

A Resource Saving MAC/IP Handling Method for Low-Power Wireless TCP Packet Network

Jong Beom Kim*, Jae Woo Ahn*, Jong Oh Kim**, Seong Gon Choi*

**Information & Communication Engineering, Chungbuk University, Cheongju-si Chungcheongbuk-do, Korea*

***Fisys, 168, Gajeong-ro, Yuseong-gu, Daejeon, Korea*

dragonslash@cbnu.ac.kr, jwahni@cbnu.ac.kr, jokim@fisys.co.kr, choisg@cbnu.ac.kr

Abstract— This paper presents the technology and implementation results of manipulating the MAC and IP Address of the transmitting side in MCU(Microcontroller Unit), when data communication with remote APs is performed on CC3220 Wi-Fi module associated with STM32 MCU. Because the existing packet sending and receiving process also changes in the CRC as MAC and IP change, CRC checksum errors occur when MAC and IP are manipulated. The method presented in this paper reduces the resources required for computations to match the CRC checksum in the transmission and reception environment between MCU and external AP, and it also allows the sending-receiving and exchanging data with minimal manipulation.

Keywords— IP Handling, MAC, MCU, Checksum, Protocol

I. INTRODUCTION

One of the most active areas in network-related technologies is IoT (Internet of Things) technology that connects to the Internet including communication functions in various network objects and sensors. As demand for IoT technology increases and R&D is activated, the market size is continuously expanding, and the quantity of IoT equipment is also increasing. There are about 7 billion IoT devices in the market as of 2018, and it is expected that 21 billion IoT devices will be in the market by 2025^[1].

The initial IoT environment was shaped like a basic sensor network in which measuring device, such as sensors, delivered information to the server that managed the data, and was connected to the server through a wired and wireless short-range network. Currently, however, support for wired and wireless data communication technologies is required as various types of sensors are connected to medium to long range wireless networks, and IoT network topologies are also designed based on medium to long range wireless networks.

In this paper, we propose a method to improve the part that required inefficient operation in existing communication based on the environment where two devices communicate through MCU and Wi-Fi module. During the communication between the MCU and the remote AP via the Wi-Fi module, the MAC address and the IP address change when one hop occurs. If the MAC address and the IP address are changed in the process of transmitting the data packet, an operation for the MAC checksum and the IP checksum is additionally

performed. If the checksum operation is not performed, an error occurs in the CRC check process and the packet is not normally transmitted. Therefore, in the existing communication method, it is necessary to perform checksum calculation on MAC and IP.

However, in a sensor network where a network must be maintained with minimal power consumption, additional resource consumption is directly associated with additional power consumption. Further, in a closed-form sensor network in which there is no risk of packet modulation from the outside, additional power is consumed because resources required for the checksum calculation is occupied.

To solve this problem, this paper proposes a method to improve the efficiency of the whole network by eliminating the need for checksum calculation by manipulating the MAC and IP existing in the packet header and reducing unnecessary additional resource consumption.

We apply the method presented in the paper to the actual STM32 MCU and CC3220 Wi-Fi module environment and test the effectiveness of the proposed technology by experimenting with sending and receiving network packets. After the sending side sends the packet containing the specific data, the receiving side confirms whether the data is normally transmitted through the terminal program and packet analyser program.

In chapter 2, we describe the basic structure of the TCP packet and the process of converting the MAC and IP addresses in the hop generation that is the basis of the method presented in this paper. Chapter 3 describes the description of the technique. Chapter 4 describes the experimental environment and the implementation results to confirm the operation of the technology presented in this paper. Chapter 5 presents conclusions.

II. RELATED WORKS

In the communication process of the MCU, information transmission is performed through packet communication like other communication structures. Therefore, the existing communication structure is similar to the utilization or conversion method of MAC address or IP address. In this clause, we describe the process of converting the MAC address or IP address as hop moves, and the header structure of TCP packet which included in network packet.

A. MAC/IP Usage in TCP Packet Networking

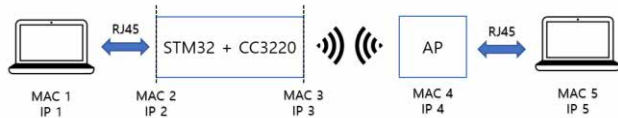


Figure 1. MAC/IP Conversion in Typical TCP Packet Network

In the TCP packet communication structure, each device performs communication with a unique identification scheme. There are many kinds of identification schemes that appear in the form of addresses. There are IP addresses in Internet Protocol, and MAC (Media Access Control) addresses that are unique to NIC(Network Interface Card)^[2].

In a typical network environment, as packets are delivered, the sender information contained in the data packet changes every time it goes through each device. With Figure 1, when a device with MAC 1 and IP 1 transmits a data packet to a device with MAC 5 and IP 5, the transmission is changed to the MAC and IP of new sender device whenever it goes through devices such as an AP.

When there are changes in MAC address and IP address, checksum values for changed MAC address and IP address are newly calculated in a typical communication environment. This checksum value is used to provide the integrity of the transmitted data, and if the checksum values do not match, it can be determined that the data is corrupted or modified, called CRC errors^[3].

B. TCP Header Structure

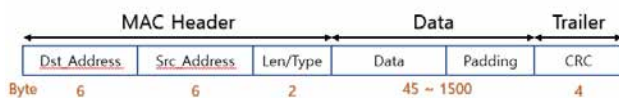


Figure 2. TCP Header Structure

The TCP network packet carries data including the header packet structure as shown in Figure 2. The TCP header has a total of 14 bytes of MAC header, including 6 bytes of Destination Address(DA), 6 bytes of Source Address(SA), and 2 bytes of Length. After the MAC header, it has data of up to 1,500 bytes and finally has 4 bytes of CRC data^{[4]-[5]}.

Cyclic Redundancy Check(CRC) means a result calculated through a specific method to check whether data transmitted over a network has an error. The CRC value is calculated according to the IP address and the MAC address when the data is transmitted, and sent as attached to the header. When the receiver receives the data, the CRC value is calculated through the information of the received packet^[3].

III. MAC/IP HANDLING METHOD

In this clause, we describe how to save resources required for checksum calculation through handling MAC address and IP address proposed in this paper.

In the typical TCP packet networking process, the MAC address and the IP address of the source are converted into the

MAC address and the IP address of the new source for each hop, and then the checksum value used for the integrity check is newly calculated and attached to the header. The method of providing integrity through the checksum value is necessary in order to prevent the transmission error or modification of the packet in the general network environment. However, in the IoT environment having the closed network form not connected to the external network, there is no need to consider the external modification, further calculation of the new value causes waste of resources.

Therefore, in this paper, we propose a method to maintain the MAC address and IP address of the original packet without converting the MAC address or IP address for each hop in order to perform data transmission without performing additional checksum calculation. This method eliminates the need for checksum calculation and saves resources required for computing.

A. MAC/IP Handling in MCU

In the general TCP packet communication structure, the MAC header and the IP address of the transmitting device are included in the TCP header, and the data packet and the CRC are transmitted together.

The key point of this paper is to maintain the existing MAC address of the first transmission without converting it every time for each hop. To do so, the MAC address and IP address of the packet must be obtained and stored in the MCU.

Figure 3 and Figure 4 show the structure of the code that obtains the MAC address and IP address in the packet and stores in the buffer in the MCU. Referring to the TCP header structure shown in Figure 2, it can be confirmed that the MAC header part is up to the first 14 bytes of the packet. Therefore, the buffer array in Figure 3 stores the packet data from address number 0 to 13.

```
if(Eth_tx_Flags == 1) {
    memcpy(buffer, Ethrxbuf, spicount+14);
    //dprintf3("Ethernet TX\r\n");
    buffer[0] = CamMac[0];
    buffer[1] = CamMac[1];
    buffer[2] = CamMac[2];
    buffer[3] = CamMac[3];
    buffer[4] = CamMac[4];
    buffer[5] = CamMac[5];
    buffer[6] = 0;
    buffer[7] = 0x80;
    buffer[8] = 0xe1;
    buffer[9] = 0;
    buffer[10] = 0;
    buffer[11] = 0;
    buffer[12] = 0x08;
    buffer[13] = 0;

    HAL_ETH_TransmitFrame(&heth,
        spicount+14);
}
```

Figure 3. Ethernet Transmit Code

```

if (buffer[14] == 0x45 && buffer[15] == 0x00) {
    //dprintf3("Ethernet RX\r\n");
    if (rear == front+1) || (rear == 0 && front ==
39)) {
        dprintf3("eth drop\r\n");
    } else {
        memcpy(bbbuffer[front],
buffer+sizeof(Macbuf), 1500);

        front++;
        if(front > 39) front = 0;
    }
}

```

Figure 4. Ethernet Receive Code

The header information stored in the buffer includes the initial MAC address and the IP address information of the data packet to be transmitted. By storing and maintaining these two addresses in the buffer, it is possible to skip the checksum calculation while keeping the initial MAC header information the same even when transmitting information to the next hop.

B. MAC/IP Address Handling in Wi-Fi Module

In the Wi-Fi module connected to the MCU, a TCP/IP socket is opened and communication is performed such as general packet network communication to perform packet communication with a remote AP.

```

//socket open
SerialWifi_TCP.sockTcpServer =
sl_Socket(SL_AF_INET, SL_SOCKET_RAW,
SL_IPPROTO_TCP);
UART_PRINT("ReceiveTCP socket : %d\r\n",
SerialWifi_TCP.sockTcpServer);

//include ip header option
retVal =
sl_SetSockOpt(SerialWifi_TCP.sockTcpServer,
SL_IPPROTO_IP, SL_IP_HDRINCL,
&IncludeIpHeader, sizeof(IncludeIpHeader));
if( retVal < 0 )
{
    UART_PRINT("Failed to set TCP socket
options 1\r\n");
    sl_Close(SerialWifi_TCP.sockTcpServer);
    return;
}

```

Figure 5. TCP Socket Receive Code

```

//socket open
SerialWifi_RAW.sockTcpServer =
sl_Socket(SL_AF_INET, SL_SOCKET_RAW, 0);
UART_PRINT("SendRAW socket : %d\r\n",
SerialWifi_RAW.sockTcpServer);

//include ip header option
retVal =
sl_SetSockOpt(SerialWifi_RAW.sockTcpServer,
SL_IPPROTO_IP, SL_IP_HDRINCL,
&IncludeIpHeader, sizeof(IncludeIpHeader));
if( retVal < 0 )
{
    UART_PRINT("Failed to set RAW socket
options 1\r\n");
    sl_Close(SerialWifi_RAW.sockTcpServer);
    return;
}

```

Figure 6. TCP Socket Send Code

Figure 5 and Figure 6 show the structure of the code that opens TCP/IP socket in Wi-Fi module and sends and receives packets through socket. Figure 5 and Figure 6 show that the process of including IP header is included through the `sl_SetSockOpt()` function while opening the socket and sending and receiving. By using `sl_SetSockOpt()` function, the MAC address and IP address or the packet are set to the value stored in the buffer.

The changed address is overwritten by the source address and the destination address of the same IPv4 header as the structure of Figure 7. Since the source address and the destination address do not change, the checksum calculation by the changed address is unnecessary. As a result, data communication can be performed while saving resources required for the operation.

IV. IMPLEMENTATION RESULT

In this clause, we apply the proposed technique to the MCU and Wi-Fi environment which are used for actual data communication and show that the proposed technology handles MAC address and IP address normally.

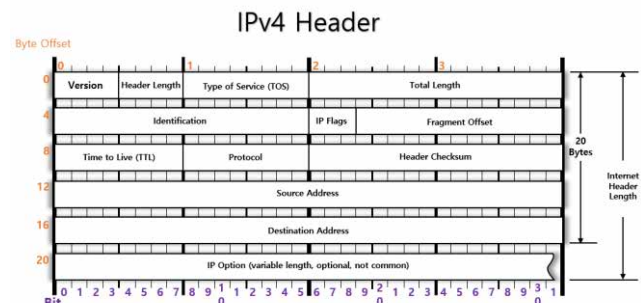


Figure 7. Figure of IPv4 Header

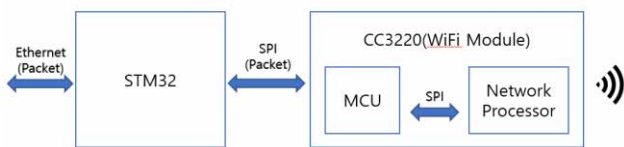


Figure 8. Implementation Environment using STM32 and CC3220

As shown in Figure 8, the experimental environment is configured to attach a Wi-Fi module to the MCU connected to the transmitting device, perform wireless communication, and connect to the remote AP to transmit data to the receiving device. The STM32 32-bit ARM Cortex MCU was used as the MCU, and the CC3220 SimpleLink Wi-Fi module was used as the Wi-Fi module. The STM32, used as an MCU, is one of STMicroelectronics' microprocessor series and uses an 32-bit ARM processor. The CC3220, used as an Wi-Fi module, is a Texas Instruments Wi-Fi module that provides low-power Wi-Fi connectivity.

The implementation follows the network structure of Figure 8 and performs TCP/IP packet networking through MCU and Wi-Fi module. The code composed to perform TCP/IP packet communication was created using μ Vision IDE and Code Composer Studio. The network packet was verified using packet analysis software such as Wireshark and terminal software such as Tera Term VT.

We applied the packet transmission in the Ethernet environment, the TCP socket creation in the Wi-Fi module, and the packet transmission code using the socket, proposed in Chapter 3, to the MCU and Wi-Fi module environment. Implementation were conducted to create test packets and transmit them to verify that the transmission was correct.

The following two functions are used to perform TCP/IP socket communication in MCU and Wi-Fi environment.

```
sl_Recv(SerialWifi_TCP.socket, WiFirecv, 1500, 0);
sl_Send(SerialWifi_RAW.sockTcpServer, WiFirend,
packet_size, 0);
```

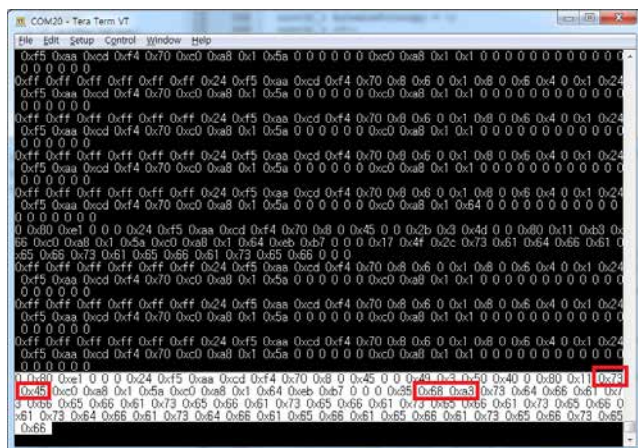


Figure 9. Packet Data shown in Tera Term VT

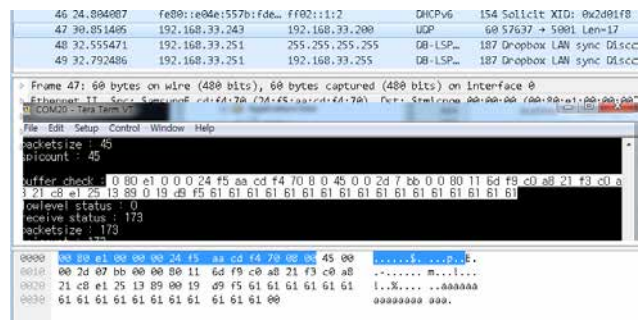


Figure 10. Header Data Stored in Buffer

Two functions, **sl_Recv()** and **sl_Send()**, are functions that receive and send sockets in TCP/IP socket communication, respectively. It is a function that is used to transmit the packet data composed of the contents proposed in Chapter 3 through the Wi-Fi module in the communication with the remote AP connected to the device set as the destination of the test packet.

Figure 9 shows the output of test packet data using Tera Term VT. The highlighted part is the address part modified through the method proposed in this paper.

Figure 10 compares the transmitted test packet data with the packet information stored in the buffer. In the Ethernet section, there is a code to store the packet information in the buffer, and the stored buffer is output so that it can be confirmed through the terminal program. The transmitted packet information was verified by the packet analysis software Wireshark. When the transmission result is checked, it is confirmed that the contents stored in the buffer and the transmitted packet data coincide with each other.

Figure 11 confirms that the transmitted and received packets correctly delivered all data with no errors. As in Figure 10, we looked at the packets sent and received by Wireshark and confirmed that the strings in the test packets were successfully transmitted without missing.

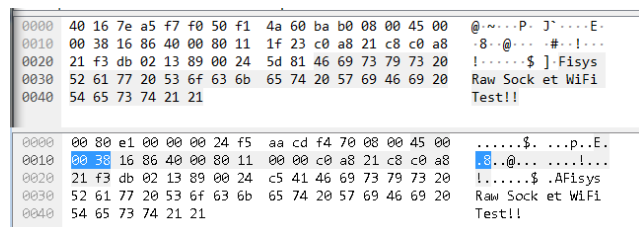


Figure 11. Transmitted/Received Packet Inspection by Wireshark

V. CONCLUSIONS

In this paper, we propose a method to save the resources required for checksum calculation during wireless socket networking to remote AP by handling MAC address and IP address existing in packet header. In the existing TCP/IP packet networking, the MAC address and the IP address are changed every hop, and additional resources are consumed to newly calculate the checksum. However, the MCU handles the MAC address and the IP address of the packet, thus reducing additional resource consumption.

By using the method presented in this paper, we confirmed that the packet is normally transmitted without any additional checksum calculation in the experimental environment of real MCU and Wi-Fi module.

In addition, the proposed method can be applied not only to MCU and remote AP but also to many remote sensors, so it can be widely applied to IoT environment where low-power consumption is essential.

ACKNOWLEDGMENT

This research was supported by the KIAT(Korea Institute for Advancement of Technology) grant funded by the Korea Government(MSS : Ministry of SMEs and Startups). (No. N0001132, HRD program for 2018)

* Corresponding Author : S.G. Choi(choisg@cbnu.ac.kr)

REFERENCES

- [1] (2018) IOT ANALYTICS website. [Online]. Available: <https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/>
- [2] (2004) Laneye website. [Online], Available: <http://www.laneye.com/network/how-network-works/mac-address-and-ip-address-relationship.htm>
- [3] Howstuffworks website. [Online], Available: <https://coMCUter.howstuffworks.com/encryption.htm>
- [4] TCP Header, Request for Comments 793, Sep 1981
- [5] IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture, IEEE Std 802-2001, 8 March 2002
- [6] Life.augmented website. [Online]. Available: <https://www.st.com/ko/stm32/stm32/stm32intro.html>
- [7] TEXAS INSTRUMENTS website. [Online]. Available: <http://www.ti.com/product/CC3200MOD>



Jong Beom Kim received B.S. degree in the College of Electrical & CoMCUter Engineering, Chungbuk National University, Korea in 2017. He is currently a M.S. candidate in School of Electrical & CoMCUter Engineering, Chungbuk National University. His research interest is network programming.



Jae Woo Ahn received B.S. degree in the College of Electrical & CoMCUter Engineering, Chungbuk National University, Korea in 2017. He is currently a M.S. candidate in School of Electrical & CoMCUter Engineering, Chungbuk National University. His research interest is network programming.



Jong Oh Kim received B.S. and M.S. degree in Electronics Engineering from Kyeongbuk National University in 1990 and 1992 respectively. He is currently CEO in Fisys Inc. His research interests are SDN, NFV.



Seong Gon Choi received B.S. degree in Electronics Engineering from Kyeongbuk National University in 1990, and M.S. and Ph.D. degree from KAIST, Korea in 1999 and 2004, respectively. He is currently a professor in the College of Electrical & CoMCUter Engineering, Chungbuk National University. His research interests include smart grid, IoT, mobile communication, high-speed network architecture and protocol