



Article

Enhancing TCP Airtime Fairness through Precise Computation for Upload and Download Flows in WiFi Networks

Yuhao Chen  and Jinyao Yan * 

State Key Laboratory of Media Convergence and Communication, Communication University of China, Beijing 100024, China; chen_yuhao@cuc.edu.cn

* Correspondence: jyan@cuc.edu.cn

Abstract: Airtime fairness has emerged as a key approach to enhancing wireless throughput performance. However, existing research often overlooks the precise calculation of airtime, particularly in relation to TCP acknowledgments. This paper introduces a novel method, implemented on the access point side, for accurately calculating the airtime of TCP and UDP flows. Building on this, we propose a QoS-based scheduling algorithm designed to improve fairness between upload and download traffic. The effectiveness of the algorithm is validated through experiments that accurately measure both throughput and airtime for upload and download traffic.

Keywords: WiFi; airtime; fairness; QoS; scheduling; TCP; UDP; upload; download

1. Introduction

Wi-Fi networks have revolutionized wireless connectivity, becoming an essential part of modern life. The widespread adoption of Wi-Fi technology has brought countless conveniences, enabling people to access information, communicate, and share resources from virtually anywhere. Reports [1] from WiFiForward and Cisco predicted that by 2023, the majority of global internet traffic would be transmitted via Wi-Fi. Technologies like Wi-Fi, ZigBee, and RFID have not only significantly contributed to the economy but have also transformed how users access networks in their daily lives.

However, Wi-Fi operates as a half-duplex communication system, allowing data transmission in only one direction at a time. This half-duplex nature presents challenges in resource allocation and fairness. Most Wi-Fi networks use the distributed coordination function (DCF) to enable multiple devices (stations) to share the wireless channel. Before transmitting data, a device must first listen to the channel. If the channel is idle, the device attempts to send data. If the channel is busy, the device must wait for a random period before trying again.

While DCF ensures fair access to the wireless channel, it does not guarantee fair throughput sharing. To address the upload and download throughput imbalance caused by DCF, early Wi-Fi technologies introduced the concept of per-flow fairness. Per-flow fairness aims to ensure equitable network throughput for each flow by adjusting transmission opportunities (TXOP) or contention window (CW) settings. However, this approach does not account for the relationship between the number of flows and stations, resulting in stations with more flows receiving more throughput. To address this issue, researchers propose the concept of per-station fairness, which seeks to provide each station with fair throughput regardless of the number of flows.

In recent years, Wi-Fi technology has advanced significantly, resulting in increasingly faster wireless transmission rates. However, this progress has led to substantial rate differences between devices accessing the network. When all stations have the same throughput, slower nodes occupy the wireless channel for longer periods, which reduces overall network performance. To address this, researchers have proposed airtime fairness, which aims



Citation: Chen, Y.; Yan, J. Enhancing TCP Airtime Fairness through Precise Computation for Upload and Download Flows in WiFi Networks. *Telecom* **2024**, *5*, 992–1007. <https://doi.org/10.3390/telecom5040050>

Academic Editors: Peppino Fazio, Eirini Eleni Tsiropoulou, Carlos Tavares Calafate and Danilo Amendola

Received: 16 August 2024

Revised: 15 September 2024

Accepted: 22 September 2024

Published: 2 October 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

to distribute channel occupancy time fairly (system resource fairness) rather than just data transmission opportunities (individual throughput fairness). Airtime fairness mitigates the impact of slow devices on overall throughput and enhances network performance, and it has already been implemented in many commercial routers.

Despite these advancements, challenges with airtime fairness remain. Traditional airtime fairness considers only direct airtime during data transmission and overlooks additional airtime introduced by the DCF, known as indirect airtime. This oversight can prevent accurate allocation of channel resources by the access point (AP). Furthermore, fairness at the MAC layer does not necessarily translate to fairness at the transport layer [2–4]. For example, TCP flows, which provide reliable transmission through acknowledgment packets (ACKs), consume part of the wireless channel resources without contributing to data transmission, unlike UDP flows. This discrepancy necessitates the design of algorithms at the MAC layer to balance airtime between TCP and UDP flows, ensuring accurate resource allocation and better airtime fairness.

To improve wireless airtime fairness, we propose a quality of service (QoS) scheduling algorithm based at the access point. This algorithm assesses the wireless status of each flow to calculate its expected airtime, enabling the AP to select the appropriate data from the sending queue. The key contributions of this paper are as follows:

- We extend the concept of airtime fairness by integrating both direct airtime elements of data transmission and indirect elements, including those related to wireless protocols and TCP acknowledgments. We develop precise methods for calculating airtime for TCP and UDP flows, considering factors such as the state of the wireless layer, transport layer, and traffic direction. Furthermore, we introduce a QoS scheduling algorithm at the access point that ensures airtime fairness for both upload and download traffic.
- We propose an algorithm to achieve airtime fairness by integrating support logic and control logic. The control logic manages Wi-Fi QoS scheduling, determining the order of flows in the transmission queue. The support logic calculates each flow's quota, based on the accurate average airtime value of the flow. We demonstrate that the cooperation between support logic and control logic effectively ensures airtime fairness.
- We implement and test our algorithms in the ns-3 simulation [5] environment. The experimental results reveal that, without the QoS scheduling algorithm, expected fairness cannot be guaranteed. Traditional airtime fairness algorithms, due to inaccurate airtime calculations, perform poorly in fairness indices. In contrast, our QoS scheduling algorithm successfully achieves airtime fairness for both upload and download traffic, with results closely approximating the optimal solution.

This paper is organized as follows: Section 2 reviews related work and summarizes the development of Wi-Fi transmission fairness. In Section 3, we model and analyze the airtime components of TCP and UDP, deriving specific calculation methods. Section 4 presents the framework of our algorithm, detailing the queue scheduling and airtime quota calculation methods. In Section 5, we validate the algorithm's effectiveness through experiments. Finally, Section 6 concludes the paper.

2. Background

2.1. Wireless Network Data Transmission

Wi-Fi is a wireless local area network (WLAN) technology that enables multiple devices to transmit data without using physical cables. Wi-Fi networks operate in half-duplex mode, meaning they cannot send and receive data simultaneously. To manage access to the network by multiple devices, Wi-Fi employs the DCF protocol. The core of DCF is the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocol, which aims to ensure fairness and efficiency by preventing collisions and allocating transmission opportunities based on contention rules.

When a device wants to send data, it first listens to the wireless channel to check if it is idle. If another device is transmitting, the device will back off and wait for a random period. This backoff period consists of random slots and a fixed duration, after which the device will attempt to transmit again. If the channel is detected as idle, the device sends its

data frames. While the backoff mechanism reduces collisions, simultaneous transmissions and signal attenuation can still cause packet loss. To address this, the receiver sends an acknowledgment frame (ACK) upon successfully receiving a data frame. If the sender does not receive an ACK within a specific time, it will retransmit the data.

To efficiently manage the wireless channel, DCF defines several time intervals. The Short Inter-Frame Space (SIFS) is the waiting time for high-priority control frames, the Distributed Inter-Frame Space (DIFS) is for data frames and management frames, and the Extended Inter-Frame Space (EIFS) is for error frames. These intervals help prioritize different types of transmissions, ensuring orderly and reliable data transmission within the network.

2.2. Wireless Network Fairness

In wireless networks, fairness primarily concerns the equitable sharing and allocation of resources based on fair and reasonable strategies. Unfair resource allocation can lead to resource scarcity, waste, or redundancy. The definition of fairness in Wi-Fi is broad and can vary depending on specific requirements [6], including fair energy consumption [7], power control [8], flow scheduling [9], channel allocation [10], rate control [11], and routing selection [12]. In this paper, we focus on site-based fairness and airtime fairness in flow scheduling.

Wi-Fi networks can cover extensive indoor areas. To ensure stable connectivity for devices at various locations and to reduce wireless packet loss due to signal attenuation, devices closer to the access point typically maintain higher wireless rates, while devices farther from the AP maintain lower wireless rates. We can implement a site-based fairness algorithm at the AP. For example, consider two devices, A and B, connected to the AP, where device A is farther from the AP, and device B is closer. The channel occupancy process is illustrated in Figure 1.

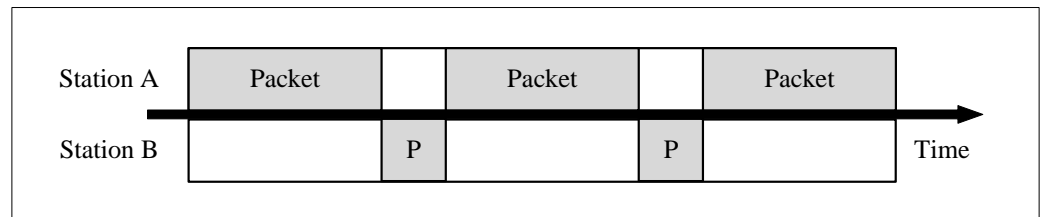


Figure 1. Per-station fairness in Wi-Fi transmission, where P refers to packet.

The per-station fairness algorithm ensures that devices A and B experience equitable throughput when transmitting data by alternately sending frames of the same size to both devices. However, because device A is farther from the AP and has a slower transmission speed, it requires more channel time to transmit the same size frame. Consequently, even though device B has a higher transmission rate, its actual transport layer rate is limited by device A's slower rate. This means that as the number of connected devices increases, the overall performance of the wireless network is determined by the rate of the slowest device.

To address this issue, researchers proposed airtime fairness, also known as time-based fairness. Airtime fairness is widely recognized as a solution to the performance anomaly in Wi-Fi networks with varying speeds [13] and has been deployed in many router devices [14–16]. Its goal is to provide each user with an equal amount of airtime. In the scenario described, when the AP's scheduling algorithm is replaced with airtime fairness, the channel occupancy process for the two devices is illustrated in Figure 2.

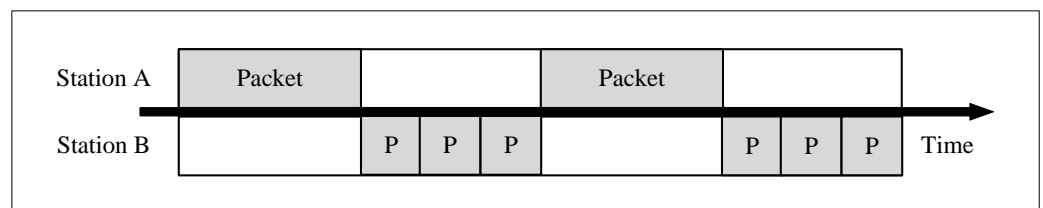


Figure 2. Airtime fairness in Wi-Fi transmission, where P refers to packet.

Airtime fairness shifts the focus from providing each station with equal throughput to allocating transmission time equally among devices. This ensures that all devices have equal access to the wireless channel, preventing slower devices from reducing overall network performance. For example, a node with a faster wireless rate, such as node B, will receive more transmission opportunities from the AP. This allows node B to utilize its higher rate advantage, transmitting more data and thus increasing the overall network throughput. Conversely, a node with a slower transmission rate, such as node A, will have reduced transmission opportunities to prevent it from consuming excessive airtime and affecting the performance of other devices.

In IEEE 802.11ac, transmission rates can vary significantly between nodes. For instance, with a guard interval (GI) of 400 ns, an mcs0 node has a wireless rate of 32.5 Mbps, while an mcs9 node has a rate of 433.3 Mbps. When transmitting the same amount of data, the airtime ratio between these nodes is 13.3 to 1. Under traditional fairness, both nodes would transmit the same amount of data, resulting in an average access point bandwidth of approximately 60 Mbps. However, with airtime fairness, both nodes occupy the same amount of airtime, leading to an average access point bandwidth of about 233 Mbps. This represents a 388% increase in total throughput compared to traditional fairness.

3. How to Calculate Accurate Airtime at the AP Side

Paper [17] introduces the concept of responsible airtime fairness, suggesting that airtime fairness should encompass not only data transmission time but also all related overhead. However, paper [17] does not precisely determine the responsible airtime for each data frame. Its scheduling algorithm calculates based on the throughput of each node, while a more accurate approach should consider the rate and wireless transmission logic of each node. Additionally, paper [17] does not address how to handle the airtime issues caused by TCP ACKs.

This section first analyzes the airtime composition of UDP and TCP. UDP's airtime is simpler as it does not include transport layer acknowledgment packets. It then examines the wireless queuing issues caused by TCP ACKs in TCP upload data and provides solutions. For ease of derivation and explanation, the calculation analysis in this paper excludes the RTS and CTS mechanisms of DCF. Including RTS and CTS would only increase the complexity of the derivation without affecting the paper's conclusions.

3.1. Overview

Traditional airtime fairness only considers the direct airtime for transmitting data. In this paper, we firstly extend the definition of airtime fairness and provide a more precise analysis, as illustrated in Figure 3. Our extended definition includes both the direct airtime for transmitting effective data and the indirect airtime associated with the wireless DCF function.

Direct airtime, as in traditional methods, represents the wireless time spent transmitting effective data. Indirect airtime, on the other hand, encompasses fixed-length segments such as SIFS, Slot, and wireless ACK, as well as backoff time, which has a random and uncertain length. Although backoff time is somewhat random, it helps prevent wireless interference and packet loss caused by multiple stations transmitting data frames simultaneously. By using AP-side algorithms, we can derive the expected value of the backoff time for a single data transmission from a theoretical perspective, allowing us to accurately calculate the airtime for each transmission.

Furthermore, our concept of airtime fairness considers the type of flow, whether TCP or UDP. TCP and UDP are both transport layer protocols, with TCP focusing on reliability and orderliness of data transmission and UDP emphasizing transmission efficiency and lower latency. From a wireless airtime fairness perspective, TCP flows include an additional component: the airtime required for transmitting TCP ACKs over the wireless channel. TCP ACKs ensure reliable transmission of TCP flows but do not carry effective transmission content. Therefore, the wireless airtime of TCP flows should also include the airtime for TCP ACKs, which is part of the indirect airtime.

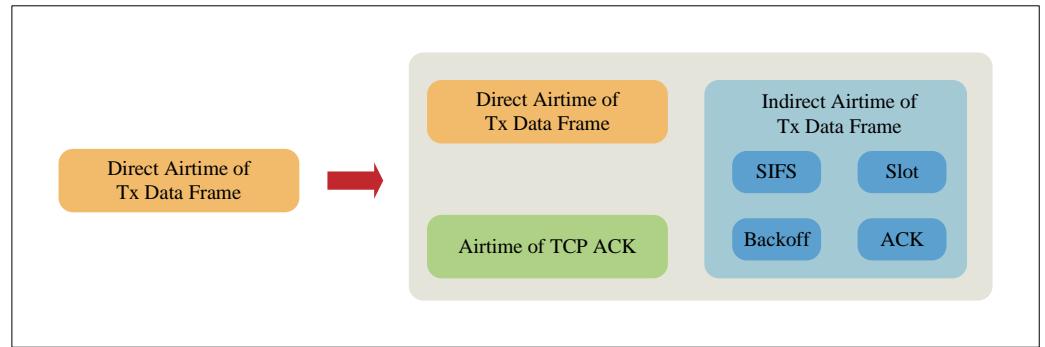


Figure 3. Overview of our airtime definition.

This paper analyzes the composition of airtime and models it in Sections 3 and 4. The symbols and their meanings involved are shown in Table 1. Additionally, some parameters (e.g., SIFS, Slot, etc.) used in the following analysis are related to the network scenarios. Similar to the 802.11 MCS rate table, AP can reference a preset data table to find the specific parameter values. In different network conditions, the calculation method of airtime in our algorithm remains the same but with different values of parameters. This allows our approach to maintain consistent performance across different network scenarios.

Table 1. Notations in airtime fairness calculation.

Notation	Description
F	The set of flows
T_{udp}	One segment airtime of a UDP flow
T_{tcp}	One segment airtime of a TCP flow
t_{dir}	The direction airtime of one frame
t_{ind}	The indirection airtime of one frame
s	Size of frame
r	Wireless rate of frame
W_i	Contention window of the i -th retransmission
R	Maximum number of wireless retransmissions
p	Packet loss probability obtained from AP statistics
d	Efficiency of TCP Reply ACK

3.2. Analysis of UDP Flows

UDP is suitable for applications requiring high real-time performance with low data integrity demands. It does not guarantee reliable transmission and lacks an acknowledgment mechanism. UDP streams involve only unidirectional transmission, with wireless latency comprising both the direct transmission delay of data packets and the indirect delay occupying the wireless link.

Figure 4 illustrates the process of an AP sending UDP data to a STA. The “Data” section in Figure 4 represents the direct time required for transmitting actual data over the wireless channel, while the other sections represent the indirect time caused by the wireless DCF design. This indirect time overhead does not involve the transmission of useful data but ensures the orderly transmission of wireless data and reduces collisions.

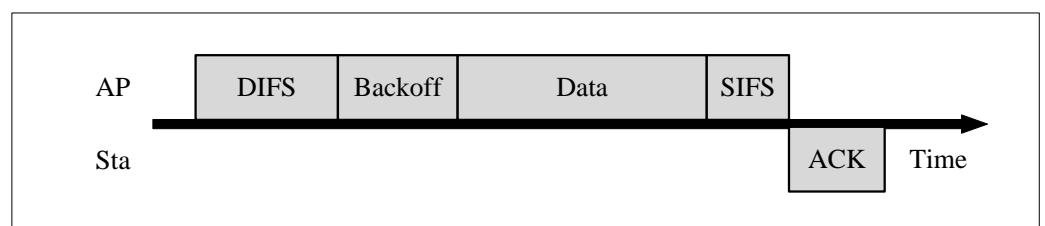


Figure 4. Airtime of a single transmission in Wi-Fi.

As shown in Figure 4, the airtime T_{udp} for transmitting a UDP stream from an AP to an STA is equal to the airtime T_{data} of the data frame, which can be expressed as the sum of the direct transmission time t_{dir} and the indirect transmission time t_{ind} :

$$T_{udp} = T_{data} = t_{dir} + t_{ind} \quad (1)$$

The direct transmission time of a wireless data frame is calculated as follows, in seconds:

$$T_{dir} = \frac{s}{r} \quad (2)$$

where s represents the frame size in bits and r represents the wireless rate in bps. The indirect transmission time t_{ind} is the delay required by the DCF to avoid collisions. If the RTS and CTS mechanisms are not used, t_{ind} can be calculated as follows:

$$t_{ind} = DIFS + Backoff + SIFS + MAC_ACK \quad (3)$$

The formula includes both fixed and variable time components. Let us first introduce the fixed-length components, which include $DIFS$, $SIFS$, and MAC_ACK . Different IEEE 802.11 standards define different SIFS times and Slot times. The AP can obtain parameters such as modulation type, coding scheme, and data rate during operation. Thus, it can retrieve the corresponding SIFS and Slot values from the configuration file. In the formula, the length of DIFS is equal to SIFS plus twice the Slot time. Additionally, the ACK in the wireless MAC layer is usually transmitted at the lowest rate. By using the length of the wireless ACK frame and the following formula, the fixed airtime required for wireless ACK transmission can also be calculated.

To prevent data packet collisions when multiple devices send data simultaneously, wireless communication protocols introduce the backoff mechanism and backoff algorithm, with the most common being the Binary Exponential Backoff (BEB) algorithm. When sending data, the device will select a random backoff time, commonly referred to as the contention window. The random range of this window dynamically adjusts based on the number of collisions, doubling the range with each collision.

There has been modeling research on the backoff delay of wireless nodes [18–20]. The research indicates that the expected backoff time is related to the number of streams N , the minimum contention window size CW_{min} , the maximum contention window size CW_{max} , and the maximum number of wireless retransmissions (R).

Specifically, the expected backoff time in this scenario is calculated as follows. Let i represent the number of retransmissions and W_i represent the range of the CW window for the i -th transmission attempt. The backoff time will be a random integer chosen from the interval $[0, W_i - 1]$. Here, $W_0 = CW_{min}$, and $W_i = \min(2 \cdot W_{i-1}, CW_{max})$. Let p represent the average packet loss probability, which can be obtained through AP-side statistics. The backoff process follows a general Markov process, and its expected value is:

$$Backoff = \sum_{i=0}^R \frac{W_i - 1}{2} \times (1 - p) \times p^i \quad (4)$$

Therefore, the indirect transmission time of a UDP stream can be expressed as follows:

$$t_{ind} = Backoff + 2 \times SIFS + 2 \times Slot + \frac{s}{r} \quad (5)$$

where s is the frame size of the wireless ACK, and r is the minimum data frame rate supported by the wireless network. In summary, the AP side can calculate the exact airtime required for a single transmission of a UDP stream based on the state of each stream.

3.3. Analysis of TCP Flows

TCP ensures reliable data transmission through sequence numbers, acknowledgments, and retransmission mechanisms. The difference between TCP and UDP in terms of wireless airtime is that TCP includes an acknowledgment mechanism, which allows the receiver to inform the sender of the sequence number of the next data segment expected. Therefore, the airtime for a TCP stream includes both the airtime for TCP data packets and the airtime for TCP ACKs:

$$T_{tcp} = T_{data} + T_{ack} \quad (6)$$

When considered separately, the transmission logic for TCP data packets and TCP ACKs over wireless is the same as for UDP. They both include direct airtime and indirect airtime. Thus, using the method for calculating UDP airtime described in Section 3.2, combined with the frame sizes of TCP data and TCP ACK, their respective airtime can be calculated.

TCP can adjust the sending frequency of TCP ACKs through parameters. Typically, after successfully receiving a data segment, the receiver will combine acknowledgments for multiple data segments and then reply with one ACK message. This reduces the number of acknowledgment messages and improves network efficiency. In the short term, the position of the Sta is relatively fixed, so it can be assumed that the wireless transmission rate for both TCP data packets and ACKs is equal. If d (Delayed Acknowledgment) represents the reply frequency of TCP ACKs, that is, d (default is two) TCP data segments for one ACK, then the single airtime for a TCP stream can be expressed as:

$$T_{tcp} = \frac{s_{data} + s_{ack}/d}{r} + \left(1 + \frac{1}{d}\right) \times t_{ind} \quad (7)$$

where s_{data} and s_{ack} represent the frame sizes of the TCP data packet and ACK, respectively, r represents the wireless rate of the stream, and t_{ind} is calculated using Formula (5). So far, we have been able to calculate the total airtime for both TCP and UDP. This paper will address the following question in Section 4.2: How can we design an appropriate queue scheduling algorithm that, in combination with TCP ACKs, targets both upload and download traffic separately?

4. Algorithm Design

This section introduces the algorithm proposed in this paper. First, we outline the basic framework of the algorithm in Section 4.1. Next, we describe the method for calculating airtime resources in Section 4.2. Finally, we detail the AP-based queue scheduling algorithm in Section 4.3.

4.1. Workflow

The algorithm framework consists of two main components, support logic and control logic, which work together to implement a queue scheduling algorithm designed to achieve airtime fairness, as illustrated in Figure 5.

The support logic does not directly handle data frame transmissions; instead, it provides the necessary calculations and services for the algorithm's operation. Specifically, it determines the airtime resources (discussed in Sections 3.2 and 3.3) and quotas (detailed in Section 4.2) for each flow. The airtime resources for each flow depend solely on its own state and must be recalculated whenever the wireless rate changes. Conversely, the quotas that ensure fairness are influenced by the states of all flows. Thus, when a flow joins, leaves the network, or experiences a rate change, its airtime resources must be recalculated based on the current network state, while quotas are updated to reflect the conditions of all flows accessing the AP.

The control logic is responsible for selecting specific frames from the sending queue for transmission. When a wireless frame is ready to be sent, its type is identified by its header field. The proposed queue scheduling algorithm focuses on TCP and UDP data frames and does not affect wireless control frames. The algorithm regulates the number of data packets sent by each flow according to the quotas calculated by the support logic, using a DRR-based

approach. Ultimately, the product of the total data transmitted by each flow and its respective airtime resource results in their total airtime allocation, thereby improving airtime fairness.

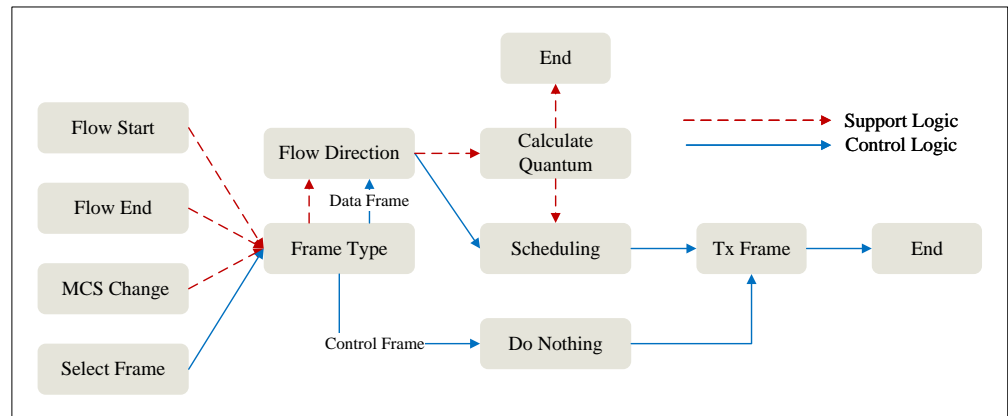


Figure 5. Workflow of our algorithm.

4.2. Queue Scheduling

Most airtime fairness scheduling implementations are based on the Deficit Round-Robin (DRR) algorithm [21]. In this approach, each flow is assigned a queue and a service quantum. If a queue is not selected in the current round, its remaining quantum is carried over to the next round. This process tracks deficits, ensuring that any unfairness from one round is compensated in subsequent rounds. While [21] effectively addresses unfairness caused by varying packet sizes among flows, our approach incorporates packet size into the quantum calculation from the outset, preemptively resolving this issue.

Paper [22] improved the airtime Deficit Round Robin algorithm to prevent nodes with high packet loss rates from monopolizing the wireless channel, thereby optimizing system performance. However, this method requires real-time estimation of airtime during decision-making, leading to higher overhead. In contrast, our method calculates airtime only when a flow changes its Modulation and Coding Scheme (MCS). During decision-making, it relies solely on this pre-calculated airtime, thereby reducing overhead.

Figure 6 illustrates the logic of our proposed queue scheduling algorithm. This algorithm is a QoS scheduling method deployed on top of an active queue management algorithm, responsible only for selecting frames from the queue. Unlike the approach in [21], our algorithm requires just a single queue, reducing both complexity and deployment overhead.

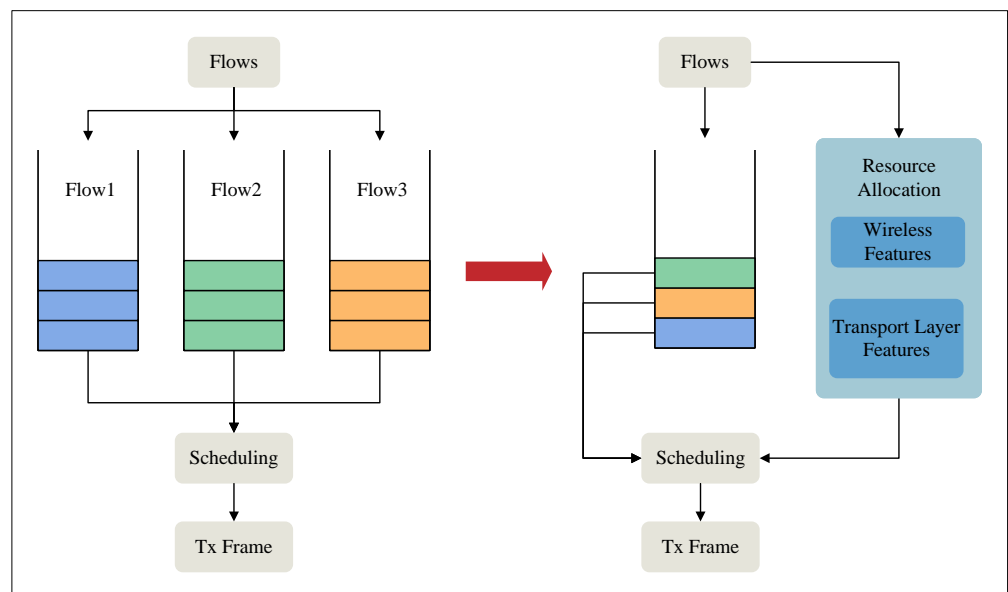


Figure 6. Our queue scheduling algorithm.

The queue scheduling algorithm proposed in this section relies on the quotas determined by the resource allocation algorithm (discussed in Section 4.3). Adhering to the logic of the DRR algorithm, it selects the most appropriate flow at any given time and then transmits the earliest frame from that flow. The detailed steps of this queue scheduling algorithm are presented in Algorithm 1.

Algorithm 1 Queue scheduling algorithm based on QoS.

Input: *deficit_counter, quantum*
Output: *selected flow*

```

1: Init: deficit_counter[] = quantum[];
2: function peek_flow()
3:   min_dc = deficit_counter[0];
4:   selected = 0;
5:   for (i = 0; i < N; i++) do
6:     if (deficit_counter[i] > min_dc) then
7:       min_dc = deficit_counter[i];
8:       selected = i;
9:     end if
10:  end for
11:  for (i = 0; i < N; i++) do
12:    deficit_counter[i] -= min_dc;
13:  end for
14:  deficit_counter[selected] = quantum[selected];
15:  return selected;
16: end function

```

To achieve airtime fairness, we assign different transmission opportunities (quantum) to each STA node. This quantum is determined by the node's rate, transport layer type, transmission direction, and other characteristics, as calculated by the resource allocation module. When selecting a flow for transmission, we identify the minimum value of the 'deficit_counter' array ('min_dc') and its corresponding index. The flow associated with that index is then selected, and its 'deficit_counter' is reset based on its airtime quantum. For the flows not selected, we subtract 'min_dc' from their 'deficit_counter' values, increasing their chances of being selected in future transmissions, thereby achieving airtime fairness, as illustrated in Figure 2.

In this queue scheduling algorithm, flows with larger quanta have fewer opportunities to transmit wireless data frames, resulting in a smaller number of sent frames (n). In other words, the number of packets sent (n) is inversely proportional to the assigned quantum.

$$\frac{n_i}{n_j} = \frac{\text{quantum}_j}{\text{quantum}_i} \quad (8)$$

4.3. Airtime Quantum Calculation

Based on the analysis in Sections 3.2 and 3.3, the airtime for TCP and UDP flows is calculated using flow type, packet size, wireless rate, and other relevant information. However, our algorithm differentiates between upload and download TCP flows in the airtime calculation method. This difference arises from the relationship between the number of TCP data packets and ACKs, as well as the position of queuing and scheduling algorithms in end-to-end transmission. As discussed in Section 3.3, TCP flows employ delayed acknowledgment, where the receiver sends one ACK for every d TCP data packets received.

For download TCP data, the access point sends TCP data, and the stations returns TCP ACKs. When sending data, the AP pre-calculates the airtime required for the return of TCP ACKs and uses this as a basis for airtime fairness scheduling, ensuring fairness for download TCP data, as illustrated in Figure 7. The airtime for transmitting a data frame in

this context includes the airtime for one data packet plus the airtime for $1/d$ TCP ACKs, calculated according to the method and formula in Equation (7).

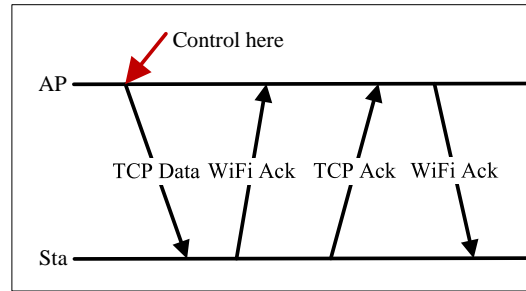


Figure 7. Control logic for airtime fairness in download scenario.

For upload TCP data, the stations transmit TCP data, and the access point returns TCP ACKs. By the time the AP sends these ACKs, the TCP data transmission has already been completed. At this stage, the AP adjusts the sending order of TCP ACKs for each TCP flow using an airtime fairness scheduling algorithm to ensure fairness for upload TCP data, as illustrated in Figure 8. The airtime for transmitting TCP ACKs includes the airtime for d data packets and the airtime for 1 TCP ACK, calculated as follows:

$$T_{tcp} = \frac{d \times s_{data} + s_{ack}}{r} + (d + 1) \times t_{ind} \tag{9}$$

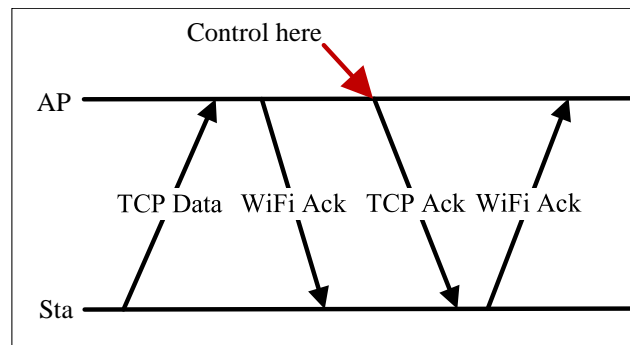


Figure 8. Control logic for airtime fairness in upload scenario.

In conclusion, we can calculate the single transmission airtime for node i , denoted as t_i . Next, we analyze the relationship between t_i , the number of transmissions n_i , the quota $quantum_i$, and the total airtime A_i . Our goal is to determine n_i based on t_i for each node, ensuring that the total airtime for each node is equal, i.e.,

$$\forall (i, j) \in F^2, A_i = A_j \tag{10}$$

Given that the total airtime of different nodes equals the product of the number of frames sent and the average airtime per frame, we have:

$$A_i = t_i \times n_i \tag{11}$$

To achieve airtime fairness between different nodes i and j , the following condition must be met:

$$\frac{n_i}{n_j} = \frac{t_j}{t_i} \tag{12}$$

By combining Equation (12) with Equation (8) from the queue scheduling algorithm in Section 4.2, we can establish the relationship between the airtime of different nodes and their allocated resource quotas:

$$\frac{quantum_i}{quantum_j} = \frac{t_i}{t_j} \tag{13}$$

This analysis demonstrates that by accurately calculating the average airtime of each flow's data frames, we can allocate the opportunity to access the wireless channel more effectively, thereby achieving more precise airtime fairness and improving overall throughput. The resource allocation algorithm in this paper is provided in Algorithm 2.

Algorithm 2 Quantum calculation algorithm of airtime.

Input: *flow features, wifi features*

Output: *quantum*

```

1: function get_quantum(){
2:   for (i = 0; i < N; i++) do
3:     backoff = 0;
4:     for (j = 0; j < R[i]; j++) do
5:       backoff += (w[i] - 1)/2 × (1 - p) × pow(p,j);
6:     end for
7:     ind = backoff + 2 × SIFS + 2 × Slot + s_w_ack/r_min;
8:     if 'UDP' == get_type(flow[i]) then
9:       airtime[i] = ind + s[i]/r[i];
10:    end if
11:    else if 'TCP' == get_type(flow[i]) then
12:      if 'UPLOAD' == get_direction(flow[i]) then
13:        airtime[i] = (d[i] × s[i] + s_tcp_ack[i])/r[i] + (1 + d[i]) × ind
14:      end if
15:      else
16:        airtime[i] = (s[i] + s_tcp_ack[i]/d[i])/r[i] + (1 + 1/d[i]) × ind
17:      end else
18:    end else if
19:    return airtime;
20:  end function

```

5. Experiment

5.1. Experimental Settings

Experimental Environment. In this study, we deployed and tested the algorithm using the ns-3 simulator, version 3.35. The experimental setup, illustrated in Figure 9, features a combination of wired and wireless links, with specific links configured to have a bandwidth of 100 Mbps and a latency of 10 ms. The wireless network operates on the 802.11ac protocol, the most advanced version supported by ns-3. In this configuration, the server functions as the sender, while the stations serve as the receivers. For wireless loss, we used the LogDistancePropagationLossModel, and for propagation delay, the ConstantSpeedPropagationDelayModel. The IdealWifiManager algorithm was selected for wireless rate control.

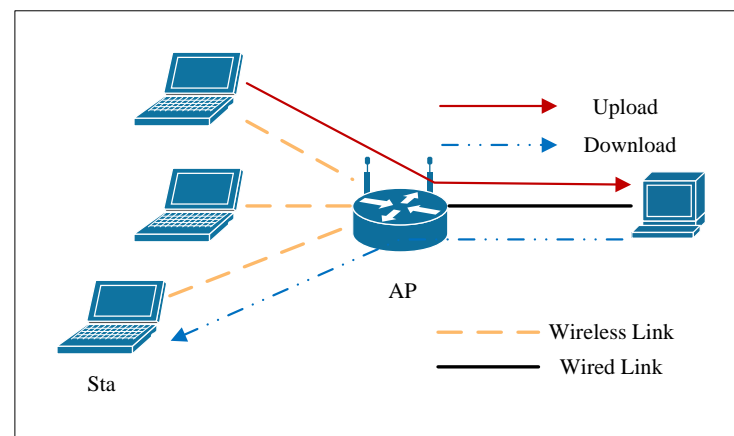


Figure 9. Wireless network topology in our ns-3 testbed.

Each server and station pair maintains a TCP stream using the Reno [23] congestion control algorithm, with the application layer generating continuous data. In the experiment, the distances between the three Stas and the access point vary, with Sta1 being the closest, followed by Sta2, and Sta3 being the farthest. This distance variation leads the WiFi rate selection algorithm to assign different modulation and coding schemes (MCSs) to each Sta: MCS8 for Sta1, MCS6 for Sta2, and MCS4 for Sta3.

Airtime statistics in ns-3. We record the timestamps of each data frame transmitted by the AP and Sta in the wireless network using the ns-3 API, correlating TCP data packets and acknowledgments by their sequence numbers. Figure 10 illustrates the airtime calculation method for a TCP stream, which includes $t1 + t2$. Since UDP streams lack transport layer acknowledgment packets, airtime for UDP is calculated using only $t1$.

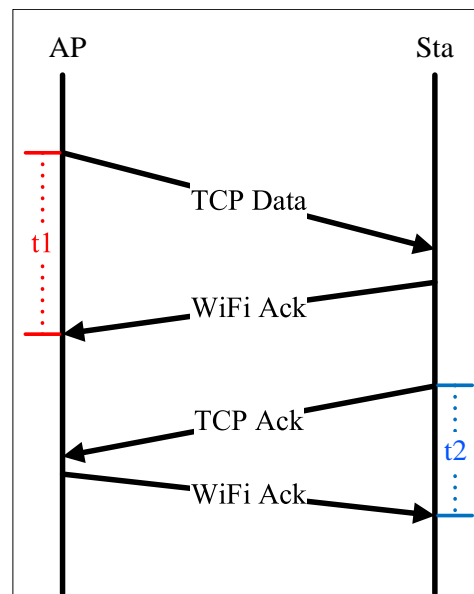


Figure 10. Airtime statistics method in ns-3.

Evaluation Metrics. We collect two key metrics: the algorithm's throughput and airtime fairness, measured using Jain's fairness index. The calculation methods are outlined below, with higher values of F indicating better fairness:

$$F(x) = \frac{(\sum_{i=1}^n x_i)^2}{n \times \sum_{i=1}^n x_i^2} \quad (14)$$

5.2. Evaluation

The airtime fairness algorithm proposed in this paper is a QoS scheduling algorithm designed specifically for Wi-Fi. Unlike general Wi-Fi queue scheduling algorithms, QoS scheduling and queue scheduling serve distinct purposes but work together to enhance the performance and user experience of wireless networks. The primary goal of a Wi-Fi QoS scheduling algorithm is to manage and optimize the transmission of various types of traffic within the wireless network to meet the specific quality of service requirements of different applications.

A common Wi-Fi QoS mechanism is Enhanced Distributed Channel Access (EDCA), defined in the IEEE 802.11 standard. EDCA classifies traffic into four types: voice, video, background, and best effort. Each has different priorities and parameters to ensure better service for high-priority traffic. In contrast, Wi-Fi queue scheduling algorithms manage packet queues on access points or terminal devices, handling enqueue and dequeue operations, transmission resource allocation, and traffic control. Examples of common queue scheduling algorithms include Drop-tail [24], RED [25], and CoDel [26], which determine the order and timing of packet transmission. In this study, the classic Drop-tail queue was utilized.

Download Scenario. We begin by comparing the performance of three QoS algorithms in a download scenario: the default QoS algorithm, general airtime fairness, and the airtime fairness algorithm proposed in this paper. Throughput results are tested under varying TCP ACK delay conditions, as shown in Table 2.

Table 2. Throughput comparison of QoS algorithms in DL scenario.

DelACK	QoS Algorithm	Sta1	Sta2	Sta3	Total Throughput
1	Default	6.73	6.75	6.79	20.27
1	General Airtime Fairness	14.07	4.56	3.06	21.69
1	Our Airtime Fairness	8.39	7.42	5.33	21.14
2	Default	8.38	8.38	8.37	25.14
2	General Airtime Fairness	17.36	5.73	3.86	26.94
2	Our Airtime Fairness	10.38	9.29	6.36	26.03

The experimental results reveal that the default QoS algorithm in this scenario aligns with throughput fairness, as the throughput for the three stations is nearly identical. In contrast, general airtime fairness tends to favor stations with higher wireless rates, providing them with more transmission opportunities. This approach results in higher effective throughput for these stations, thereby increasing the overall throughput of the wireless link.

Our proposed airtime fairness algorithm, however, produces more balanced throughput among the stations by adjusting airtime calculations differently for each station. As a result, while the total throughput achieved by our algorithm exceeds that of the default QoS algorithm, it falls short of the throughput provided by general airtime fairness. We would like to further explain that better airtime fairness does not equate to higher total throughput. For instance, if all wireless channel resources are allocated to the fastest STA, the highest total throughput would be achieved. However, this is clearly not a reasonable approach.

Consistent results across experiments with different TCP delayed ACK parameters demonstrate that our QoS algorithm effectively adjusts the stations' access opportunities to the wireless channel based on their airtime, thereby achieving airtime fairness. The specific airtime for each STA in this experiment is detailed in Table 3.

Table 3. Airtime comparison of QoS algorithms in DL scenario.

DelACK	QoS Algorithm	Sta1	Sta2	Sta3	Fairness Index
1	Default	0.16	0.17	0.24	0.9658
1	General Airtime Fairness	0.33	0.11	0.1	0.7589
1	Our Airtime Fairness	0.2	0.19	0.19	0.9992
2	Default	0.15	0.17	0.24	0.9569
2	General Airtime Fairness	0.12	0.11	0.11	0.7788
2	Our Airtime Fairness	0.19	0.18	0.18	0.9998

The experimental results indicate that the airtime fairness index of the default QoS algorithm is quite high, even significantly surpassing that of the general airtime fairness algorithm. Moreover, the default QoS algorithm can be viewed as throughput fairness in this test scenario. This discrepancy arises because the general airtime fairness algorithm miscalculates the airtime for each STA, leading to a substantial deviation from the actual results.

In contrast, our proposed airtime fairness algorithm accurately calculates airtime, enabling the AP to achieve excellent airtime fairness through a QoS-based scheduling approach. The experimental results closely align with the theoretical optimal solution, which has a fairness index of 1.

TCP Congestion Control. Since the primary goal of this paper is to improve the TCP airtime fairness, it is essential to evaluate the performance of our QoS-based scheduling method under different congestion control algorithms. This is because congestion control is one of the core components of TCP. We conduct experiments in this download scenario,

where we sequentially set the same congestion control algorithms (Reno [23], Vegas [27], Cubic [28], and BBR [29]) to the three STAs. All other experimental conditions are kept consistent with those previously described. The results are presented in Table 4.

Table 4. Performance comparison under different TCP congestion control algorithms.

DelACK	TCP Congestion Control Algorithm	Total Throughput	Airtime Fairness Index
1	Reno	21.14	0.9992
1	Vegas	21.15	0.9992
1	Cubic	21.14	0.9992
1	BBR	21.12	0.9972
2	Reno	26.03	0.9998
2	Vegas	25.96	0.9996
2	Cubic	25.97	0.9996
2	BBR	25.97	0.9995

The experimental results show that our method is effective and demonstrates consistently high fairness when facing different TCP parameters and popular congestion control algorithms. The reason is that we have incorporated the TCP parameters that affect our QoS-based scheduling algorithm into the modeling analysis.

Upload Scenario. Next, we examined the impact of each algorithm on upload TCP flows using the network topology shown in Figure 9, with the throughput for each STA detailed in Table 5.

Table 5. Throughput comparison of QoS algorithms in UL scenario.

DelACK	QoS Algorithm	Sta1	Sta2	Sta3	Total Throughput
1	Default	6.86	6.8	6.9	20.57
1	General Airtime Fairness	13.98	4.45	3.09	21.61
1	Our Airtime Fairness	8.46	7.37	5.32	21.16
2	Default	11.76	6.83	7.52	26.11
2	General Airtime Fairness	14.45	7.32	5.09	26.87
2	Our Airtime Fairness	11.33	8.83	6.45	26.61

Building on our airtime analysis of TCP ACKs in Section 4.3, we extended the AP-based airtime fairness to upload traffic. The comparison between the default QoS algorithm and the two airtime fairness approaches demonstrates that our algorithm effectively controls changes in upload traffic throughput at the AP, thereby achieving airtime fairness.

However, in this scenario, the default QoS algorithm only achieves throughput fairness when the delayed ACK is set to 1. In contrast, our algorithm consistently achieves airtime fairness across various network environments, leading to an improvement in the overall throughput of the wireless network. The airtime for each STA in this scenario is presented in Table 6.

Table 6. Airtime comparison of QoS algorithms in UL scenario.

DelACK	QoS Algorithm	Sta1	Sta2	Sta3	Fairness Index
1	Default	0.16	0.17	0.24	0.9642
1	General Airtime Fairness	0.32	0.11	0.1	0.7659
1	Our Airtime Fairness	0.19	0.18	0.18	0.9995
2	Default	0.21	0.14	0.22	0.9611
2	General Airtime Fairness	0.26	0.15	0.15	0.9202
2	Our Airtime Fairness	0.2	0.18	0.19	0.9961

In the upload scenario, the airtime results for each station generally align with those observed in the download scenario. However, it is notable that when the delayed ACK is set to 2, the general airtime fairness algorithm produces significantly higher results compared

to other scenarios. In contrast, our airtime fairness algorithm consistently achieves high fairness across different scenarios, owing to its precise calculation of each station's airtime.

In summary, the results of all the experiments reveal that the general airtime fairness algorithm calculates transmission airtime based on frame size and wireless rate, which is then used in the scheduling algorithm. This approach allows stations with higher wireless rates to have a greater opportunity to occupy the wireless channel, leading to a positive correlation between the throughput of each node and its wireless rate, thereby increasing total throughput. However, this approach presents two main issues:

Inaccurate Airtime Calculation: As the modulation and coding scheme increases under the same wireless protocol, the wireless transmission rate speeds up, reducing the proportion of time needed for data transmission. When the wireless rate is high, the general airtime fairness algorithm's airtime calculation only reflects a fraction of the total time cost. This method significantly underestimates the actual airtime for nodes with higher rates, erroneously allocating them more transmission opportunities, which results in a poor airtime fairness index.

TCP ACK Considerations: TCP data packets require the return of TCP ACKs. Given that the wireless channel operates in half-duplex mode, the airtime cost associated with ACKs should be included in the total airtime calculation for the corresponding TCP data frame. The airtime for a single TCP data frame should therefore include both the forward data packet and the reverse ACK. This total airtime should be pre-calculated during AP scheduling to inform the scheduling logic. Our algorithm better reflects the concept of airtime fairness by incorporating these considerations.

6. Conclusions

This paper analyzes the current issues with airtime fairness in wireless networks, specifically the inaccuracy of airtime calculation. Based on this analysis, a QoS scheduling algorithm deployed on the AP is proposed, which accurately calculates the airtime of each flow and then allocates channel access opportunities accordingly, achieving airtime fairness. The proposed algorithm is effective for both upload and download traffic and has produced consistent results across different test scenarios. In all test scenarios, our algorithm achieves the Jain's fairness index greater than 0.995. This numerical result is very close to the theoretical optimal value and outperforms all other comparison algorithms. In the future, we plan to test the performance of this algorithm in a wider range of scenarios. A more potential challenge is to integrate the analysis methods in this paper with other wireless networks, such as exploring and optimizing the airtime fairness in 5G or ZigBee networks.

Author Contributions: All authors have made relevant contributions to this work. The proposal of the problem was conducted by Y.C. and J.Y., and it was developed in numerous meetings involving Y.C. and J.Y. The proposal, development, and programming of the algorithm were conducted by Y.C. The writing and editing of the paper were performed by Y.C. and J.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Fundamental Research Funds for the Central Universities (Grant No. CUC24CGJ08).

Data Availability Statement: The data that support the findings of this study are available from the corresponding authors upon reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Cisco Annual Internet Report (2018–2023) White Paper. Available online: <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html> (accessed on 15 August 2024).
2. Pilosof, S.; Ramjee, R.; Shavitt, Y.; Sinha, P. Understanding TCP fairness over Wireless LAN. In Proceedings of the INFOCOM 2003, San Francisco, CA, USA, 30 March–3 April 2003; Volume 2, pp. 863–872.

3. Wu, Y.; Niu, Z.; Zheng, J. Study of the TCP upstream/downstream unfairness issue with per-flow queuing over infrastructure-mode WLANs. *Wirel. Commun. Mob. Comput.* **2005**, *5*, 459–471. [CrossRef]
4. Sathya Priya, S.; Murugan, K. Enhancing TCP fairness in wireless networks using dual queue approach with optimal queue selection. *Wirel. Pers. Commun.* **2015**, *83*, 1359–1372. [CrossRef]
5. Network Simulator 3. Available online: <https://www.nsnam.org/> (accessed on 15 August 2024).
6. Huaizhou, S.; Prasad, R.V.; Onur, E.; Niemegeers, I. Fairness in wireless networks: Issues, measures and challenges. *IEEE Commun. Surv. Tutor.* **2013**, *16*, 5–24. [CrossRef]
7. Mohajerzadeh, A.H.; Yaghmaee, M.H.; Eskandari, Z. Tree based energy efficient and congestion aware routing protocol for wireless sensor networks. In Proceedings of the 2008 11th IEEE Singapore International Conference on Communication Systems, Guangzhou, China, 19–21 November 2008; pp. 1707–1711.
8. Le, L.B.; Hossain, E. Resource allocation for spectrum underlay in cognitive radio networks. *IEEE Trans. Wirel. Commun.* **2008**, *7*, 5306–5315. [CrossRef]
9. Arora, A.; Yoon, S.-G.; Choi, Y.-J.; Bahk, S. Adaptive TXOP allocation based on channel conditions and traffic requirements in IEEE 802.11 e networks. *IEEE Trans. Veh. Technol.* **2009**, *59*, 1087–1099. [CrossRef]
10. Huang, R.; Kim, S.; Zhang, C.; Fang, Y. Exploiting the capacity of multichannel multiradio wireless mesh networks. *IEEE Trans. Veh. Technol.* **2009**, *58*, 5037–5047. [CrossRef]
11. Torabzadeh, M.; Ajib, W. Packet scheduling and fairness for multiuser MIMO systems. *IEEE Trans. Veh. Technol.* **2009**, *59*, 1330–1340. [CrossRef]
12. Selvaradjou, K.; Handigol, N.; Franklin, A.A.; Murthy, C.S.R. Energy-efficient directional routing between partitioned actors in wireless sensor and actor networks. *IET Commun.* **2010**, *4*, 102–115. [CrossRef]
13. Tokar, L.; Krasnozheniuk, Y. Analysis of Airtime Fairness Technology Application for Fair Allocation of Time Resources for IEEE 802.11 Networks. In *International Scientific-Practical Conference “Information Technology for Education, Science and Technics”*; Springer: Cham, Switzerland, 2022; pp. 635–655.
14. What is TP-Link EAP/CAP Airtime Fairness? Available online: <https://www.tp-link.com/ae/support/faq/2095/> (accessed on 15 August 2024).
15. Chapter: Air Time Fairness. Available online: https://www.cisco.com/c/en/us/td/docs/wireless/controller/9800/17-6/config-guide/b_wl_17_6_cg/m_air_time_fairness.html (accessed on 15 August 2024).
16. Questions About Airtime Fairness. Available online: <https://www.linksys.com/gb/support-article/?articleNum=50167> (accessed on 15 August 2024).
17. Yu, S.I.; Park, C.Y. A Responsible Airtime Approach for True Time-Based Fairness in Multi-Rate WiFi Networks. *Sensors* **2018**, *18*, 3658. [CrossRef] [PubMed]
18. Bellalta, B.; Carrascosa, M.; Galati-Giordano, L.; Geraci, G. Delay analysis of IEEE 802.11 be multi-link operation under finite load. *IEEE Wirel. Commun. Lett.* **2023**, *12*, 595–599. [CrossRef]
19. Dai, L. Stability and delay analysis of buffered Aloha networks. *IEEE Trans. Wirel. Commun.* **2012**, *11*, 2707–2719. [CrossRef]
20. Nithya, B.; Mala, C.; Kumar, V. Simulation and performance analysis of various IEEE 802.11 backoff algorithms. *Procedia Technol.* **2012**, *6*, 840–847. [CrossRef]
21. Shreedhar, M.; Varghese, G. Efficient fair queuing using deficit round-robin. *IEEE/ACM Trans. Netw.* **1996**, *4*, 375–385. [CrossRef]
22. Riggio, R.; Miorandi, D.; Chlamtac, I. Airtime deficit round robin (ADRR) packet scheduling algorithm. In Proceedings of the 2008 5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, Atlanta, GA, USA, 29 September–2 October 2008; pp. 647–652.
23. Jacobson, V. Congestion avoidance and control. *ACM Sigcomm Comput. Commun. Rev.* **1988**, *18*, 314–329. [CrossRef]
24. Rastogi, S.; Zaheer, H. Comparative analysis of queuing mechanisms: Droptail, red and nlred. *Soc. Netw. Anal. Min.* **2016**, *6*, 70. [CrossRef]
25. Floyd, S.; Jacobson, V. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.* **1993**, *1*, 397–413. [CrossRef]
26. Nichols, K.; Jacobson, V. Controlling queue delay. *Commun. ACM* **2021**, *55*, 42–50. [CrossRef]
27. Brakmo, L.S.; O’malley, S.W.; Peterson, L.L. TCP Vegas: New techniques for congestion detection and avoidance. In Proceedings of the Conference on Communications Architectures, Protocols and Applications, London, UK, 31 August–2 September 1994; pp. 24–35.
28. Ha, S.; Rhee, I.; Xu, L. CUBIC: A new TCP-friendly high-speed TCP variant. *Acm Sigops Oper. Syst. Rev.* **2008**, *42*, 64–74. [CrossRef]
29. Cardwell, N.; Cheng, Y.; Gunn, C.S.; Yeganeh, S.H.; Jacobson, V. Bbr: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time. *Queue* **2016**, *14*, 20–53. [CrossRef]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.