International Journal of Networking and Computing – www.ijnc.org, ISSN 2185-2847
Volume 8, Number 2, pages 367–386, July 2018

A Contention Window Control Method Using Priority Control Based on the Number of Freezes of Wireless LAN

Tomoki Hanzawa
Graduate School of Systems and Information Engineering,
University of Tsukuba
1-1-1 Tennodai, Tsukuba, Ibaraki, 305-8573, Japan

and

Shigetomo Kimura
Faculty of Engineering, Information and Systems,
University of Tsukuba
1-1-1 Tennodai, Tsukuba, Ibaraki, 305-8573, Japan

Received: February 1, 2018
Revised: May 5, 2018
Accepted: June 11, 2018
Communicated by Eitaro Kohno

Abstract

Because of the widespread adoption of mobile devices, many applications now provide support for wireless LAN (WLAN). This makes it an important issue to provide good WLAN quality of service (QoS). For this purpose, Dhurandher et al. improved the distributed coordination function (DCF). Although the highest-priority throughput increased using this method, throughput for other priorities decreased significantly. The authors proposed a minimum contention window control method based on the collision history for two (high and low) priorities, to improve Dhurandher's method. That method can keep the same average throughput for high-priority traffic as Dhurandher's method yields, but decreases the average total throughput of not only the low-priority traffic but also the high-priority traffic when congestion occurs. To overcome this problem, the method introduced in this paper improves the contention window control method by using a priority control based on the number of freezes, which is defined as a count of the times that a mobile node lets its backoff timer be frozen by the other nodes. The number of freezes is introduced as a more suitable indication of congestion, and to control the contention window for the low priority. Finally, network simulations executed on two scenarios using continuous flows or on-off flows for the low-priority and demonstrated that the proposed method retains the average total throughput of both priority frames as well as reducing the number of freezes of both priorities and the packet delay and jitter for high-priority frames, compared with the controls in Dhurandher's method and the previous method. However, the fairness-index for low-priority flows showed a little unfair in the proposed method.

Keywords: Number of Freezes, Contention Window Control Method, Priority Control, DCF, Wireless LAN

1 Introduction

Recently, wireless LAN (WLAN) has become a major communication avenue as WLAN devices became cheaper and featured improved performance. WLAN is usually used within the home and office, while public WLAN services are also popular. As mobile devices such as smartphones and tablets have become widespread, various Internet services are also accessed via WLAN. Under the circumstances, it is important to provide good WLAN quality of service (QoS). For example, for real-time applications like voice or video streaming, if the packet loss and/or jitter increase, then the QoS rapidly decreases. Therefore, the packets from such real-time applications should be transmitted with a higher priority than data packets. However, WLAN that is based on IEEE 802.11 cannot guarantee QoS because it processes these packets in simple first-in first-out order. Various methods have been proposed to guarantee QoS in WLAN. The IEEE 802.11 working group proposed IEEE 802.11e and IEEE 802.11aa for this purpose, in which various methods are suggested to guarantee QoS [1], [2]. However, because these standards are optional, most of the devices in the consumer market do not support them. Thus, it is difficult to adopt these methods on a typical WLAN network. Furthermore, in [2] it is reported that throughput is not sufficient when IEEE 802.11e and IEEE 802.11aa are used in combination. Therefore, it is impossible for a public WLAN to guarantee QoS since many unspecified users exist. Many methods have also been proposed for WLAN devices that do not use IEEE 802.11e and/or IEEE 802.11aa to guarantee QoS. In [3], traffic is categorized into two priority classes. When a low-priority frame is sent, this method generates a random number. If this number exceeds a threshold, the frame is allowed to attempt to send. In [4], in an environment where data packets and real-time packets are queued together for sending, before a packet is inserted into the queue on a WLAN access point, it may be discarded based on a probability calculated using the queue length. In [5], Dhurandher et al. improved the distributed coordination function (DCF), which is the mandatory media access control in WLAN. In this method, the contention window (CW) is divided into multiple ranges. Each range is independent of all other ranges and is assigned to a different priority. Although the throughput of the highest priority frames increased using this method, throughput for the other priorities was significantly decreased. The authors proposed a minimum contention window control method for two (high and low) priorities [6]. In the following, this method is referred to as the previous method. The previous method keeps the CW for high-priority traffic at a low value, and controls the CW for low-priority traffic based on the collision history. The method can maintain the average throughput for high-priority traffic to equal Dhurandher's method, but decreases the average total throughput of not only the low-priority traffic but also the high-priority traffic when congestion occurs.

To overcome this problem, this paper improves the previous method based on the number of freezes, which is defined as a count of the times that a mobile node lets its backoff timer be frozen, or be isolated from the other nodes. In this method, all nodes are assumed to process real-time applications such as voice and video streaming, as well as data transmission such as FTP, HTTP, etc. The former real-time frames are high priority and are sent by UDP. The latter data frames are low priority and are sent by TCP. The purpose of the proposed method is not only to provide good QoS for the high-priority frames, but also to prevent decreasing the QoS for low-priority frames. For this purpose, the number of freezes is introduced as a more suitable way to take into account the level of congestion and to control the CW for low-priority traffic. Finally, network simulations on two scenarios using continuous flows or on-off flows for the low-priority quantified the average total throughput and the number of freezes for TCP and UDP flows, and the average total packet delay and jitter for UDP flows, the fairness index for TCP flows. The results showed that the proposed method maintains the average total throughput of both priorities, as well as reducing the number of freezes of both priorities and the packet delay and jitter for high-priority frames, compared with using Dhurandher's method or the previous method. However, the fairness-index also showed a little unfair for low-priority flows in the proposed method.

The rest of this paper is organized as follows. Section 2 introduces DCF, Dhurandher's method, and the previous method. Section 3 proposes a contention window control method using priority control based on the number of freezes. Section 4 presents the communication experiments. Section 5 concludes the paper and describes future work.

2 Contention Window Control Method for QoS

This section introduces the distributed coordination function (DCF), Dhurandher's method [5], the previous method [6].

2.1 Distributed Coordination Function (DCF)

The distributed coordination function (DCF) is the mandatory distributed channel access method defined in IEEE 802.11, which is also called carrier sense multiple access with collision avoidance (CSMA/CA). Suppose the infrastructure network has an access point (AP) and nodes A and B. As shown in Figure 1, when all nodes including the AP try to send a frame, each node senses a carrier on the channel. If the channel is idle for the duration of the DCF interframe space (DIFS), each node generates a random number and starts a backoff timer, the expiration time of which is the slot time multiplied by a random number. All nodes decrease their own backoff time. In Figure 1, node A transmits a data frame because its backoff timer reaches zero first. The other two nodes detect the carrier and then stop decreasing their own backoff times.

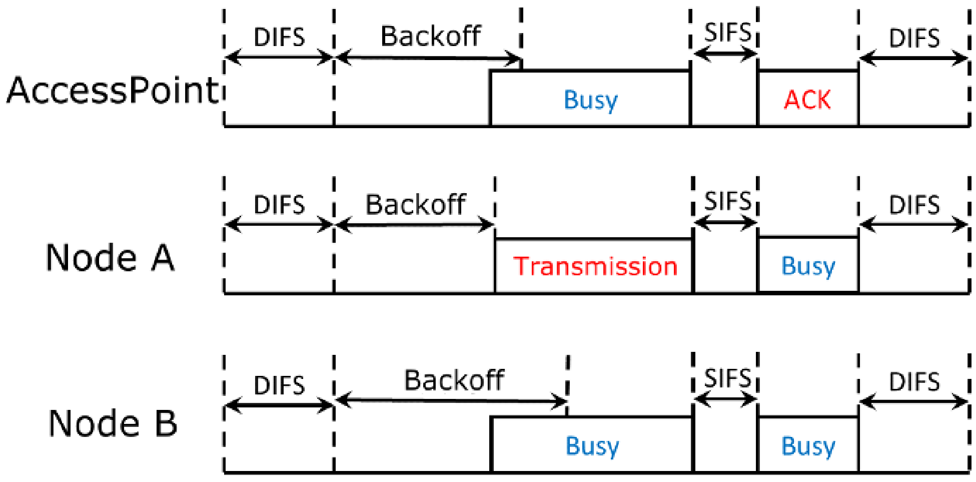


Figure 1: Media Access Control by CSMA/CA

When a data frame is received correctly, the receiver waits for the duration of the short interframe space (SIFS) and then transmits an acknowledgment (ACK) control frame. Only when node A receives the ACK control frame does the transmission succeed. When the ACK control frame is broken, or more than two data frames are transmitted by node A and another node because their backoff timers reach zero simultaneously, then a collision occurs such that both frame transmissions will fail. After the channel is idle for the next DIFS period, node A proceeds with a countdown to a new transmission time by generating a random number as follows to retransmit the frame, while the AP and node B also restart decreasing their own backoff times.

In the above procedures, a random number is generated between 0 and CW. CW ranges from the minimum CW ($\mathrm{CW}_{\min}$) to the maximum CW ($\mathrm{CW}_{\max}$) and is represented by the following equation:

$$\mathrm{CW} = \min((\mathrm{CW}_{\min} + 1) \times 2^n - 1, \mathrm{CW}_{\max}) \tag{1}$$

where n is the number of successive collisions. When a node first tries to transmit a frame, a random number is generated from 0 to $\mathrm{CW}_{\min}$. For example, $\mathrm{CW}_{\min}$ is 7 in Figure 2. The CW range of the random number for the first transmission is from 0 to 7 as in the top row in the figure. When the first transmission fails, the CW range of the random number doubles as shown in the middle row in Figure 2. If the second transmission also fails, the range doubles as shown in the bottom row in the figure. After the frame is transmitted correctly, the sender resets CW to $\mathrm{CW}_{\min}$. In this way, if the transmission continues to fail, then CW is exponentially increased to decrease the probability of collision.

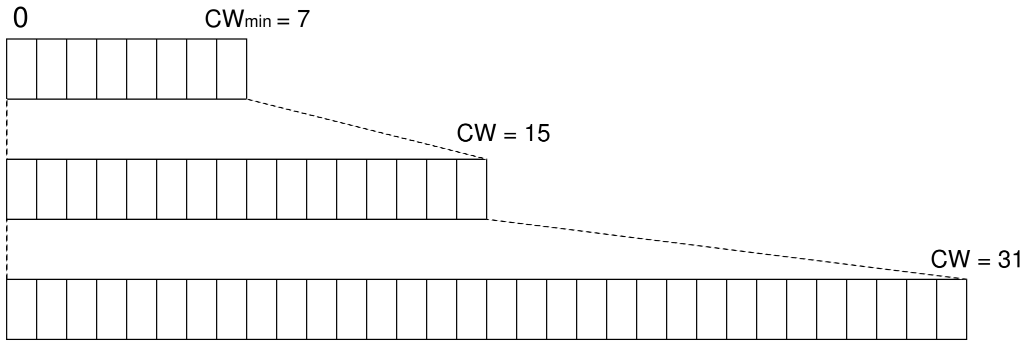


Figure 2: Expansion of CW range by collisions.

The above algorithm is called binary exponential backoff (BEB), which has also been adopted for the Ethernet. Since the backoff algorithm has an impact on throughput in WLAN, several backoff algorithms have been proposed. For example, the multiplicative increase and linear decrease (MILD) algorithm slows down the change of CW, and smart exponential threshold linear (SETL) controls CW based on a threshold and the transmission success times [7]. However, the more nodes that try to send a frame, the more collisions can occur. This causes longer backoff times and decreases WLAN throughput [8]. Moreover, the current DCF is designed such that all nodes are equally likely to be able to transmit their frame. This fact also causes a transmission imbalance between uplink and downlink, so that no application can guarantee QoS [9], [10].

2.2 Dhurandher's method

In [5], a contention window control method is proposed to guarantee QoS. Although DCF generates a random number regardless of the priority of a frame, in this method the CW range is divided into non-overlapping independent priority levels. For example, the CW range is divided into four priority levels as shown in Figure 3. When a node sends a frame at the highest priority, it generates a random number within the range of Priority 0. When a node sends at the lowest priority, the number is generated within Priority 3. When a collision occurs, the CW range doubles, but the priority levels do not overlap with each other, as shown in the middle and bottom rows in Figure 3. By generating random numbers from different ranges based on the priority, specific frames at higher priorities can be transmitted rapidly.

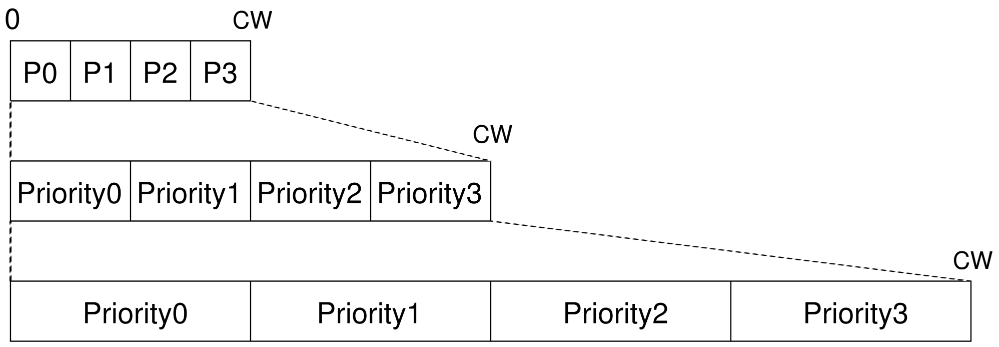


Figure 3: CW range divided by priority levels.

Using this method, throughput at Priority 0 is improved. However, since Priority n must always wait for frames ranging from Priority 0 to Priority $n-1$, the throughput for other priorities is significantly lower than that of Priority 0.

2.3 Previous method

To solve this problem, the authors improved Dhurandher's method for two (high and low) priority levels.

2.3.1 Contention window range at first transmission

Since Dhurandher's method divides the CW range into non-overlapping independent priority levels, throughput except at the highest priority level is greatly decreased. The authors proposed that the low-priority CW range should include some of the high priority, as shown in Figure 4. For the first transmission, the latter range is the lower half of the former range.

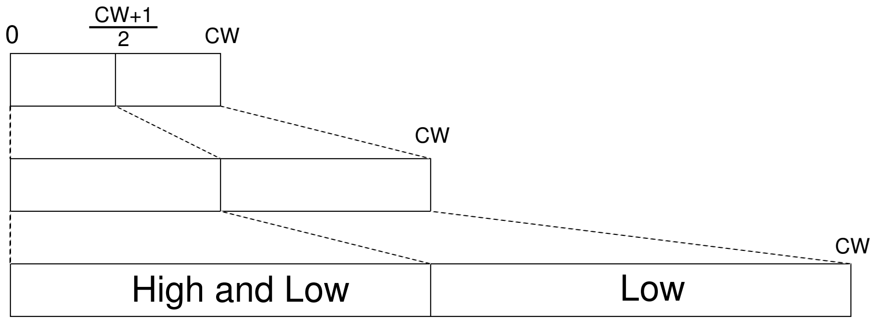


Figure 4: Expansion of CW range by previous method

When collision occurs, each CW range doubles, as in DCF. Then, the CW of each priority level is defined as:

$$\mathrm{CW} = \begin{cases} \min\left(\dfrac{(\mathrm{CW}_{\min}+1)\times 2^n}{2}-1, \dfrac{\mathrm{CW}_{\max}+1}{2}-1\right) & \text{(High Priority)} \\ \\ \min\left((\mathrm{CW}_{\min}+1)\times 2^n - 1, \mathrm{CW}_{\max}\right) & \text{(Low Priority)} \end{cases} \quad (2)$$

where $\mathrm{CW}_{\min}$ and $\mathrm{CW}_{\max}$ are the initial and maximum values of CW defined in IEEE 802.11 and n is the number of successive collisions. From the above equation, a node generates a random number from 0 to $((\mathrm{CW}_{\min}+1)\times 2^n)/2-1$ to transmit a high-priority frame. If a collision occurs, the CW range doubles until the range reaches $((\mathrm{CW}_{\max}+1)\times 2^n)/2-1$. When the node transmits a low-priority frame, it generates a random number from 0 to $(\mathrm{CW}_{\min}+1)\times 2^n-1$ as defined in IEEE 802.11. If collisions occur, CW doubles, up to $\mathrm{CW}_{\max}$. For the high priority, the CW range is always half of that of the low priority, i.e., that of DCF, for the same n, until the first transmission succeeds. Therefore, the probability of transmitting a high-priority frame is greater than for a low-priority frame. In addition, since the low-priority CW range is included in the high range, a low-priority frame can be transmitted more easily than in Dhurandher's method.

2.3.2 Contention window range after transmission succeeds

As shown in Figure 5, the previous method also changes the CW range after a node succeeds in transmitting the frame. In Dhurandher's method, after transmission succeeds, the CW range is reset to the initial range of each priority. In this method, the high-priority CW range is also reset from 0 to$(\mathrm{CW}_{\min}+1)/2-1$, but that of low priority is set from 0 to half of the current CW. If the half-CW value is smaller than $\mathrm{CW}_{\min}$, then the value becomes $\mathrm{CW}_{\min}$. The range is based on the collision history because the range is too wide to reduce the probability of transmitting low-priority frames when the node experiences a high collision rate for them. When the node does not send a low-priority frame for the period of CW $\times$ *Slot_time*, the proposed method reduces the current CW range by half. Using the above control, low-priority frames do not significantly prevent high-priority frames from transmitting continuously.

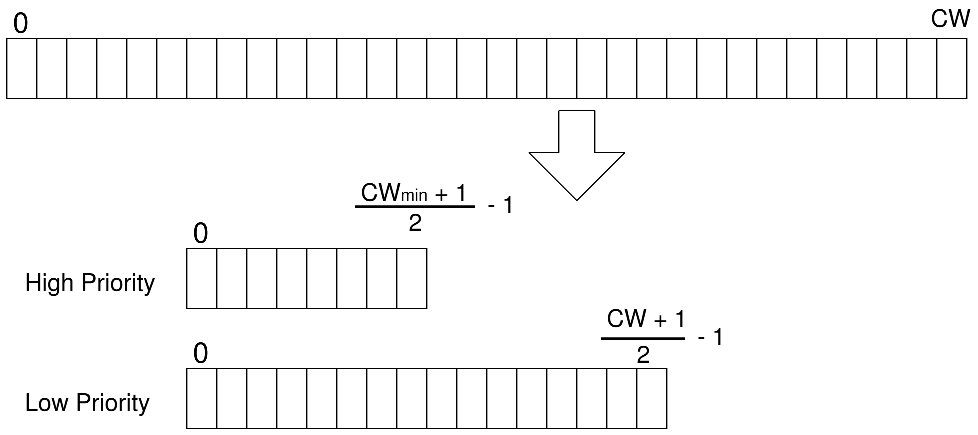
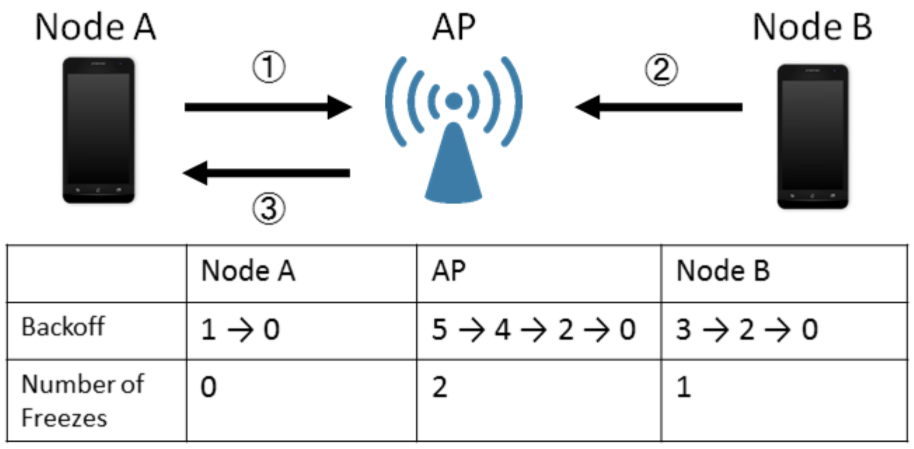


Figure 5: CW range after transmission succeeds

3 A contention window control method using priority control based on the number of freezes

This section assumes two (high and low) priorities, and proposes a contention window control method using priority control based on the number of freezes to improve on the previous method. In the proposed method, all nodes are assumed to use real-time applications, such as voice and video streaming, or data communication such as FTP, HTTP, etc. The former real-time frames are high priority and are sent by UDP. The latter data frames are low priority and are sent by TCP.

3.1 The number of freezes



	Node A	AP	Node B
Backoff	1 → 0	5 → 4 → 2 → 0	3 → 2 → 0
Number of Freezes	0	2	1

Figure 6: Backoff and Number of Freezes under CSMA/CA

The number of freezes is defined as a count of the times that a mobile node lets its backoff timer be frozen by the other nodes. Assume that the topology in Figure 6 includes two nodes, A and B, within the cell of an access point AP. In the figure, AP and nodes A and B sense a carrier and then find that the channel is idle. After they generate a random number, they begin their backoff procedure. In this case, since node A, AP, and node B generate 1, 5 and 3, respectively, node A, with the minimum backoff time, can be the first to send a frame. Upon receiving the carrier from node A, AP and node B must freeze the countdown on their backoff timers. After node A finishes sending the frame, AP and node B restart the countdown on their backoff timers. In the figure, since node B's backoff timer is the next to reach zero, it starts to send a frame, but AP must again freeze its backoff timer. As a result, the freezes at node A, AP, and node B number 0, 2, and 1, respectively. When the number of nodes trying to send a frame increases, the number of freezes will also increase. It should be noted that nodes can easily calculate that number, because they must always listen to the shared medium to know when to stop their backoff timers when the medium is busy in CSMA/CA.

3.2 Contention windows control method based on the number of freezes

In the proposed method, the CW range for both priorities at first transmission and the CW range for the high priority after transmission succeeds are the same as in the previous method. The previous method decreases the throughput of low-priority traffic, because the CW range of the low-priority traffic cannot easily be reduced to maintain the throughput of the high-priority traffic. The proposed method controls the CW range of the low priority based on the number of freezes as follows.

After a node succeeds in transmitting a low-priority frame, the CW range is set from 0 to CW′ defined as:

$$\mathrm{CW}' = \max(\mathrm{CW}_{\min}, \min(\mathrm{CW}_{\max}, \frac{\mathrm{CW}+1}{2^{\max(0,n_{\max}-n)}} - 1)) \tag{3}$$

where CW is the current contention window, n is the number of freezes, and $n_{\max}$ is the maximum retransmission time. $\mathrm{CW}_{\min}$ and $\mathrm{CW}_{\max}$ are the initial and maximum values of the contention window defined in IEEE 802.11. If $n_{\max}$ is lesser than or equals n, then CW′ keeps the same value as the current CW. CW′ is a power of 2 minus 1 and changes within the range of $\mathrm{CW}_{\min}$ to $\mathrm{CW}_{\max}$ in the same way as CSMA/CA. As shown in Equation (1), CW forms into the expression of $(\mathrm{CW}_{\min}+1) \times 2^m - 1$ for the number m of successive collisions. To provide compatibility with terminals that do not support the proposed method, CW′ in Equation (3) is suitable to form into the expression. Moreover, to reduce CW′ quickly even when CW is large but the number n of freezes is small, the proposed method adopted the approach dividing by an exponential value $2^{\max(0,n_{\max}-n)}-1$ to update CW′.

When the number n of freezes is smaller, we can consider that the congestion is lighter. In this situation, from Equation (3), since $n_{\max}-n$ is larger, $2^{\max(0,n_{\max}-n)}$ becomes sufficiently large value. Thus, CW′ becomes a quite small value to transmit the next frame in a short time. In contrast, when the number n of freezes is larger, $2^{\max(0,n_{\max}-n)}$ becomes sufficiently small value or 1 from the max function, and then CW′ keeps its larger value so as to wait longer to transmit the next frame and thus promote the transmission of high-priority frames. In the previous method, the CW value is always maintained at a larger value when congestion occurs. However, the proposed method can change the CW value based on the degree of congestion to provide better throughput when needed. It should be noted that a low-priority flow can still generate a small random number from the CW range, even if the CW value becomes sufficiently large. Therefore, some lucky low-priority flow may continue to get small random numbers to transmit frames before it is frozen, and maintain its narrow CW range. This fact may give bad effect to the fairness of low-priority flows.

For example, when many high-priority flows are generated as shown in Figure 7, CW of the low-priority flow is 1023 ($= \mathrm{CW}_{\max}$) and the number of freezes is 7 ($= n_{\max}$), which means the low-priority flow is frozen 7 times by the high-priority flows. Even in such highly congested situation, CW of the high priority flows is reset to the initial CW, if they succeed to transmit a frame. However, since $2^{\max(0,n_{\max}-n)} = 1$, CW′ of the low priority flow keeps 1023 even if it succeeds to transmit.

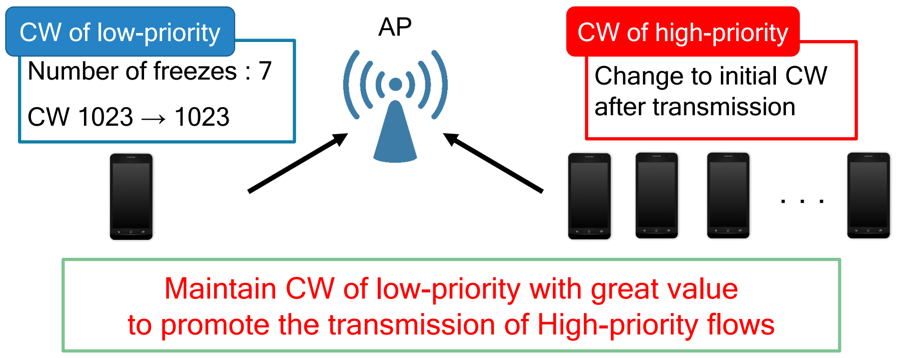

CW of low-priority
Number of freezes : 7
CW 1023 → 1023
AP
CW of high-priority
Change to initial CW after transmission
. . .
Maintain CW of low-priority with great value to promote the transmission of High-priority flows

Figure 7: Proposed method at high congested situation

Thus, the proposed method maintains CW of the low priority with great value to promote the transmission of the high-priority flows.

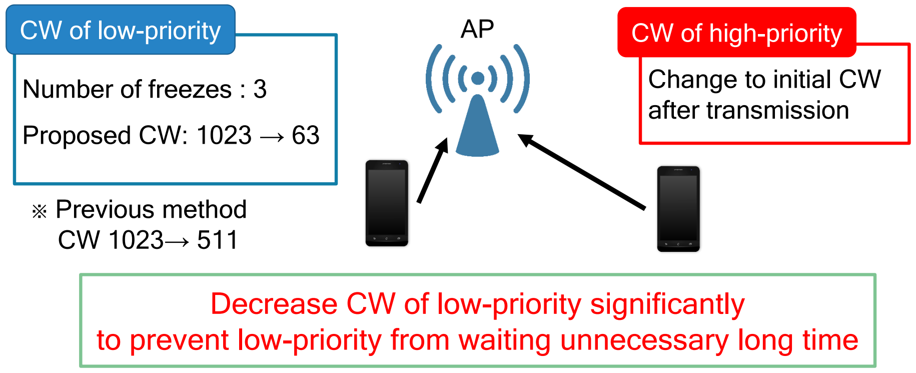

CW of low-priority
Number of freezes : 3
Proposed CW: 1023 → 63
※ Previous method CW 1023→ 511
AP
CW of high-priority
Change to initial CW after transmission
Decrease CW of low-priority significantly to prevent low-priority from waiting unnecessary long time

Figure 8: Proposed method at less congested situation

On the other hand, when several high-priority flows are generated as shown in Figure 8, CW of the low-priority flow is 1023 ($= CW_{\max}$) but the number of freezes is only 3. Then, in the proposed method, since $2^{\max(0, n_{\max}-n)} - 1 = 16$, CW' changes from 1023 to 63, if the flow succeeds to transmit a frame. However, in the previous method, CW' changes from 1023 to 511. Therefore, in such less congested situation, the proposed method can decrease CW' significantly to prevent low-priority flows from waiting for unnecessary long time.

4 Simulation experiments

This section describes the evaluation of the proposed method, which uses Network Simulator 3 to implement the proposed method, executing two simulation experiments referred as Scenarios 1 and 2 to compare the results of the proposed method with Dhurandher's method and the previous method.

4.1 Simulation environment

Table 1 lists the experimental conditions. The network topology is shown in Figure 9.

In both scenarios, IEEE 802.11g is used, because it is well supported in Network Simulator 3. UDP flows are assumed to be bidirectional voice communications between the access point (AP) and nodes. The UDP packet size is 320 bytes and the flow is a constant bit rate (CBR) of 64

Kbps. TCP flows continuously transmit frames as many as the TCP congestion window allows in Scenario 1 to give the heavy load over the wireless link. On the other hand, TCP flows are alternated between transmission interval and stop interval in Scenario 2 to provide more practical data transmissions. The both of transmission interval and stop interval obey on the logarithmic normal distribution where the mean is 1.0 second and the standard deviation is 0.07 seconds to generate the heavy-tailed distribution. The simulation time is 100 seconds.

Scenario 1 will show how each method will maintain high-priority flows and will reduce the average total throughput and fairness-index of low-priority flows in the heavy load. However, in average usages, such heavy load is not often appeared. Thus, Scenario 2 will show the effectiveness of each method for the average usages that may include the heavy-tailed traffic such as a transmission of huge files.

Table 1: Experimental Conditions

Simulator	Network Simulator 3
Wireless LAN Standard	IEEE 802.11g
Data Rate	54Mbps
Minimum CW	15
Maximum CW	1023
Maximum Retransmission Times	7 times
Simulation Time	100 seconds
Simulation Count	10 counts
Node's Queue type	Single Drop-Tail Queue
Node's Queue length	100 Packets
UDP Traffic Pattern	Real Time Application (VoIP)
UDP Packet Size	320 bytes
UDP Bit Rate	64Kbps (CBR)
TCP Traffic Pattern	FTP
TCP Maximum Segment Size	1460 Bytes

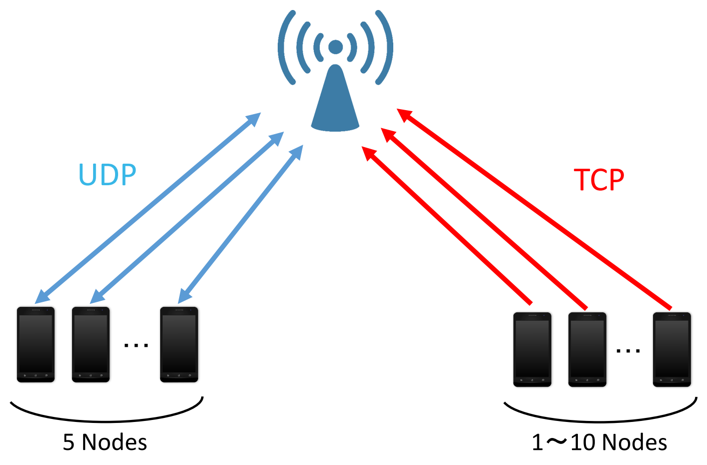


Figure 9: Simulation Topology

In the simulation, five bidirectional UDP flows are fixed as shown in Figure 9. The number of TCP uplink flows from the nodes to AP is varied from 1 to 10 to show the influence of the low-priority uplink flows.

Under the experimental conditions, the simulation is executed 10 times to determine the average and 95% confidence interval of the total throughput for both priority flows and packet delay, packet drop rate and jitter for the high-priority flows. These values are monitored by the flow monitor module in Network Simulator 3 [11]. Moreover, the fairness-index is monitored based on Jain's fairness-index [12] to evaluate the fairness of low-priority flows.

4.2 TCP total throughput and Number of freezes

Figure 10 and Figure 11 show the average total TCP throughputs as the line graphs, and those of the number of freezes as the bar graphs in Scenarios 1 and 2, respectively.

In Figure 10, when the number of TCP uplink flows was 1, the average total TCP throughputs of the previous method and the proposed method were 11.7 Mbps. Since there are only five bidirectional UDP flows with a narrower bandwidth except the TCP flow, the throughput is considered to be the maximum for TCP flows in the network environment. In Dhurandher's method, the average total throughput was 11.3 Mbps, which is a little smaller than those of the other two methods, since the method always generates a large random value from the upper half of the CW range for the low-priority flows, even when there is little congestion. As the number of TCP flows increases, the proposed method kept the average total throughputs between 11.6 and 11.7 Mbps, while the previous method decreased the range to 10.7 to 11.3 Mbps. When the number of TCP uplink flows was greater than or equal to 8, the average total TCP throughputs of Dhurandher's method were greater than those of the proposed method, since the average total UDP throughputs of Dhurandher's method decreased, as we will see later. This is because Dhurandher's method always reset CW to the minimum CW, and TCP flows are transmitted faster than in the other methods, so the opportunity to transmit UDP flows decreased. The average number of freezes in the proposed method was up to 6.1, however that in Dhurandher's method and the previous method were up to 9.8 and 13.1, respectively.

In Figure 11, the average total throughputs increased linearly, but those of the proposed method increased more rapidly than the other methods. The average total throughputs of the proposed method were from 0.17 to 0.81 Mbps higher than that of Dhurandher's method when the number of TCP uplink flows is from 1 to 10, and from 0.17 to 0.85 Mbps higher than that of the previous method when the number of TCP uplink flows is from 2 to 10. The number of freezes of the proposed method was also lower than those of the other methods.

The difference in Scenarios 1 and 2 is as follows. The average total throughput of all methods in Figure 10 is always larger than that in Figure 11, since the congestion in Scenario 1 is heavier than that in Scenario 2. When the number of TCP uplink flows is 1 in the both figures, the average number of freezes of all methods is about 2.8, which is considered as the minimum number frozen from the high-priority flows even if the congestion is lighter. When the number of TCP uplink flows is larger than 1, the average number of freezes of Dhurandher's method and the previous method in Figure 10 from 0.9 to 3.0 higher than that in Figure 11, however the former of the proposed method is at most 0.5 higher than the latter. From these results, the proposed method yields more effective transmission than the other two methods and limits increasing the number of freezes for heavy congestion.

4.3 UDP total throughput and number of freezes

Figure 12 and Figure 13 show the average total UDP throughputs and those of the number of freezes in Scenarios 1 and 2, respectively. It should be noted that the throughput range of the vertical axis in Figure 13 is different from that in Figure 12.

In Figure 12, the average UDP throughput in the proposed method remained at about 0.64 Mbps (= 64Kbps $\times$ 5 $\times$ 2) because the UDP bit rate is small and the nodes that transmit UDP datagrams can generate a random number for transmitting independently. In Dhurandher's method, the throughput of the UDP flows decreased because only the AP can generate a random number for all the UDP downlink flows while all the nodes create their own random numbers for each other for all the UDP and TCP uplink flows. Therefore, all the UDP downlink flows in this method have to wait longer than in the other two methods. Those of the previous method also kept this value, which was 0.01 Mbps smaller than that of the proposed method when the number of TCP uplink flows was 10. Moreover, those in Dhurandher's method ranged from 0.45 to 0.58 Mbps when the number of TCP uplink flows was greater than or equal to 5, since the many TCP uplink flows obtained their throughput from the high-priority UDP flows. The average number of freezes in the proposed method was as high as 4.0, however those in the Dhurandher's method and the previous method rose to 7.8 and 7.0, respectively.

In Figure 13, the results were almost same as in Figure 12. However, since the load of TCP uplink flows not so high, The average total UDP throughputs of Dhurandher's method were only about 0.06 Mbps lower than the other methods. From these results, the proposed method also provides more effective transmission for UDP flows than do the other two methods.

The difference in Scenarios 1 and 2 is as follows. For Dhurandher's method and the previous method, the average total UDP throughput and the number of freezes in all methods in Figure 12 are equal to or larger than those in Figure 13, since the load of TCP uplink flows in Scenario 1 is higher. For the proposed method, the average total UDP throughput in both Scenarios remained at about 0.64 Mbps, but the number of freezes in Scenario 1 was about 0.389 lower than that in Scenario 2. This result shows that the proposed method more restricts the TCP uplink flows to prevent from freezing UDP flows if the congestion is higher.

4.4 UDP average packet loss rate

Figure 14 and Figure 15 show average UDP packet drop rate in Scenarios 1 and 2, respectively. In the figures, the packet loss rate of uplink flows in all methods was small value from 0 to 0.0065% in Scenario 1 and from 0 to 0.22% in Scenario 2. On the other hand, the packet loss rate of downlink flows in previous and proposed method was small value. However, the packet loss rate of downlink flows in Dhurandher's method increased to 57.38% in Scenario 1 and 16.9% in Scenario 2.

This is because this method resets CW of both priorities to minimum CW. Therefore, the opportunity of sending high-priority frames was lower than other methods and high-priority packets were lost because of queue overflow or frame collision.

The average UDP packet loss rate in Figure 15 was much reduced from that in Figure 14, when the number of TCP uplink flows was greater than 3. However, when the number of TCP uplink flows was 10, the average UDP packet loss rate of the previous method was about 2% in both Scenarios. In this situation, TCP ACK segments sent to 10 TCP nodes stayed with UDP packets of the downlinks in the single drop-tail queue of the AP. When these segments and packets filled the queue, new UDP packets were dropped as this result. However, in the proposed method, since the average number of freezes was smaller than that of the previous method, the queuing delay was also shorter. As a result, the queue was not filled and then the average UDP packet loss rate of the proposed method was 0 in both Scenarios.

4.5 UDP average packet delay

Figure 16 and Figure 17 show the average UDP packet delay in Scenarios 1 and 2, respectively. In the figures, the delay for uplink flows in all methods was up to 2.2 msec in Scenario 1 and 3.3 msec in Scenario 2. The delay for downlink flows in the proposed method was as much as 3.1 msec in Scenario 1 and 3.1 msec in Scenario 2. However, in Scenario 1, those in the previous method increased from 12.6 to 185.9 msec when the number of TCP uplink flows was greater than 4. Further, when the number of TCP uplink flows was 4 or more, those of Dhurandher's method increased from 238.7 to 378.1 msec. In Scenario 2, the delay for downlink flows of the previous method increased from 37.8 to 76.8 msec when the number of TCP uplink flows was greater than 7. Those of Dhurandher's method increased from 51.9 to 162.5 msec when the number of TCP uplink flows was greater than 4.

Although ITU G.114 recommends that a one-way delay of 400 msec for speech transmission should not be exceeded, the delay in Figure 16 and Figure 17 included only the transmission time between AP and nodes. If another delay, such as the processing time for a speech codec and a transmission delay within the WAN must be included, the packet delays in Dhurandher's method and the previous method may affect the QoS of speech transmission.

As shown in the previous subsection, when the number of TCP uplink flows was 10, some UDP packets were lost in the previous method in both Scenarios. However, Figure 16 and Figure 17 show that some packets stays in several time in the queue of the AP, when the number of TCP uplink flows was greater than 4. From this result, if the AP replies many TCP ACK segments in a short time in both Scenarios, UDP packets may also be dropped in the previous method even when the number of TCP uplink flows is lower than 10. In contrast, since the average UDP packet delay was almost 0 in both Scenarios, the proposed method may not be affected even if the AP replies many TCP ACK segments in a short time.

4.6 UDP jitter

Figure 18 and Figure 19 show the average jitter in Scenarios 1 and 2, respectively. In the figures, the average jitter in uplink and downlink flows for the proposed method ranges from 1.1 to 1.7 msec in Scenario 1 and from 0.9 to 1.6 msec in Scenario 2. Jitter in uplink flows using the previous method and Dhurandher's method ranged from 2.5 to 4.0 msec in Scenario 1 and from 1.2 to 3.3 msec in Scenario 2, but this increased in downlink flows to a 2.8 to 15.0 msec range in Scenario 1 and a 0.9 to 11.7 msec range in Scenario 2 for the previous method, and a 6.9 to 12.0 msec range in Scenario 1 and a 0.9 to 11.8 msec range in Scenario 2 for Dhurandher's method

The jitter in downlink flows in Dhurandher's method increased rapidly in Scenario 2, because as the packet loss rate decreased in Scenario 2 compared to Scenario 1, the cases that successive packets arrived increased to result the inter arrival time becomes short, but the other cases that intermediate packets dropped also remained. In contrast, the jitter in downlink flows of the previous method in Scenario 1 increased grater than that of the Dhurandher's method, when the number of TCP uplink flows was greater than 5. As shown in Figure 14, the average UDP packet loss rate is small in the previous method. Thus, as the UDP average packet delay increased as shown in Figure 16, the variation of the delay was expanded, and then the jitter in downlink flows of the previous method also increased.

4.7 TCP Fairness index

Figure 20 and Figure 21 show TCP fairness index in Scenarios 1 and 2, respectively. In Scenario 1, the fairness index of the proposed method was 0.08 to 0.10 lower than that of the previous method when the number of TCP uplink flows was greater than 1. In the proposed method, as shown in Section 3.2, frozen nodes tend to expand their CW range for low-priority flows, while some lucky nodes that send frames before they are frozen can keep their narrow CW range. Therefore, since some specific nodes must wait for a long time, the fairness index of the proposed method became a little lower than that of the previous methods.

In Scenario 2, the fairness index of all methods were grater than 0.89, since the load of TCP uplink flows was not so heavy. However, the fairness index of the proposed method in Scenario 2 was greater than that in Scenario 1, when the number of TCP uplink flows was greater than 6. From this result, since the lucky nodes increased in the light congestion, the fairness index of the proposed method decreased.

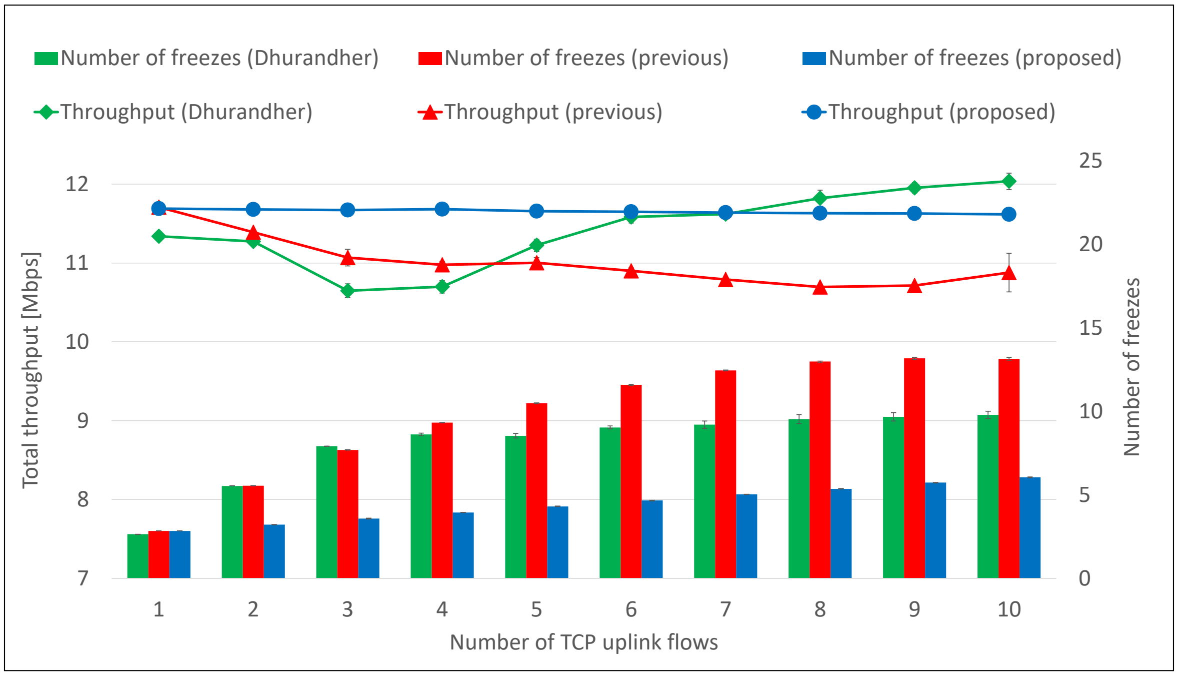


Figure 10: Average total TCP throughputs and number of freezes in Scenario 1

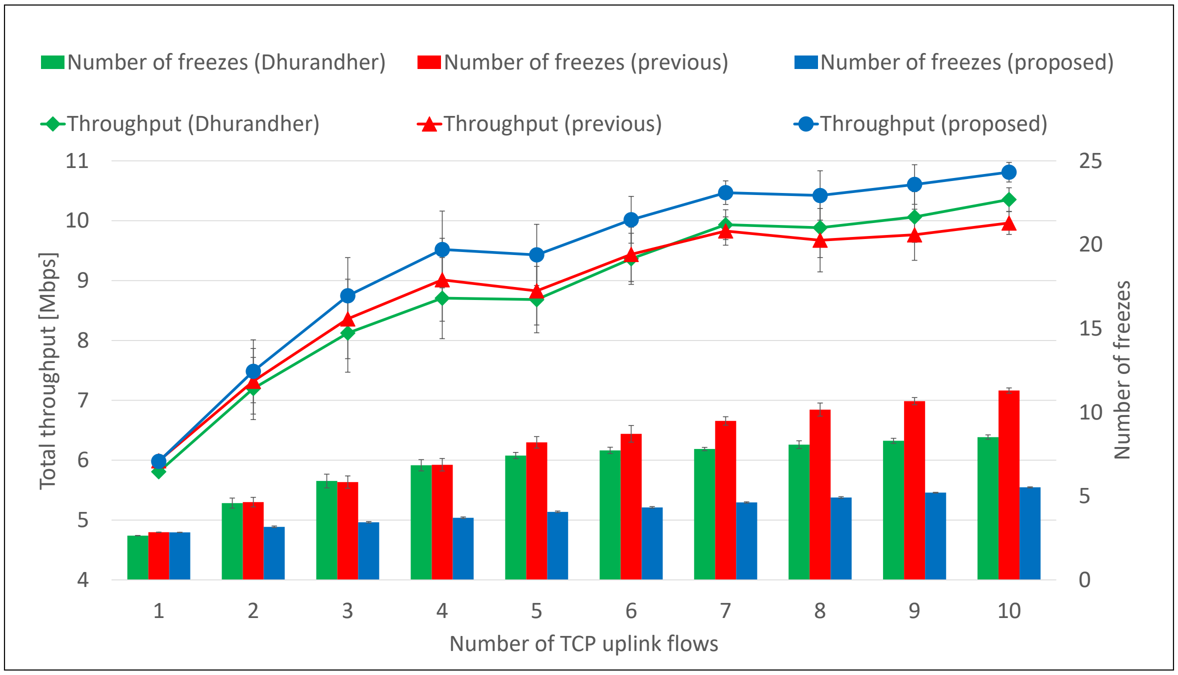


Figure 11: Average total TCP throughputs and number of freezes in Scenario 2

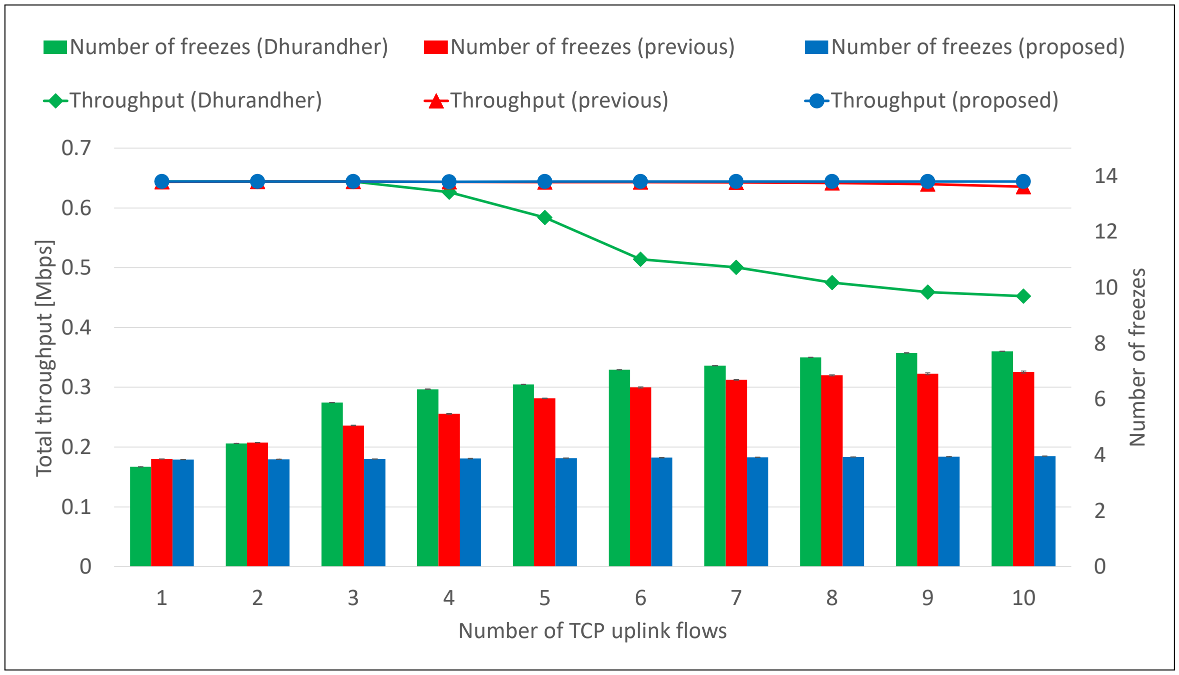


Figure 12: Average total UDP throughputs and those of number of freezes in Scenario 1

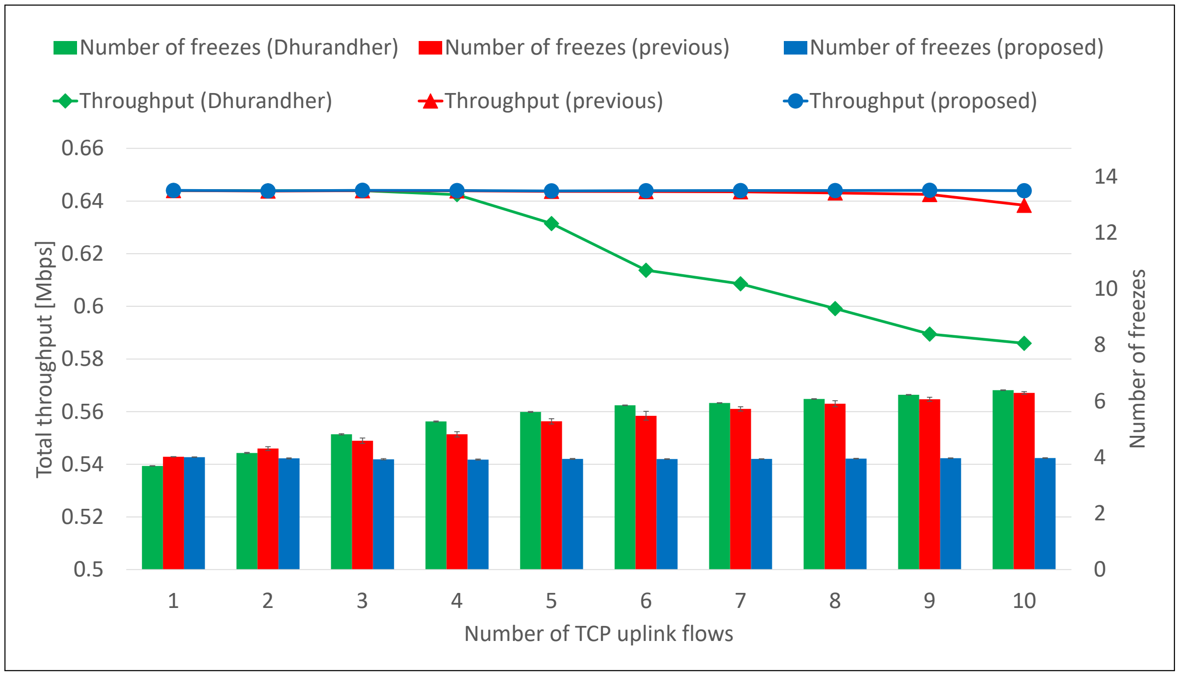


Figure 13: Average total UDP throughputs and those of number of freezes in Scenario 2

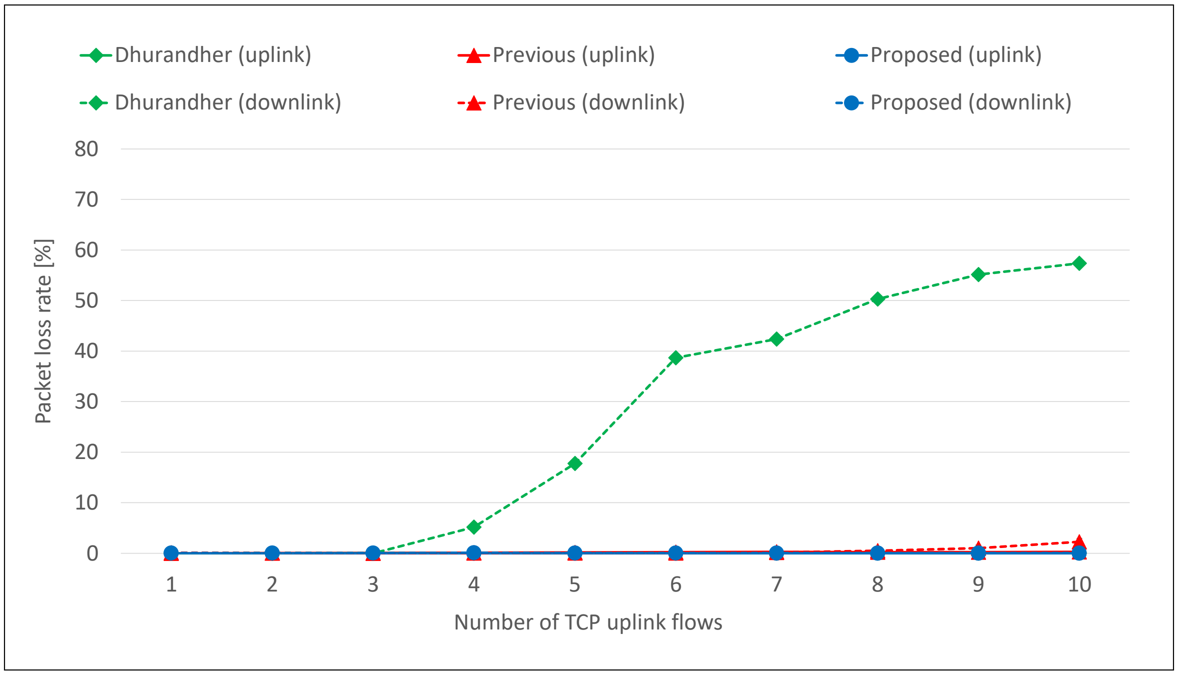


Figure 14: Average UDP packet loss rate in Scenario 1

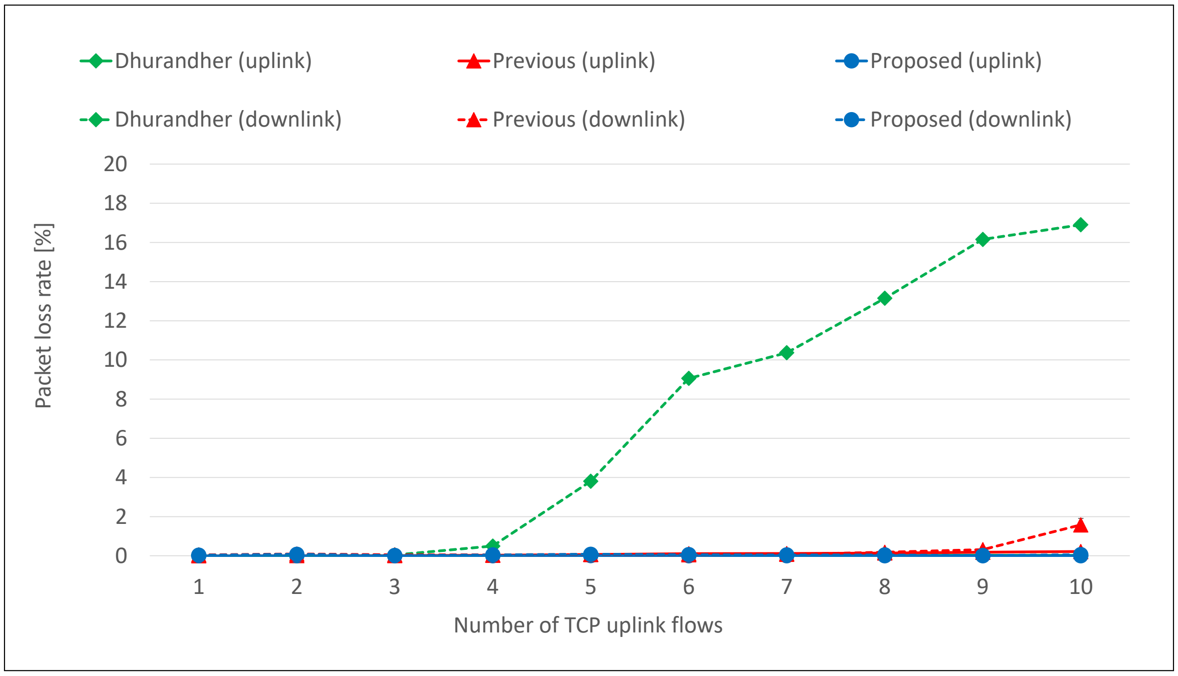


Figure 15: Average UDP packet loss rate in Scenario 2

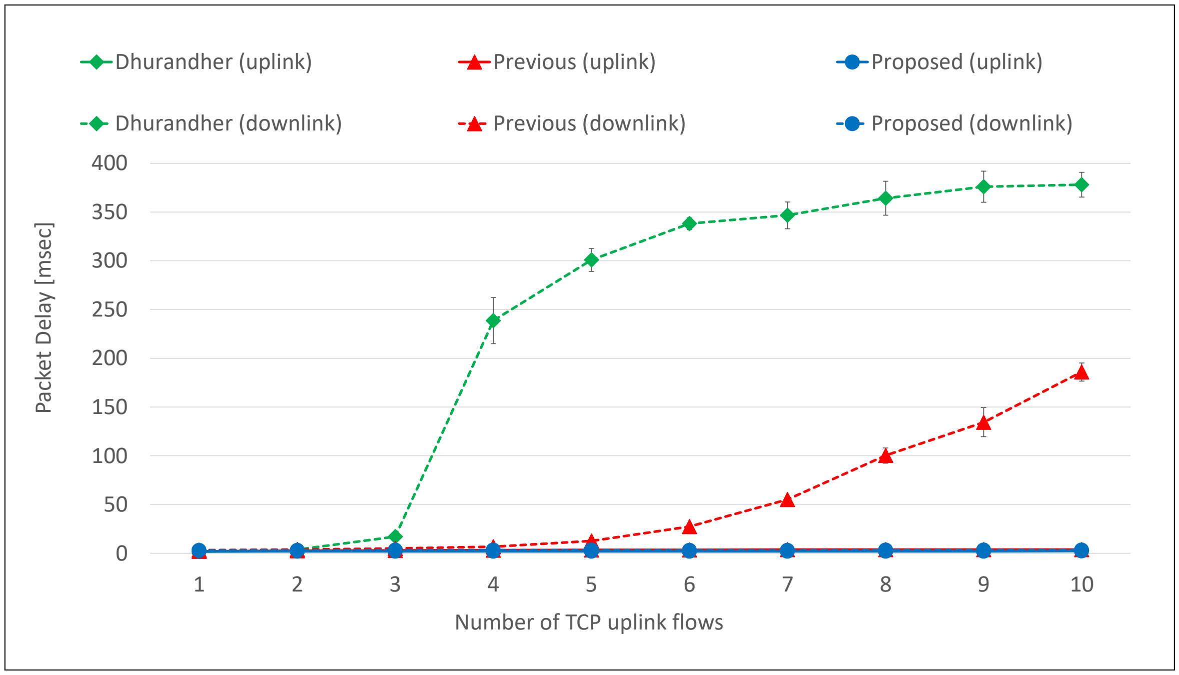


Figure 16: Average UDP packet delay in Scenario 1

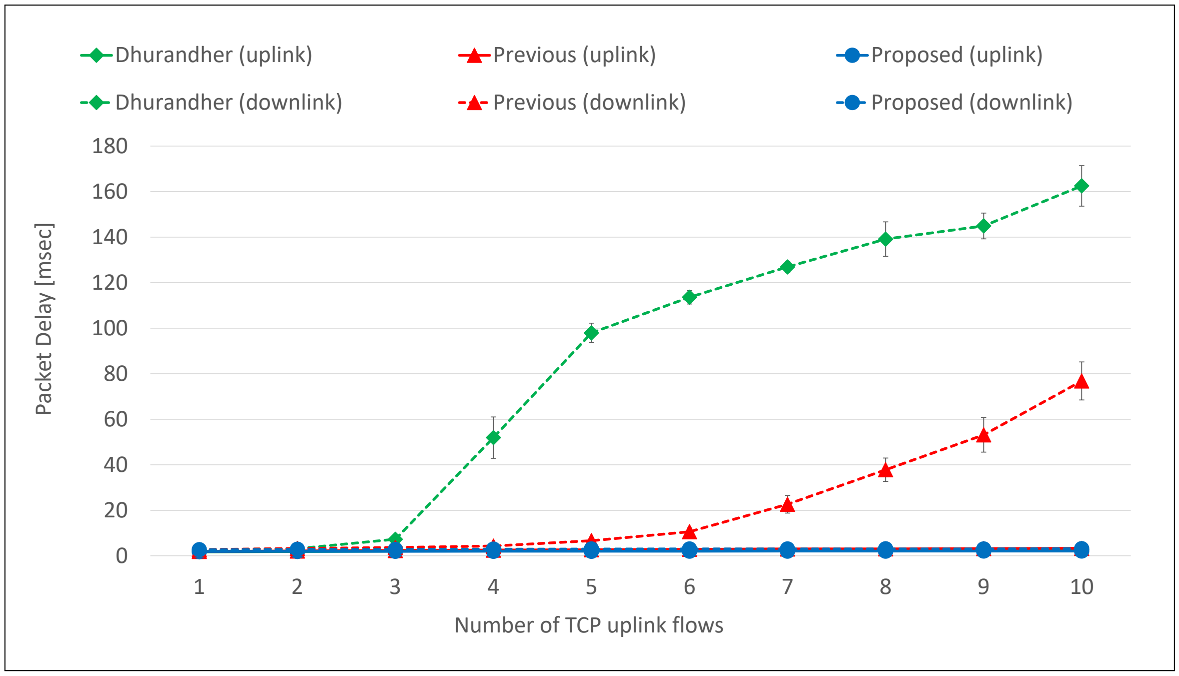


Figure 17: Average UDP packet delay in Scenario 2

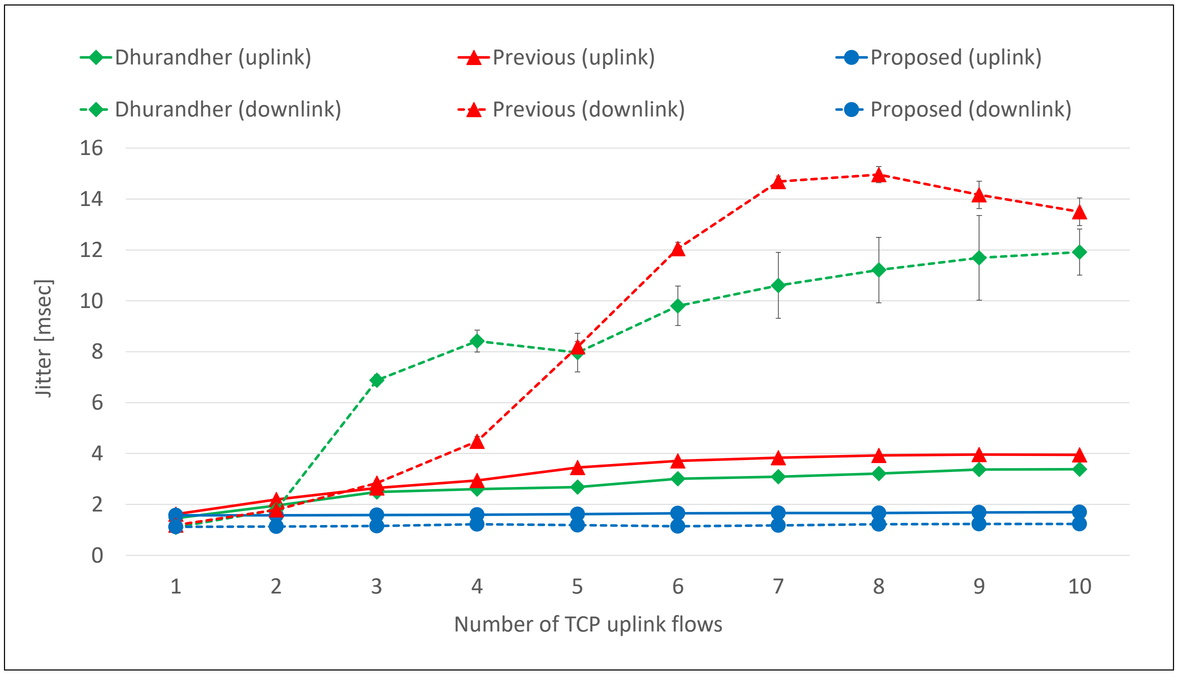


Figure 18: Average UDP packet jitter in Scenario 1

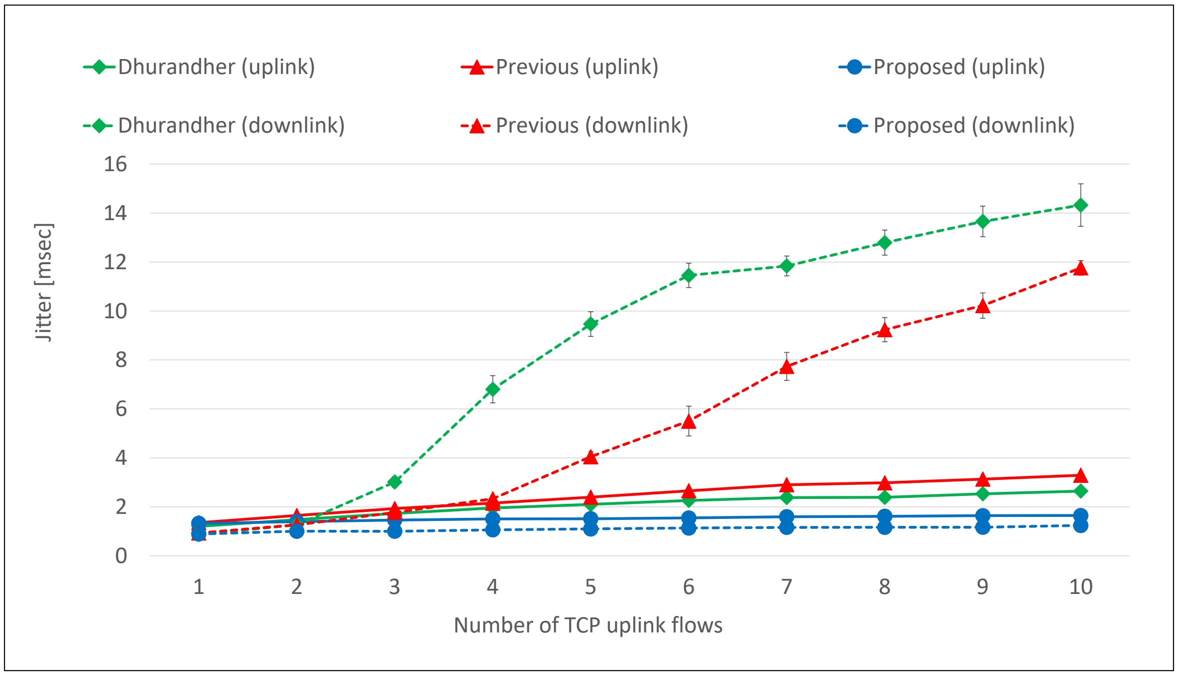


Figure 19: Average UDP packet jitter in Scenario 2

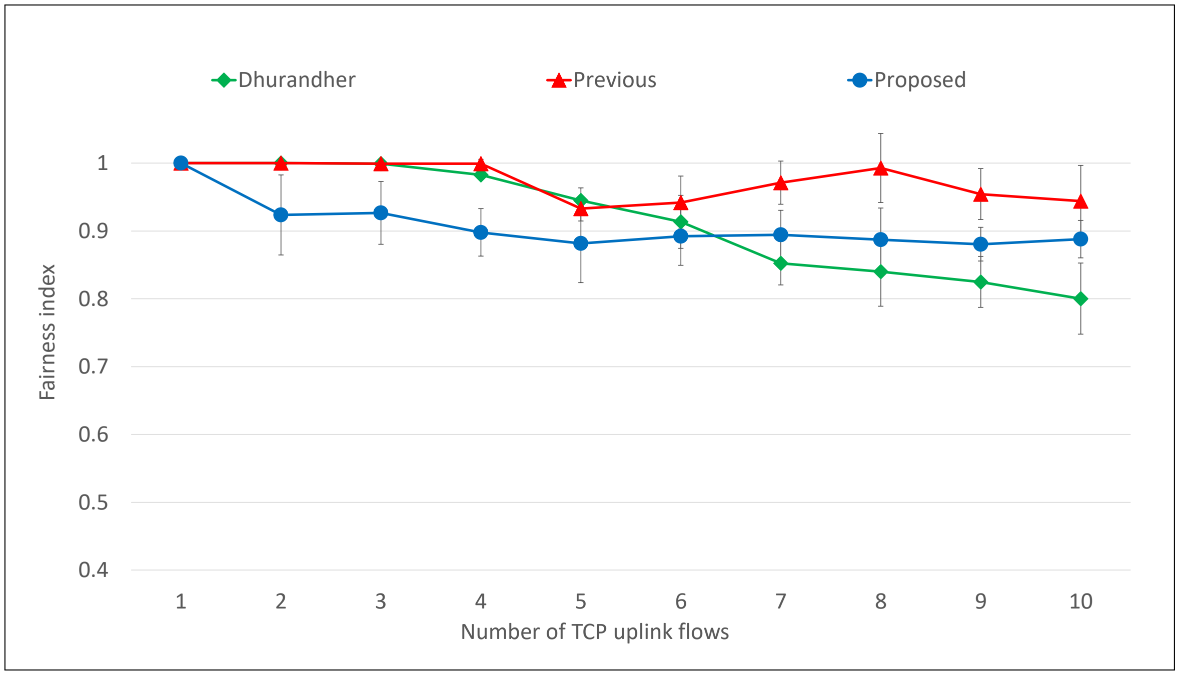


Figure 20: TCP fairness index in Scenario 1

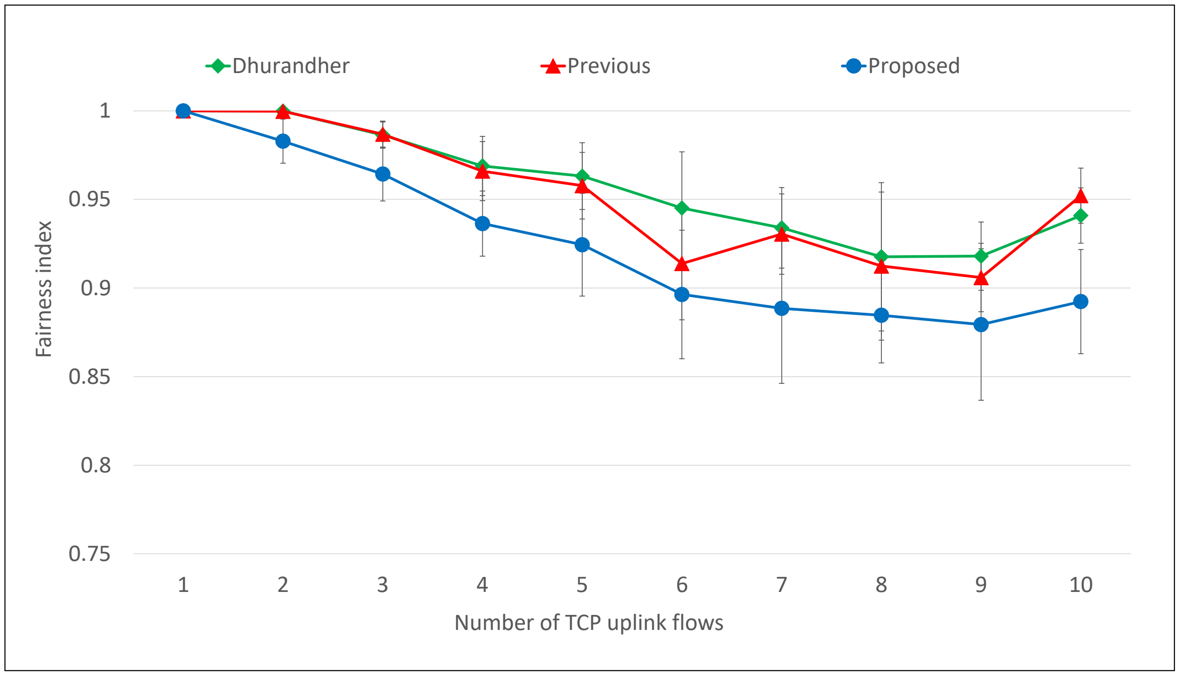


Figure 21: TCP fairness index in Scenario 2

5 Conclusions

This paper proposed a contention window control method using priority control based on the number of freezes for two priority levels. With this, network simulations in two scenarios demonstrated that the proposed method retained not only the average total throughput of low- and high-priority frames, but also the average packet delay and jitter of high-priority frames in both directions compared to the previous method and Dhurandher's method. However, the fairness-index for the low-priority flows of the proposed method was a little lower than those of the previous method. Although the higher throughput may be more important than the lower throughput with high fairness in practical cases, the proposed method should also be considered to improve the fairness of the low-priority. In the simulations, all nodes were assumed to support the same frame transmission control protocol. However, it should be verified in the case that the proposed method and other access control methods such as DCF and IEEE 802.11 Enhanced Distributed Channel Access (EDCA) are mixed to be used. Moreover, the proposed method should verify whether or not the method can provide the same performance even when the traffic volume changes continuously, as it does in practice. In the proposed method, only two priority levels can be supported, for voice flows and data flows. However, the proposed method should be expanded to more than two priority levels to support other kinds of flows such as video traffic.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number JP26280027.

References

[1] Teerapat Sanguankotchakorn, Aravinthan Gopalasingham, and Nobuhiko Sugino. Adaptive channel access mechanism for real time traffic over IEEE 802.11e Wi-Fi network. In *Proceedings of International Conference on Intelligent Systems, Modelling and Simulation*, pages 486–491, 2013.

[2] Katarzyna Kosek-Szott, Marek Natkaniec, and Lukasz Prasnal. IEEE 802.11aa intra-AC prioritization — a new method of increasing the granularity of traffic prioritization in WLANs. In *Proceedings of IEEE Symposium on Computers and Communication (ISCC)*, pages 1–6, 2014.

[3] Mehaseb Ahmed, Gamal Abdel Fadeel, and Ibrahim Ismail Ibrahim. Differentiation between different traffic categories using multi-level of priority in DCF-WLAN. In *Proceedings of International Conference on Telecommunications*, pages 263–268, 2010.

[4] Nao Emoto, Shigetomo Kimura, and Yoshihiko Ebihara. Evaluation of random discard method on wireless LAN APs for real-time communications (in Japanese). In *Proceedings of the 74th National Convention of IPSJ*, pages 507–508, 2012.

[5] Sanjay K. Dhurandher, Issac Woungang, Sahil Sharma, and Veeresh Goswami. A priority based differentiation for contention mechanism in legacy DCF method. In *Proceedings of International Conference on Advanced Information Networking and Applications Workshops*, pages 478–482, 2013.

[6] Shigetomo Kimura Tomoki Hanzawa. A minimum contention window control method for lowest priority based on collision history of wireless lan. *International Journal of Networking and Computing*, 7(2):295–317, 2017.

[7] C. Mala and B. Nithya. Dynamic sliding contention window adjustment in saturated wireless networks. In *Proceedings of International Conference on Network-Based Information Systems*, pages 186–193, 2014.

[8] Ikram Syed, Bosung Kim, Byeong hee Roh, and Il hyuk Oh. A novel contention window backoff algorithm for IEEE 802.11 wireless networks. In *Proceedings of IEEE/ACIS 14th International Conference on Computer and Information Science (ICIS)*, pages 71–75, 2015.

[9] Nao Emoto, Shigetomo Kimura, and Yoshihiko Ebihara. IEEE 802.11 contention window control method based on the number of wireless terminals to improve transmission probability for APs (in Japanese). In *Proceedings of the 72th National Convention of IPSJ*, pages 167–168, 2010.

[10] Hongxian Tian and Wei Yang. Study on QoS of multimedia traffics in MAC layer based on 802.11. In *Proceedings of International Conference on Information Science and Applications (ICISA)*, pages 1–6, 2012.

[11] Pedro Fortuna Gustavo Carneiro and Manuel Ricardo. Flowmonitor: a network monitoring framework for the network simulator 3 (ns-3). In *Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools*, volume 1, pages 1–10, 2009.

[12] Ragendra Jain, Dah ming W.Chiu, and William R. Hawe. *A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer System.* 1984.