine when to issue "idle hints" which trigger the CPU to transition into the appropriate idle state.

In terms of idle states, C0 represents full activity, C1 indicates a halt state that allows for rapid wake-up, while deeper states like C3 and C6 offer more power savings at the cost of a minimum wake-up latency of 100 μs; therefore, we prioritize short pauses over longer sleep states. Thresholds are tuned by comparing queue occupancy against a DPDK busy-wait mode; increased occupancy indicates that the algorithm is overly conservative, and aggressive tuning is avoided to prevent unnecessary increases in CPU active time or frequency.

We assign processing cores to manage the receive (RX) and transmit (TX) queues of one or more VFs, each equipped with a single RX and TX queue. Each core continuously monitors its RX queues for incoming packets. If no packets are detected, the core increments an empty poll counter. If this counter

Algorithm 1: DPDK Per-Core Packet Polling with Integrated Active and Idle Power Management

```
while true do
    lcore_max_freq_hint, lcore_min_idle_hint ← 0;
    foreach rxq in RX queues of the lcore do
        if READ rxq.packets = 0 then
            rxq.empty_poll_cnt ← rxq.empty_poll_cnt + 1;
            if rxq.empty_poll_cnt ≥ 10 then
                if rxq.empty_poll_cnt < 300 then
                    rxq.idle_hint ← 1 µs;
                else
                    rxq.idle_hint ← 300 µs;
        else
            rxq.empty_poll_cnt ← 0;
            READ rxq_packet_cnt;
            if rxq_packet_cnt > 96 then
                rxq.freq_hint ← 2; rxq.trend ← 0;
            else if rxq_packet_cnt > 64 then
                rxq.trend ← rxq.trend + 250;
            else if rxq_packet_cnt > 32 then
                rxq.trend ← rxq.trend + 3;
            if rxq.trend ≥ 10,000 then
                rxq.freq_hint ← 1; rxq.trend ← 0;
    lcore_min_idle_hint ← min_{rxq∈lcore queues} (rxq.idle_hint)
    lcore_max_freq_hint ← max_{rxq∈lcore queues} (rxq.freq_hint)
    if lcore_max_freq_hint = 2 then
        STEP to highest P-state;
    else if lcore_max_freq_hint = 1 then
        STEP to next P-state;
    if lcore_min_idle_hint < 300 then
        PAUSE for lcore_min_idle_hint µs;
    else
        TURN_ON_INTERRUPTS; GO_TO_SLEEP;
        while true do
            if RXQ_EVENT then
                WAKE_UP; TURN_OFF_INTERRUPTS;
                break;
            Wait for 10 ms;
```

reaches 10, it issues an idle hint that is either 1 or 300 μs. When packets arrive, the core checks the packet count in the RX queue. If the count exceeds predefined limits, the algorithm may either accumulate a trend or directly issue a frequency hint.

After processing the RX queues, the core determines the minimum idle hint and the maximum frequency hint across all queues, generating a consolidated hint for the core. Based on this hint, the system either increases the frequency or shifts to an idle state. If the idle hint is less than 300 μs, the core performs a pause to avoid unnecessary context switches. Otherwise, it enables interrupts on interfaces and enters a sleep state, awaiting an event triggered by a packet arriving on one of the core's RX queues that wakes the core up.

III. Experimental Evaluation

Our evaluation setup consists of the Device Under Test (DUT), which features an Intel® Xeon® Gold 5418Y CPU with 48 cores, a base frequency of 2.20 GHz, and a turbo frequency of 3.80 GHz, along with 256 GB of DDR5 RAM,

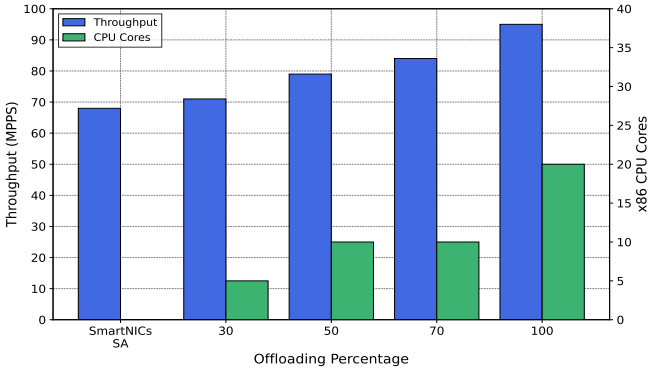

Fig. 2: Throughput and x86 CPU core vs. offloading percentage.

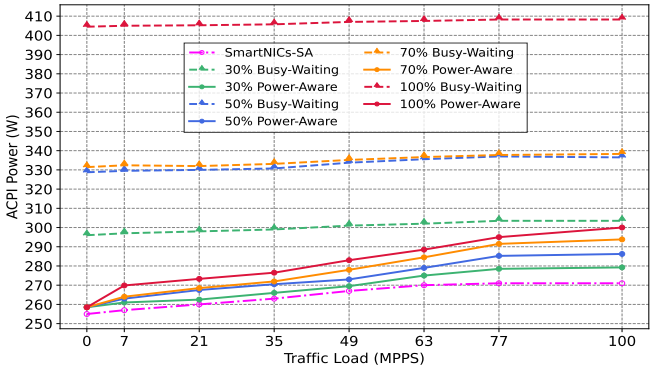

Fig. 3: ACPI power vs. load for diff. power management strategies.

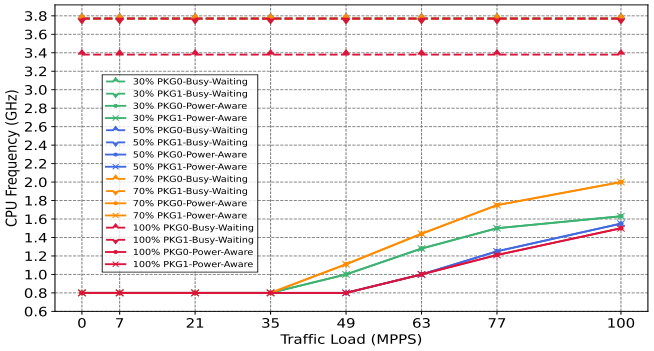

Fig. 4: CPU freq. vs. load for diff power management strategies.

Ubuntu 18.04, and DPDK v20.08. DUT is the node where the gNB network function is executed. It has five Netronome Agilio CX 2x40G SmartNIC, each featuring 60 processing cores called micro-engines, running at 800 MHz. We use a separate x86 server as a traffic generator using Trex v.3.04, which connects directly with fiber to the DUT's SmartNIC. We generate downlink traffic with a packet size of 128 bytes, and varying destination IP addresses to mimic 64 k UEs. Trex measures throughput by sending packets to the DUT and monitoring the rate at which they are processed and returned without any drop.

We measure power consumption using Advanced Configuration and Power Interface (ACPI), which reads power data from a power meter chip embedded in the motherboard and exposes it through a virtual system bus in Linux, allowing us to identify connected components. In our setup, the Netronome SmartNICs and CPUs are connected to this bus, while RAM is not. For RAM power and CPU frequency measurement, we use Intel SoCWatch, capturing average values over 20 s after TRex traffic reaches the target load, with RAM power consistently ranging between 10.5 W and 12.5 W. Summing the power reported by ACPI and RAM, and factoring in the efficiency of the Supermicro PSU PWS-920P-1R2, which has a maximum power output of 920 W and operates at 90% efficiency at 20% load, 92% at 50% load, and 89% at full load aligns closely with the total power draw measured by the PE8324G, an ECO Power Distribution Unit (PDU).

Figure 2 presents the relationship between throughput and CPU core usage for different offloading percentages. This illustrates how CPU demand increases with higher offloading until the SmartNIC becomes a bottleneck in packet injection to the host, limiting further improvements. Figure 3 demonstrates how ACPI power consumption responds to various power management strategies. At 100% offloading, which is the worst-case scenario with higher CPU demands, power readings confirm that adaptive power management limits total power increases to 11% compared to 52% without such management. In Figure 4, CPU frequency behavior is illustrated under different conditions, highlighting that busy-waiting strategies keep the CPU frequency at its peak regardless of traffic load, except under 100% offloading in PKG-0 with three connected SmartNICs, where thermal and load constraints prevent the CPU from reaching its maximum turbo frequency; in contrast, power-aware approaches dynamically adjust frequency based on load.

Acknowledgements

This work was partially funded by the Bavarian State Ministry of Education, Science, and Art through the High-Tech Agenda (HTA) and the Knowledge Foundation of Sweden through the DRIVE project.

References

[1] M. Liu, S. Peter, A. Krishnamurthy, and P. M. Phothilimthana, "E3: Energy-Efficient Microservices on SmartNIC-Accelerated Servers," in *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, Renton, WA, July 2019, pp. 363–378.

[2] F. Biersack, M. Liess, M. Absmann, F. Lotter, T. Wild, and A. Herkersdorf, "ecoNIC: Saving Energy Through SmartNIC-Based Load Balancing of Mixed-Critical Ethernet Traffic," in *2024 27th Euromicro Conference on Digital System Design (DSD)*, August 2024, pp. 185–193.

[3] J. Huang, J. Lou, S. Vanavasam, X. Kong, H. Ji, I. Jeong, D. Zhuo, E. K. Lee, and N. S. Kim, "HAL: Hardware-assisted Load Balancing for Energy-efficient SNIC-Host Cooperative Computing," in *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, 2024, pp. 613–627.

[4] X. Li, W. Cheng, T. Zhang, F. Ren, and B. Yang, "Towards Power Efficient High Performance Packet I/O," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 4, pp. 981–996, 2020.

[5] Intel Corporation, "SK Telecom White Paper," 2024, white paper. [Online]. Available: https://cdrdv2-public.intel.com/772736/SK-Telecom-WP-Final.pdf

[6] M. Memarian, A. Kassler, K.-J. Grinnemo, S. Laki, G. Pongracz, and J. Forsman, "Utilizing Hybrid P4 Solutions to Enhance 5G gNB with Data Plane Programmability," in *2024 15th IFIP Wireless and Mobile Networking Conference (WMNC)*, 2024, pp. 47–54.