A Packet Processing Scheme in 5G MAC Protocol Using DPDK

Jaesu Song, Heesang Chung
Mobile Communication Research Division
Electronics and Telecommunications Research Institute
Dae-Jeon, Korea
{heretic, hschung}@etri.re.kr

***Abstract*— In this paper, we propose a packet processing scheme in the MAC protocol of 5G NR. In general, data plane protocols such as MAC, RLC, and PDCP use DPDK library for high-speed data processing. We implement the MAC layer of 5G NR using DPDK library on high performance server platform and measure the packet processing time of the MAC layer. Specifically, the multiplexing and demultiplexing functions of the MAC protocol are implemented and tested on a server platform. Through these measurement results, we found that what affects the packet processing time is the number of MAC SDUs multiplexed into the MAC TB rather than the size of the MAC PDU. We also compare the packet processing time according to the existing packet processing scheme and the newly proposed method applied to 5G NR MAC. Through this, it was confirmed that the new packet processing method shows better performance when a large number of packets are multiplexed into one MAC PDU.**

***Keywords*— 5G NR, MAC, packet processing, DPDK, 3GPP**

I. Introduction

Looking at the direction of evolution of mobile communication systems such as 4G LTE, 5G NR and beyond 5G, it is characterized by using wider bandwidth to support application services requiring high data rates. For example, in the case of a 4G (LTE) system, the system bandwidth is 20 MHz, but 5G (NR) system supports up to 400 MHz in case of FR2. A characteristic of such ultra-wideband system is that the size of a transport block (TB) to be processed in the MAC layer increases according to the system bandwidth. MAC PDU(Protocol Data Unit) or TB has to be processed within one TTI which is scheduling unit of MAC protocol. So from the transmission point of view of a MAC protocol, it is necessary to construct one TB with a plurality of packets received from the upper layer within one TTI and transfer the TB to a physical layer. From the packet processing point of view, as the size of the TB increases and the number of packets constituting one TB increases, it takes more time to construct the TB. Therefore, since the MAC basically has to perform all functional operations within one TTI, a high-speed packet processing method is required. For this reason, the data plane protocol usually uses the DPDK library[1] for high-speed packet processing.

In general, since packets received from a network interface are firstly processed in the kernel, context switching between kernel space and user space occurs when we need to process the packets in user space. This context switching is one of the factors that make high-speed packet processing difficult. DPDK is a software tool developed by Intel that enables high-speed data processing by enabling users to access and process the packets from network interface bypassing the kernel. By using DPDK, we can perform network packet processing while avoiding context switching and data copy. Figure 1 compares standard packet processing and DPDK packet processing scheme.

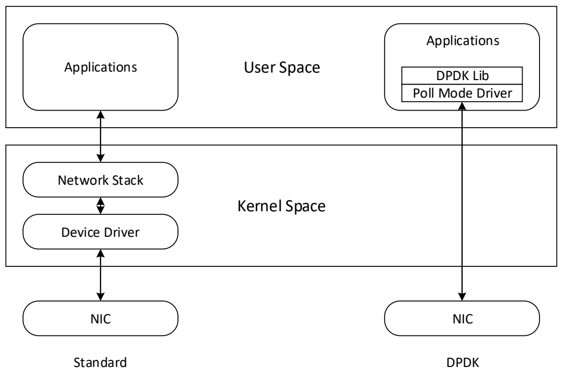


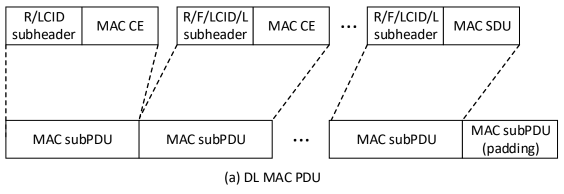
Fig. 1. Standard and DPDK packet processing[2]

In this paper, we propose a new packet processing scheme in the MAC protocol of 5G NR and implement the proposed scheme using the DPDK library on a high performance server platform. To show the superiority of the proposed method, we also measure and analyze the packet processing time of the MAC layer. In particular, the packet processing time is measured according to the TB size and the number of packets constituting the TB.

This paper is organized as follows: Section 2 describes the implementation of MAC protocol using DPDK library and section 3 includes packet processing scheme of DPDK. In section 4, we perform the measurement of packet processing time and analyze the results and conclude this paper in section 5.

II. Implementation of MAC protocol

One example of a MAC PDU of 5G NR MAC is shown in Figure 2. The MAC PDU consists of one or more MAC subPDUs and each MAC subPDU is made of a control signal (MAC CE) generated by the MAC protocol itself or data (MAC SDU) received from an upper layer. At the end of the MAC PDU, an additional padding MAC CE is added to make the MAC PDU match a TB size. In order to perform the multiplexing function of the MAC layer, a MAC subPDU is created by attaching a MAC subheader to a MAC CE or MAC SDU, and sequentially copied to a continuous memory area to generate a MAC PDU. To this end, multiple memory allocations and copying must be accompanied, which



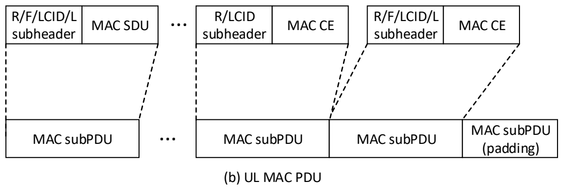


Fig. 2. Example of MAC PDU[3]

degrades packet processing performance. In order to do these tasks at high speed, DPDK provides the following APIs.

- rte_pktmbuf_prepend(): API that appends a MAC subheader to a MAC CE or a MAC SDU
- rte_pktmbuf_chain(): API that links MAC subPDUs into a linked list
- rte_pktmbuf_linearize(): API that combines the linked list of MAC subPDUs into one packet

DPDK manages internally packets in the form of packet buffer as shown in Figure 3. A structure called mbuf is located at the front of the packet buffer, and the actual data follows it with head room and tail room. When the rte_pktmbuf_chain() API is called, the packet buffer structure is changed into a linked list as shown in Figure 4. Each MAC subPDU is stored in the linked list as one data. The mbuf at the beginning of each packet buffer has a pointer pointing to the next data in the linked list. After chaining the MAC subPDUs, in order to combine MAC subPDUs stored in the linked list into one MAC PDU, it is necessary to call the rte_pktmbuf_linearize() API. The result of calling the API is a packet buffer composed of one data as shown in Figure 3, and the length of the data is equal to the sum of all data in the linked list.

When considering the packet processing procedure of DPDK described above, the factors that require processing time to generate the MAC PDU are as follows.

- Attaching MAC subHeader
- Chaining MAC subPDUs into a linked list
- Linearizing(combining) MAC subPDUs into a MAC PDU

There is a head room in front of and a tail room at the end of the actual data in the packet buffer of DPDK as shown in Figure 3. When protocol headers or tails are added in DPDK, the contents are stored in the head room or tail room, so additional memory allocation or memory copying does not occur. Therefore, little processing time is required to perform the subHeader attachment. Secondly, in order to chain a MAC subPDU into a linked list, it is required to search for the mbuf of the last data and link the MAC subPDU as the last data of the linked list. Therefore, when a MAC subPDU is added to a linked list with N data, N search operations are needed. Lastly, memory copying occurs in the processing of combining MAC subPDUs in a linked list into one MAC PDU. At this time, the processing time required to perform memory copy is determined by the number of data in the list and the size of the data.

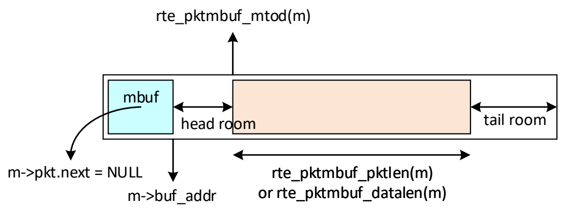


Fig. 3. Packet buffer(one-segment) in DPDK[4]

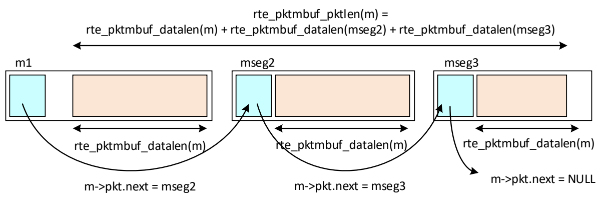


Fig. 4. Packet buffer(multi-segment) in DPDK[4]

III. PACKET PROCESSING SCHEME

We consider two methods for transforming MAC subPDUs that exist in the form of a linked list into one MAC PDU.

A. *Scheme-1(One linearize after all chain)*

In this scheme, all MAC subPDUs are inserted to a linked list by calling a rte_pktmbuf_chain() API, and after that a MAC PDU is created with single rte_pktmbuf_linearize() call. Figure 5 shows this scheme. The MAC subPDU is added to the linked list sequentially and combined into one MAC PDU together at the end. In this method, when a MAC subPDU is attached to a linked list with N data, N search operations are needed. Therefore, if a MAC PDU is to be generated with N MAC subPDUs, the total number of searches required is as follows.

$$\frac{N(N-1)}{2}$$

B. *Scheme-2(Repetition of chain & linearize) – proposed*

One MAC subPDU is added to a linked list by calling a rte_pktmbuf_chain() API and combined into one data through rte_pktmbuf_linearize() API call. This procedure is repeated N-1 times for each MAC subPDU. In this method, since the number of data in the linked list is always 1, the number of searches required when adding MAC subPDU is also 1. Therefore, the total number of searches required when generating a MAC PDU with N MAC subPDUs is N-1 times. In this paper, we propose this method as a new scheme. Figure 6 describes this scheme. As you can see from the figure, rte_pktmbuf_chain() and rte_pktmbuf_linearize() are always called in pairs. The number of data of the linked list does not exceed 2.

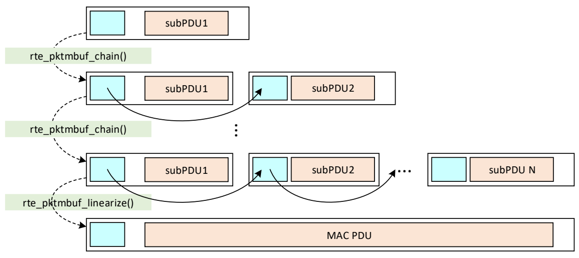


Fig. 5. Scheme-1 (One linearize after all chain)

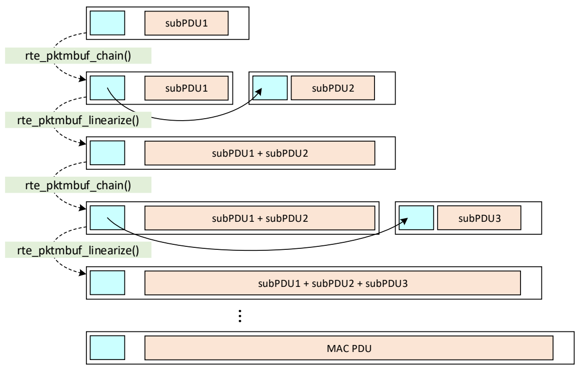


Fig.6. Scheme-2 (Retptition of chain & linearize)

We compare the two methods described above. The number of rte_pktmbuf_chain() API call is the same for both schemes. However, the number of rte_pktmbuf_linearize() API call is 1 and N-1 times in the Scheme-1 and Scheme-2 respectively. However, regardless of the number of the API calls, the number of memory copy is the same. So the complexity of combining MAC subPDUs into a MAC PDU is not different. However, the number of searches of the linked list is different between two schemes. The Scheme-1 is N(N-1)/2 and the Scheme-2 is N-1.

IV. Performance Evaluation

5G NR MAC protocol is implemented on high performance server platform. The specifications of the platform are shown in Table 1. We measured the packet processing time to generate the MAC PDU using the implemented MAC protocol. The environment of performance test is shown in Figure 7. The MAC protocol on the test platform receives IP packets from the traffic generator, creates MAC PDUs, and then transmits them to the PHY emulator whthin TTI.

The packet processing time according to the two methods described above is measured and compared. In Figure 8, the packet processing time is measured while varying the number of MAC subPDUs constituting one MAC PDU. At this time, the size of the MAC PDU is fixed at 5000 bytes. As shown in Figure 8, the two schemes show similar performance up to 50 MAC subPDUs. However, when the number of MAC subPDUs is more than 50, the packet processing time of Scheme-2 is measured to be lower than Scheme-1. As mentioned in the previous section, this performance gap can be understood as being caused by the difference in the number of linked list searches between the two schemes. When the number of MAC subPDUs is 100, the performance difference between the two methods is about 37%.

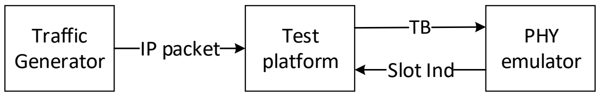


Fig. 7. Performance test environment

TABLE I. TEST ENVIRONMENT

Item	Description
Server platform	Supermicro SuperServer 5019D-FN8TP
CPU	Intel® Xeon® Processor D-2183IT @ 2.2GHz 16 cores
Memory	DDR4, 64GB: 16GB x 4 DIMMs
NIC	Dual LAN with 10Gbase-T
DPDK version	18.05
OS	Ubuntu 18.04.1 LTS
gcc version	7.5.0
kernel version	4.15.0-60-generic

In Figure 9, the packet processing time is measured while fixing the number of MAC subPDUs constituting one MAC PDU and changing the size of the MAC PDU. Here, the number of MAC subPDUs constituting the MAC PDU is fixed at 20. As shown in Figure 9, there is little performance difference between the two methods. Comparing Figure 8 and Figure 9, we can see that the packet processing time is greatly affected by the number of MAC subPDUs consisting of one MAC PDU but not by the size of the MAC PDU.

V. Conclusion

A packet processing method using DPDK in MAC protocol is proposed. We showed that the proposed method (Scheme-2) is superior to the existing scheme (Scheme-1) by measuring the packet processing time on high performance server platform. It is also shown that the packet processing time is determined by the number of MAC subPDUs multiplexed into one MAC PDU rather than the size of the MAC PDU. It is expected that future mobile communication systems will evolve into ultra-broadband system. This means that the size of a MAC PDU increases in a mobile communication system, and accordingly, the number of MAC subPDUs constituting one MAC PDU increases. Therefore, the method proposed in this paper will need to be applied to future ultra broadband mobile communication systems.

Acknowledgment

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2023-00216221, Development of service coverage extension technologies for 5G-Advanced mobile communications based on reconfigurable intelligent surface).

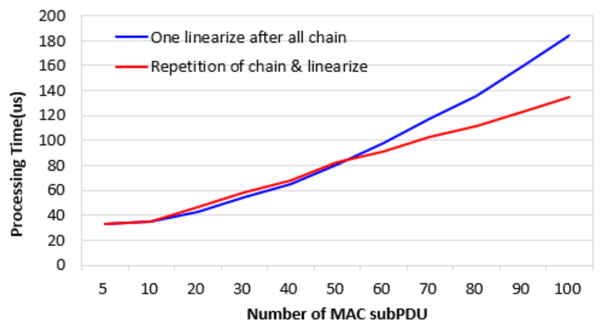


Fig. 8. MAC processing time according to Number of MAC subPDU

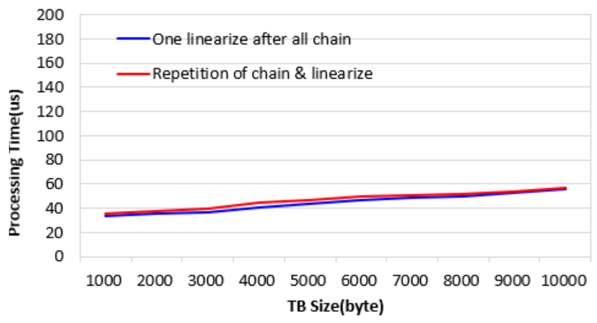


Fig. 9. MAC processing time according to the TB size

References

[1] https://www.dpdk.org/

[2] https://hackmd.io/@sohailanjum97/SkWE46ywu

[3] 3GPP TS 38.321, "NR; Medium Access Control (MAC) protocol specification", Iune, 2022.

[4] DPDK, Programmer;s Guide, Jan, 2021.