Transmission Sequence Adjustment Scheme for Improving Fairness in Wireless LANs

Young Ung Kim and Sunmyeng Kim

Abstract—In a wireless LAN, multimedia traffic is one of important traffic types. However, it is not easy to transmit multimedia traffic in a wireless LAN because it requires strict assurance for delay and loss. Although a multimedia packet is sent to a receiver without an error, it is discarded, if it does not satisfy its own delay bound. In a wireless LAN, a packet is transmitted with a FIFO (First-In First-Out) queue, i.e., a packet is transmitted in sequence regardless of delay bound of each packet. Therefore, traffic with long delay bound has good performance, though traffic with short delay bound shows poor performance. Accordingly, fairness problem between traffic with different delay bounds occur. To overcome the problem, we propose a transmission sequence adjustment scheme. The proposed scheme adjusts the transmission sequence of data packets in the queue by considering the delay bounds of all the data packets in the queue. Performance of the proposed scheme is investigated by simulation. Our results show that the proposed scheme is more effective and alleviates the fairness problem.

Index Terms—Delay bound, EDCA, fairness, transmission sequence.

I. INTRODUCTION

The IEEE 802.11 wireless LAN is widely used for wireless access due to its easy deployment and low cost. The IEEE 802.11 standard defines a medium access control (MAC) protocol for sharing the channel among stations [1]. The distributed coordination function (DCF) was designed for a contention-based channel access. The DCF has two data transmission methods: the default basic access and optional request-to-send/clear-to-send (RTS/CTS) access. The basic access method uses the two-way handshaking (DATA-ACK) mechanism. The RTS/CTS access method uses the four-way handshaking (RTS-CTS-DATA-ACK) mechanism to reserve the channel before transmitting long data packets.

The widespread use of multimedia applications requires new features, such as high bandwidth and small average delay and delay jitter in wireless LANs. Unfortunately, the IEEE 802.11 MAC protocol cannot support quality of service (QoS) requirements [2], [3]. In order to support multimedia applications with tight QoS requirements in the IEEE 802.11 MAC protocol, the IEEE 802.11e has been standardized [4].

It introduces a contention-based new channel access mechanism called enhanced distributed channel access (EDCA). The EDCA supports the QoS by introducing four access categories (ACs), AC 3, AC 2, AC 1, and AC 0 are for voice, video, best-effort data, and background traffic, respectively. To differentiate the ACs, the EDCA uses a set of AC specific parameters, which include minimum contention window (CWmin[i]), maximum contention window (CWmax[i]), arbitration inter-frame space (AIFS[i]), and transmission opportunity (TXOP[i]) for AC i (i = 0, . . . , 3). The AIFS is, at least, distributed inter-frame space (DIFS) long and is calculated with the AIFS number (AIFSN[i]). The duration of AIFS[i] is defined by AIFS[i] = SIFS + AIFSN[i]*aSlotTime, where SIFS is a short inter-frame space, and aSlotTime is the duration of a slot time. For 0 ≤ i < j ≤ 3, the EDCA has CWmin[i] ≥ CWmin[j], CWmax[i] ≥ CWmax[j], and AIFSN[i] ≥ AIFSN[j]. Note that, in the preceding inequalities, at least one must be “not equal to.” Once a TXOP is obtained using a backoff, a station is allowed to transmit more than one data packets consecutively during the TXOP. The EDCA assigns a smaller CW and shorter AIFS to higher priority classes in order to ensure that in most cases, higher priority classes will experience lower mean waiting and backoff times than lower priority ones. Therefore, in the EDCA, support of QoS can be achieved statistically by reducing the probability of medium access for lower priority classes.

There still remain problems in the EDCA since the scheduling of packet transmissions is basically first in first out (FIFO)-based mechanism. Real-time multimedia traffic is very sensitive to delay so that the traffic has to be transmitted to receivers within its delay bound. Otherwise, it is discarded at the receivers, even though the receivers receive the data packets without any error [5], [6]. Therefore, guaranteeing the delay bound is critical for real-time multimedia traffic. In the environment where multimedia data packets in the queue of a station have different delay bounds, fairness problem arises due to the FIFO queue scheduling. This is explained in detail in Section II.

In order to alleviate the fairness problem, we propose a Delay Bound based Transmission Sequence Adjustment (DBTSA) scheme. The DBTSA scheme rearranges the transmission order of data packets in the queue based on their delay bound. In the proposed scheme, the packets with long delay bound yield to the packets with short delay bound. That is, a packet with long delay bound is moved to backward in the queue as far as its delay bound allows, and a packet with short delay bound moves forward. Likewise, data packets with short delay bound are transmitted first to increase the probability of success. In this way, fairness in multimedia
traffic can be enhanced.

In order to support real-time multimedia traffic, many schemes have been proposed in the literature. However, the previous schemes still have several problems. First, some of them need modifications to the IEEE 802.11e standard [7]-[9]. Therefore, they are not backward compatible with the legacy EDCA. For example, Deng et al. proposed a surplus TXOP diverter (STXD) scheme to define the TXOP limit for per-flow but not for per-ACs [7]. However, the standard is on a per-AC basis. Second, some use analytical models to calculate the QoS metrics which are usually derived based on a few unpractical hypotheses. They do not reflect the characteristics of real traffic. Therefore, they are always inaccurate and clearly not applicable to realistic environments [10], [11]. Finally, others require feedback information from stations to consider the dynamic behavior of multimedia flows, but the feedback cannot provide an appropriate indication to the current network load conditions in a real-time manner [12], [13].

The paper is organized as follows. We illustrate the fairness problem in Section II. In Section III, the DBTSA scheme is presented in detail. In Section IV, performance studies are carried out through simulation results. Finally, we draw conclusions in Section V.

II. FAIRNESS PROBLEM

The delay bound normally refers to the end-to-end delay bound. And the residual delay bound refers to the residual time interval of the delay bound at a station. Hereafter, we will use the term delay bound to refer to the residual delay bound. It is the difference between the end-to-end delay bound set by a real-time multimedia server for each data packet and the total accumulated delay experienced by the packet while it traverses through the network [14].

Data packets at a station may have different delay bound for three reasons. First, the end-to-end delay bound of a data packet depends on the multimedia traffic characteristics. And the traffic has widely varying characteristics. Therefore, data packets may have different end-to-end delay bound. Second, each data packet experiences different time delays over various paths. Third, delay that data packets experience depends on the distance between multimedia servers and intermediate stations. If they are close, data packets sent by a multimedia server experience short delays, or otherwise they experience long delays.

In this environment, if an AP (access point) transmits data packets to receivers without additional mechanisms, the fairness problem between data packets with short and long delay bounds arises. A packet with short delay bound has high possibility that it is discarded since it does not meet its own delay bound. However, a packet with long delay bound has high possibility that it is successfully transmitted within its own delay bound. Another drawback of different delay bounds is that if a station uses much channel time to transmit the head of line packet with long delay bound, the other packets in the queue are delayed due to the FIFO scheduling. This makes the possibility of the other packets breaking their delay bounds. Therefore, data flows with long delay bound have higher performance than data flows with short one. This results in the fairness problem.

![Fig. 1. Example of the fairness problem in the EDCA.](image)

Fig. 1 shows the example of the fairness problem in the EDCA scheme. We simulated by using the simulation parameters in Tables I and Table II of Section IV. In the simulation, there are two multimedia flows. The short and long flows mean flows with short and long delay bounds, respectively. From the figure, we see that the long delay bound flow always has much better performance than the short one regardless of the variation of the number of background traffic stations.

III. DBTSA SCHEME

In the EDCA, the scheduling of packet transmissions is basically first in first out (FIFO)-based mechanism. That is, EDCA sends data packets in the order they arrive without considering the delay bound of each packet in the queue. Accordingly, even if the multimedia traffic is transmitted to a receiver successfully without an error, it is discarded because it is not met its delay bound. As a result, it causes a fairness problem of multimedia traffic according to the delay bound. To alleviate this problem, we propose a new scheme of adjusting the transmission sequence of packets in the queue according to their delay bounds. In the proposed scheme, the packets with long delay bound yield to the packets with short delay bound. That is, a packet with long delay bound is moved to backward in the queue as far as its delay bound allows, and a packet with short delay bound moves forward. Likewise, data packets with short delay bound are transmitted first to increase the probability of success. In this way, fairness in multimedia traffic can be enhanced.

![Fig. 2. Example of adjusting the packet transmission sequence.](image)
Fig. 2 shows an example of how to alleviate the fairness problem and improve network performance by adjusting transmission sequence of each packet in the queue based on its delay bound. We assume that there are three data packets in the queue, and they have delay bounds of 25, 6, and 13 respectively. Transmission time of a packet is 5. In Fig. 2(a), PKT 1 is the head packet in the queue, so that it is first transmitted to a receiver. Because PKT 1 has a larger delay bound than the packet transmission time, it is sent to a receiver successfully. After transmitting PKT 1, delay bounds for PKTs 2 and 3 are left 1 and 8, respectively. Second, PKT 2 is transmitted. However, since delay bound of this packet is smaller than the packet transmission time, it is discarded at the receiver. And additional 5 is used to transmit PKT 2, remained delay bound for PKT 3 is 3. Also, delay bound of PKT 3 is also smaller than the packet transmission time, and it is discarded at the receiver. Only PKT 1 with long delay bound is sent successfully, while PKTs 2 and 3 with small delay bound are failed. As a result, a fairness problem occurs.

Fig. 2(b) shows the queue after adjusting the packet transmission sequence. PKT 2 with the smallest delay bound moves to the head of the queue, and PKT 1 with the largest delay bound to the tail. Since PKT 2 is located at the queue head, it is the first to send. PKT 2 arrives at the receiver within its delay bound, and it is not discarded. After transmitting PKT 2, delay bounds of PKTs 3 and 1 are left 8 and 20, respectively. PKT 3 is the second to send. Because delay bound of this packet is also larger than the packet transmission time, it is transmitted successfully. Now, delay bound of PKT 1 is 15. PKT 1 also has sufficient delay bound, and it is transmitted to the receiver successfully. Unlike the result in Fig. 2(a), three packets in Fig. 2(b) are sent to the receiver within their delay bound, respectively.

Before describing the DBTSA scheme in detail, we introduce a new terminology definition. Let us denote a successful transmission interval as the time interval between two consecutive successful transmissions (see Fig. 3), which is made up of two components: idle and busy periods. An idle period extends over consecutive empty slots and a busy period corresponds to a successful transmission by other stations or collided transmission, including AIFS or EIFS closing the transmission, respectively.

The successful transmission interval (STI) is updated after a station successfully transmits its own data packet. A station measures the instantaneous successful transmission interval (STIcurrent) of the nth successful transmission. From this value, STIn is approximated by exploiting the following moving averaging window:

\[ STI_n = \alpha \cdot STI_{n-1} + (1 - \alpha) \cdot STI_{current} \]  

where, \( STI_{n-1} \) is the successful transmission interval at the end of the \((n-1)\)th successful transmission, and \( \alpha \) is a smoothing factor in the range of [0, 1].

After obtaining the successful transmission interval, the proposed scheme converts the delay bound of a packet into TDB (Transmission Delay Bound). TDB is the delay bound represented in terms of the number of transmissions. It is as follows:

\[ TDB = \left\lfloor \frac{DB_i}{STI} \right\rfloor \]  

where, \( DB_i \) is the delay bound of packet \( i \), \( \lfloor x \rfloor \) rounds to the largest integer smaller than or equal to \( x \).

After calculating TDB, it is possible to determine whether or not a packet is transmitted successfully within its own delay bound as follows:

\[ idx_i \leq TDB_i \]  

where, \( idx_i \) (\( idx_i = 1, 2, 3, \ldots \)) and \( TDB_i \) are a queue index and TDB value of packet \( i \), respectively. If the above condition is met, then it determines that successful transmission is possible within its delay bound. If not, it determines that the transmission is failed. That is, if TDB value of each packet is larger than or equal to its queue index, then successful transmission is possible.

Fig. 4 shows an example of calculating TDB and whether packets are transmitted successfully. In the figure, the AP has nine data packets with the delay bounds of (27, 11, 18, 29, 21, 13, 7, 32, and 52), respectively. STI is assumed to be 5. Using (2), for each data packet, we obtain the TDB values of (5, 2, 3, 4, 2, 1, 6, and 10). When determining whether each packet is transmitted successfully by using (3), 5 packets (PKTs 1, 2, 3, 4, and 9) are sent successfully within their delay bounds, though remaining 4 packets (PKTs 5, 6, 7, and 8) fail since they cannot meet their delay bounds.

When adjusting the data packet transmission sequence according to the obtained TDB, the following two basic guidelines shall be observed.

1) A packet which can meet its own delay bound before adjusting the transmission sequence shall satisfy the
delay bound even after adjusting the sequence.

2) The number of packets met their delay bounds after adjusting their transmission sequence shall be equal to or larger than that before the adjustment. That is, it is necessary to enhance performance.

The proposed scheme is divided into three steps: position adjustment of successful packets, position adjustment of failed packets, and continuous packet arrangement.

---

### A. Step 1: Position Adjustment of Successful Packets

By using (3), we determine whether or not a packet is transmitted successfully. Then we adjust the queue positions of packets determined as capable of sending successfully within their delay bounds. To allow failed packets move forward in the queue, we move the successfully transmitted packets to the back. To keep Basic Guideline 1, a successfully transmitted packet moves backward by its TDB value. Position of packets in the queue is adjusted according to their TDB value. If two or more packets determined as capable of sending successfully within their delay bounds, we first move failed packets with larger TDB values forward in the queue, then every packet with TDB value less than that of the successfully transmitted packet in the queue. In Fig. 4, successfully transmitted packets are PKTs 1, 2, 3, 4, and 9. First of all, among the successfully transmitted packets, PKT 1 with the lowest queue index moves first. TDB value of PKT 1 is 5 and there is no another packet at queue index 5. Therefore, PKT 1 moves to queue index 5 (Fig. 5(a)). Second, we adjust the position of PKT 2. TDB value of PKT 2 is 2 and there is no another packet at queue index 2, it moves to the position (Fig. 5(b)). Third, TDB value of PKT 3 is 3, and its position in the queue is still 3 (Fig. 5(c)). Fourth, since TDB value of PKT 4 is 5, it moves to queue index 5. However, PKT 1 is already positioned at queue index 5. So, PKT 1 moves forward to queue index 4. Then, PKT 4 moves to queue index 5 (Fig. 5(d)). Finally, PKT 9 is moving. Although TDB value of this packet is 10, the length of the queue is 9, so it stays at queue index 9 (Fig. 5(e)).

---

### B. Step 2: Position Adjustment of Failed Packets

We adjust the positions of packets determined as impossible to transmit successfully within their delay bounds by using (3). A failed packet is moved to a position in the queue except that occupied by a successfully transmitted packet in Step 1. Unlike moving of successfully transmitted packets, we first move the failed packets with larger TDB values (i.e., descending order of TDB value) to increase the number of failed packets to be sent successfully. If there are many failed packets with the same TDB value, we move the packet with the lowest queue index first.

Before moving a failed packet, it is required to determine whether the failed packet can be moved in the queue. Among queue indices lower than TDB value of a failed packet to move, if there are one or more empty queue indices, then they can be moved. However, if no empty queue index, then the failed packets cannot move. Any failed packet that cannot be moved is discarded from the queue. A packet of which TDB is 0 is discarded since there is no 0th queue index.

A failed packet also moves to a queue index corresponding to its TDB value. If there is already another packet at the position, then every packet with TDB value less than that of the failed packet is moved forward by a slot. And then, the failed packet moves to the queue index. In doing so, packets are placed in the queue in order of their TDB values.

Fig. 6 shows an example of how to adjust queue position of failed packets. In Fig. 4, failed packets are PKTs 5, 6, 7, and 8. TDB values of these packets are 4, 2, 1, and 6, respectively. They are adjusted after completing the adjustment of queue positions of successfully transmitted packets. PKT 8 with the largest TDB value moves first. In Fig. 5(e), since there are no packets at queue indices 1 and 6 among those with TDB value less than that of PKT 8, PKT 8 can be moved. Therefore, PKT 8 moves to queue index 6 corresponding to its TDB value (Fig. 6(a)). Then, we move PKT 5 with the next largest TDB value. Since there is no other packet at queue index 1 among queue indices with less than its TDB value, this packet can also be moved. Since TDB value of this packet is 4, it moves to queue index 4. However, PKT 1 is already placed at queue index 4. Thus, PKTs 2 and 3 with TDB values less than that of PKT 5 are moved forward by a slot, respectively. Then, there is no
packet in queue index 3. We move PKT 5 to queue index 3 (Fig. 6(b)). Next, we conduct the same operation to PKT 6. However, every queue index under the TDB value of this packet is already filled with a packet, this packet cannot move. Therefore, it is discarded. PKT 7 is also discarded since it cannot be moved.

In the above two steps, we complete the adjustment of failed packets. Before adjusting the packet transmission sequence, 5 packets are transmitted successfully and 4 packets failed. However, after the adjustment, 7 packets are successfully, while 2 packets failed. Thus, it seems that the proposed scheme can enhance the performance.

C. Step 3: Continuous Packet Arrangement

In the above two steps, we complete the adjustment of every packet that can be transmitted within its delay bound. However, while moving packets, there may be empty queue indices at which no packets are placed (e.g., queue indices 7 and 8 in Fig. 6(b)). That is, packets in the queue are arranged discontinuously. To remove such empty indices, packets behind the empty queue index are moved forward.

Fig. 7 shows an example of continuous packet arrangement. In Fig. 6(b), there are no packets at queue indices 7 and 8. To remove the empty indices, PKT 9 is moved to queue index 7. Thus, every packet in the queue is arranged continuously.

IV. PERFORMANCE EVALUATION

Let us discuss the simulation results of the proposed DBTSA scheme. For comparison purposes, we devise a conceptual scheme called PDDB (Packet Disuse based on Delay Bound). After obtaining the successful transmission interval, the PDDB scheme checks whether or not the head packet in the queue meets its own delay bound. The scheme transmits the head packet if it meets, or otherwise discards the packet and then checks the next packet in the queue again to transmit.

In the simulation, there are two types of traffic: multimedia and background. On the Internet, there are two real-time multimedia servers. 2 multimedia stations receive data packets sent by the real-time multimedia servers through the AP. They have different delay bounds. To distinguish between them, we call traffic with short and long delay bounds short flow and long flow, respectively. The background traffic carried in EDCA mode is used to observe its effects on real-time multimedia traffic. We assume that all the stations are within the transmission range, and the channel is error free.

System parameters used in the simulation are listed in Table I. We simulated an IEEE 802.11a network with transmission rates of 54Mbps for data packets and 6Mbps for control packets.

<table>
<thead>
<tr>
<th>Packet Size (Octets)</th>
<th>1500</th>
<th>1500</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHY PLCP Header Length (Octets)</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>PHY PLCP Preamble Length (us)</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>ACK (Octets)</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Queue Size (Packets)</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Smoothing Factor</td>
<td>0.9</td>
<td>0.9</td>
</tr>
</tbody>
</table>

The traffic parameters are listed in Table II. We use the negative exponential distribution to get the lengths of the data packet inter-arrival times. The average inter-arrival time of the distribution with arrival rate parameter $\lambda$ is $1/\lambda$. For background traffic, the average inter-arrival time is set to 5000 us ($\lambda=0.0002$). Thus, each background station generates data packets at a rate of 2.4Mbps. For real-time multicast multimedia traffic, the arrival rate parameter is set to 2500 us ($\lambda=0.0004$). To consider the burstyness of the multimedia traffic, the number of packets to be generated is randomly chosen from [1, 5] at each arrival. Therefore, the multimedia traffic sends data packets at a rate of 14.4Mbps.

Fig. 8 shows the simulation results for normalized throughput according to the number of background traffic stations. The delay bounds for the long and short flows are fixed to 20 ms and 15 ms, respectively. From the figure, we
see that for the proposed scheme short and long flows have similar normalized throughput regardless of the variation of the number of background traffic stations. However, PDDB scheme shows the big performance difference between the long and short flows. The proposed scheme reduces the performance difference although the long flow still has slightly higher performance than the short one. This means that the DBTSA scheme alleviates the fairness problem.

Fig. 8. Normalized throughput according to the number of background traffic stations.

Fig. 9 shows the simulation results for normalized throughput according to the delay bound of the long flow. There are 4 background traffic stations. For the short flow, the delay bound is fixed to 15ms. For the long flow, the bound varies from 20ms to 15ms. In the PDDB scheme, as delay bound of the long flow is getting smaller, the performance of the long flow is deteriorated, while the performance of the short flow is enhanced progressively. When the delay bounds of the two flows are equal, their performance is identical. However, the proposed scheme shows somewhat different result with the PDDB scheme. Until delay bound of the long flow is reducing to 16ms, the performance of the long flow enhances slightly, while that of the short flow deteriorates somewhat. Its cause is as follows: when delay bound of the long flow is larger, it is possible to move packets backward in the queue to adjust the transmission sequence for the short flow. Therefore, some packets of the short flow, which may be failed, move forward and are transmitted successfully. However, as delay bound of the long flow is getting smaller, the long flow packets can be still transmitted within their delay bounds, but the probability of yielding the queue sequence to the short flow is getting less. Therefore, packets in the long flow do not move backward, so those in the short flow, which may be failed, cannot move forward and then fail to transmit. As a result, the performance of the long flow is getting better somewhat, but that of the short flow is getting deteriorated. However, in the proposed scheme, the difference in the two flows is still smaller than that of the PDDB scheme.

V. CONCLUSION

Real-time multimedia traffic is very sensitive to delay so that it has to be transmitted to receivers within its delay bound. Otherwise, it is discarded at the receivers. Under the environment where real-time multimedia traffic has different delay bounds, long delay bound packets have the benefit of good performance. To alleviate the fairness problem, we propose a DBTSA scheme which rearranges the transmission order of data packets in the queue based on delay bound. In the proposed scheme, the packets with long delay bound yield to the packets with short delay bound. That is, a packet with long delay bound is moved to backward in the queue as far as its delay bound allows, and a packet with short delay bound moves forward. The simulation results show that the proposed scheme alleviates the limitations.

REFERENCES

Young Ung Kim received the B.E. degree in information and communication engineering from National Institute for Lifelong Education, Seoul, Korea, in 2011. Currently, He is pursuing the M.S. degree in the Department of Computer IT Engineering from Kumoh National Institute of Technology, Gumi, Korea. Since 2008, He has been working for Samsung Electronics, Gumi, Korea. His research interests include wireless LANs, quality of service enhancement and smart phone.

Sunmyeng Kim received the B.S., M.S., and Ph.D. degrees in information and communication from Ajou University, Suwon, Korea, in 2000, 2002, and 2006, respectively. From May 2006 to February 2008, he was a postdoctoral researcher in electrical and computer engineering with the University of Florida, Gainesville. In March 2008, he then joined the Department of Computer Software Engineering, Kumoh National Institute of Technology, Gumi, Korea, as an assistant professor. His research interests include resource management, wireless LANs and PANs, wireless mesh networks, UWSNs and quality of service enhancement.