Received 10 June 2024, accepted 9 July 2024, date of publication 22 July 2024, date of current version 31 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3431577

RESEARCH ARTICLE

Feedback-Based Control Loop Congestion Control Algorithm for Wireless Networks

RAMYASHREE VENKATESH BHAT, JETMIR HAXHIBEQIRI, (Member, IEEE), INGRID MOERMAN, (Senior Member, IEEE), AND JEROEN HOEBEKE
IDLab, Ghent University–imec, 9052 Ghent, Belgium

Corresponding author: Ramyashree Venkatesh Bhat (ramyashreevenkatesh.bhat@ugent.be)

This work was supported in part by Fonds voor Wetenschappelijk Onderzoek-Vlaanderen (FWO)-Flanders under Agreement S003921N and Agreement G055619N; in part by the Flemish Government under the ''Onderzoeksprogramma Artificiële Intelligentie Vlaanderen'' Program; and in part by the Consortium of Research Funding Organizations in Europe (CHIST-ERA) grant Sustainable and Adaptive Ultra-High-Capacity Micro Base Stations (SAMBAS) through FWO, French National Research Agency (ANR), National Research, Development and Innovation Office (NKFIH), Hungary, and UK Research and Innovation (UKRI), under Grant CHIST-ERA-20-SICT-003.

ABSTRACT Wi-Fi plays a crucial role in connecting private professional networks by providing varying data rates based on channel quality. Despite advancements in Wi-Fi protocols, there is still a problem of throughput degradation for higher data rate devices when lower data rate devices are around, due to airtime unfairness. The diverse requirements of emerging applications emphasize the necessity for network protocols that cater specifically to these needs. To tackle the problem of throughput degradation in devices that use higher physical data rates, we have designed a control-loop congestion control algorithm that decides the application data transfer rate based on the real-time network context and aggregated airtime feedback from the access point. The design is based on in-band network telemetry and stack programmability of eBPF. Using the designed algorithm, the throughput of the device that uses a higher physical data rate is improved by 58% without compromising the overall network efficiency. The average airtime difference between the end devices of the network is as low as 18%. Other than airtime fairness, the features implemented in the algorithm and monitoring and feedback system in the access point can also be utilized for priority-based airtime fairness and several such potential use cases in wireless networks in the future.

INDEX TERMS Congestion control, airtime fairness, wireless networks, INT, APP-NET.

I. INTRODUCTION

A Wi-Fi network consists of an access point (AP) with multiple end devices connected to it. The physical data rate of the end device is determined by the channel quality which varies based on the distance of the end device from the AP, obstacles in the environment, interference from other end devices, etc. Hence in a Wi-Fi network, each end device can have a different physical data rate based on the channel conditions.

Since Wi-Fi uses Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA), a pseudo-random fairness-based channel access method, in a mixed physical data rate network, the airtime consumed by the end device with a lower physical data rate is larger than that of an end device with higher physical data rate. This unfairness in airtime consumption slows down the end devices with higher physical data rates and degrades the performance of the entire network. Therefore, in a mixed physical data rate Wi-Fi network, the throughput of all the end devices irrespective of their physical data rates is degraded by the presence of an end device with very low physical data rate [5], [3], and [6].

The diversity of application requirements that utilize Wi-Fi networks in private professional environments as well as the possibility of totally different channel conditions for different devices part of the same network requires differentiation in catering to such applications and end devices. Different applications such as cloud computing, AR/VR, online streaming, and video gaming, require high throughput, while other applications such as device monitoring, industrial

The associate editor coordinating the review of this manuscript and approving it for publication was Ding Xu.

automation, etc, require higher reliability. To prioritize the demands of diverse applications, the Enhanced Distributed Channel Access (EDCA) mechanism is standardized [4], where traffic flows are distinguished in different QoS classes. However, the airtime unfairness and throughput degradation due to a multitude of physical data rates across end devices are still evident for the same QoS class. Therefore, it is essential to address the problem of throughput degradation in mixed physical data rate Wi-Fi networks, so not all devices will suffer from it.

Most of the above mentioned applications use Transmission Control Protocol (TCP) to achieve error-free, ordered, reliable data transfer. TCP provides logical host-to-host communication service and thus has only an end-to-end view of the network. Therefore, the current congestion control (CC) algorithms employed by TCP operate based on the partial end-to-end information available from the acknowledgment (ACK) packets. Moreover, the existing CC algorithms have been designed to maximize the utilization of wired networks. However, for wireless networks, in addition to the physical data rate, CC algorithms also impact the airtime usage of individual devices.

Using concepts like Application-Network Integration (APP-NET) [2] and In-band network telemetry (INT) [1] for wireless networks, applications can communicate their service requirements to the network and simultaneously learn the real-time network state through continuous, low-overhead network monitoring. This results in an application-transport-network layer interaction system where the transport layer is aware of the application requirements and the real-time end-to-end network information. The extended Berkeley Packet Filter (eBPF) has enabled the possibility of modifying the transport layer protocol including the design of the CC algorithm [9].

In this paper, the problem of throughput degradation in mixed physical data rate Wi-Fi networks is addressed by designing a Feedback-based Control loop Congestion Control algorithm (FCCC). The aim of this work is to monitor the airtime used by each end device by correcting the congestion window size based on the aggregated airtime feedback from AP. The key contributions of this paper are,

1) Design of an aggregated airtime feedback system based on airtime usage in AP.
2) Design of a control loop-based CC algorithm, FCCC.
3) Performance evaluation of the designed CC algorithm in terms of airtime fairness, throughput, and robustness to changing network conditions.

The rest of this article is structured as follows, Section II elaborates on the existing works in the area of airtime fairness in wireless networks and congestion control algorithms for wireless networks. Section III provides information on unfairness in airtime usage in Wi-Fi. The goals of the article and the essential frameworks are explained in Section IV. The design and implementation of aggregated airtime feedback system in AP are elaborated in Section V. Section VI provides a detailed explanation of the design and implementation of FCCC for wireless networks. The performance of FCCC algorithm is explained and the research outcomes are discussed in Section VII. Finally, Section VIII concludes this article with insights into future work.

II. RELATED WORK

Existing works on achieving airtime fairness in wireless networks can be categorized into two approaches, either modifications to the Medium Access Control (MAC) layer or imposing scheduling policies in the AP. In this section, different research works that explore airtime fairness in wireless networks based on these two categories are elaborated.

A. RESEARCH WORKS ON MAC LAYER MODIFICATIONS FOR AIRTIME FAIRNESS

In [11], the authors have designed an algorithm to calculate the weight of each node based on their spatial condition and the number of interfering nodes in multi-hop wireless networks. The computed weights are used to define the fragmentation threshold and buffer size and are added as an additional feature to the 802.11 MAC protocol. The authors have shown that adding this feature improves the overall performance in terms of individual throughput, end-to-end throughput, and fairness. The implementation requires too much computational power and the authors do not address the robustness of the design with changing network conditions. The authors of [12] propose an algorithm that derives the optimal uplink and downlink transmission probabilities i.e. adaptive backoff based on the estimation of backlog sizes. One of the drawbacks is that the system is implemented and tested in a simulator whereas modifying the contention window size in the existing network nodes is a challenge. In [13], the authors have designed an adaptive algorithm that modifies the contention window based on the analysis of mean packet arrival rates at end nodes to achieve airtime fairness thus maximizing the throughput. The work does not discuss the implementation in real-time wireless networks and is confined to the stability region of the arrival rate. The authors of [14] propose a differentiated reservation algorithm to reduce the collision among contending nodes and overuse of shared channels by low bitrate nodes. Once the device accesses the channel, the designed algorithm assigns a deterministic value to the backoff counter and allows certain nodes to send multiple packets in one transmission. The authors also propose a Group-based Differentiated Reservation algorithm for high-density scenarios.

B. RESEARCH WORKS ENFORCING SCHEDULING POLICIES IN AP FOR AIRTIME FAIRNESS

In [15], the authors have implemented and validated responsible airtime fairness through a fair scheduler in AP. The novelty of responsible airtime fairness lies in the fact that it achieves time-based fairness by including both data transmission time and the overhead of TCP ACK segments in TCP traffic and calculating the fairness index based on the throughput measurement. The authors do not consider the congestion in AP and the implementation is not tested in real-time wireless network environments. The authors of [16] have implemented a time-slotted scheduling approach to achieve fairness in content dissemination in mobile social networks. The authors achieve fair airtime allocation based on GO-coordinated dissemination as a generalized Nash bargaining game, where the GO-coordinated dissemination schedules the data to be sent in the application layer. The algorithm is intended for Wi-Fi-direct only and its extension to Wi-Fi is not considered. The authors of [17] aim to achieve airtime fairness using the traffic control and queuing discipline features of the Linux kernel in AP. Here, the AP collects the link capacities of the nodes, calculates the weight and quota for each node, and sets the queuing discipline based on this quota. The implementation, however, lacks the complete utilization of available bandwidth and does not consider the congestion in AP. In [18], the authors have designed a new integrated queuing system in the Linux kernel to achieve low latency in WiFi networks. Utilizing this, the authors have designed a deficit-based scheduler in AP to achieve airtime fairness. The new queuing system that bypasses the queuing discipline of Linux adds additional packet processing overhead and is less flexible. In [19], the authors have designed a REACT architecture where the end nodes bid their airtime requirements in an ad-hoc setting. The design has been extended to support differentiated airtime service as per the end node's requirement. The authors of [20] have analyzed the airtime distribution in Wi-Fi networks and tested the implications of Airtime Fairness technology implemented in AP equipment available on the market. The technology is based on Time Division Multiple Access, where each device gets equal or specific period of time for data transmission in cyclical order. The authors validate that the Airtime Fairness Technology works efficiently even in the presence of 20 or more users, networks with reduced coverage radius, and HotSpot networks. All the above research works, their novelties, and drawbacks are summarised in Table 1.

The modifications in the MAC layer generally involve setting the contention window size or buffer size based on various network parameters. In the case of scheduling policies, the APs allot specific transmitting time to each node based on different network parameters as well. Except for a few numerable works, most of the implementations are validated in a simulated environment. Existing research works also fail to consider the possible network congestion in AP, the robustness of the algorithm to changing network conditions, and their impact on a real-time wireless network. Since wireless network conditions are unpredictable and modification to commercially available network devices itself is a challenge, we specifically emphasize the fact of implementation and testing in real-time wireless networks and consider it as one of the novel features of this design.

C. RESEARCH WORKS ON ADAPTIVE CC ALGORITHMS

Unlike User Datagram Protocol (UDP), in a TCP connection, the receiver sends an ACK as it receives the data from the sender. Additionally, TCP detects packet loss based on timeout or 3 duplicate ACK events and retransmits the lost data to the receiver, thus achieving reliable and error-free data transfer. Among other mechanisms used by TCP, the CC algorithm decides the amount of data to be sent to the receiver at each instance of time by calculating the congestion window size. Moreover, the CC algorithms are designed for high bandwidth wired data links to maximize the usage of the link while at the same time achieving fairness and avoiding congestive collapse. Once the congestion window size is decided, the segment is converted to a packet and then an IEEE 802.11 frame for wireless transmission. Once the frame is ready the sender contends for the channel access and transmits the frame when the channel is idle. Therefore, along with the physical data rate, the congestion window size set by the CC algorithm is also responsible for airtime consumed.

In [21], the authors have tried combining two traffic shaping methods, the Hierarchical Token Bucket shaping method (arbitrary number of token buckets are arranged in hierarchy) and the Receive Window Tuning Method (configure TCP connections by tuning receive window buffer size) with four different CC algorithms NewReno, Vegas, Illinois, and CUBIC, to improve the quality of experience of HTTP adaptive streaming. The authors have validated that the combination of the Receive Window Tuning Method and Illinois CC algorithm offers the best quality of experience for this application scenario. The authors of [22] have

TABLE 1. Summary of research works in the area of achieving airtime fairness in wireless networks.

MAC layer modifications

Reference	Novelty	Drawbacks
[11]	Additional feature to 802.11MAC: fragmentation threshold and buffer size	Too much computational power and do not address the robustness
[12]	Adaptive backoff based on the estimation of backlog sizes	Modifying the contention window size in the commercially available wireless devices itself is a challenge
[13]	Modifying the contention window size based on mean packet arrival rates	The implementation is confined to the stability region of arrival rate
[14]	Differentiated reservation algorithm: deterministic backoff counter value and allows the nodes to send multiple packets in one transmission	Increased collision with increase in number of nodes

Scheduling policies in AP

Reference	Novelty	Drawbacks
[15]	Fair scheduler: time-based fairness based on throughput measurement	Congestion in AP is not considered in the design
[16]	Fair airtime allocation based on GO-coordinated dissemination	Implemented for Wi-Fi direct and is not extended to Wi-Fi
[17]	Traffic control and queuing discipline features of Linux kernel in AP to calculate the weight and airtime quota of each node	Does not completely utilize the available network bandwidth and does not consider the congestion in AP
[18]	Deficit-based scheduler in AP based on new integrated queuing system in Linux	The new queuing system adds additional packet processing overhead and is less flexible
[19]	REACT architecture where ends nodes bid their airtime in ad-hoc network	Not extended for infrastructure Wi-Fi networks
[20]	Time Division Multiple Access in specific AP equipment available in the market	The feature is not available in all the APs and does not support roaming clients

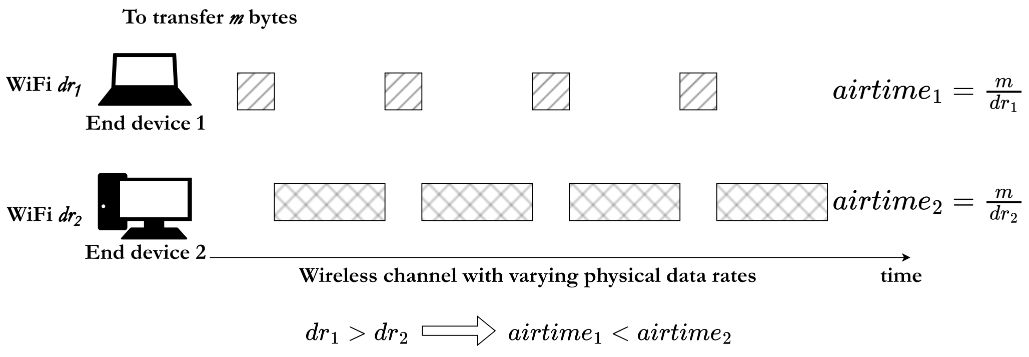
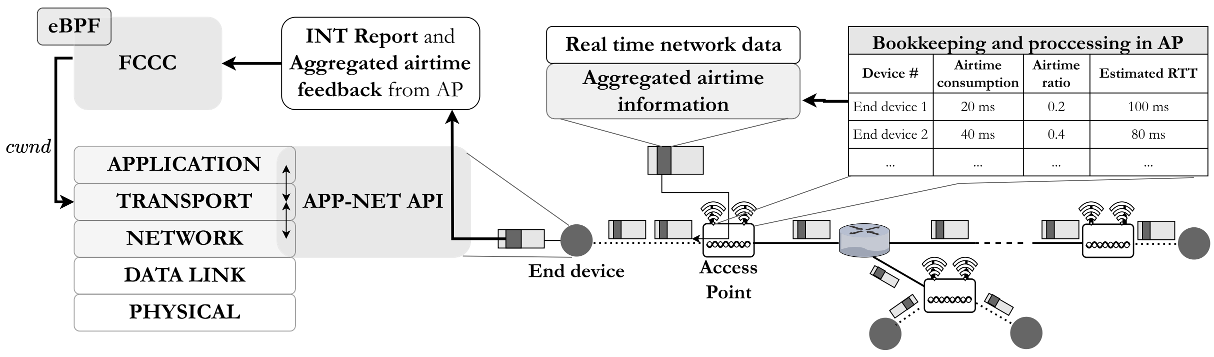

To transfer m bytes
WiFi dr_1
End device 1
$airtime_1 = \frac{m}{dr_1}$
WiFi dr_2
End device 2
$airtime_2 = \frac{m}{dr_2}$
Wireless channel with varying physical data rates
time
$dr_1 > dr_2 \Longrightarrow airtime_1 < airtime_2$

FIGURE 1. Difference in airtime in a wireless network with varying physical data rates.



Bookkeeping and proccessing in AP

Device #	Airtime consumption	Airtime ratio	Estimated RTT
End device 1	20 ms	0.2	100 ms
End device 2	40 ms	0.4	80 ms
...	...	...	...

FIGURE 2. Network architecture overview with additional processing at AP.

designed an adaptive CC method to avoid congestion in an IoT network with CoAP as an application layer protocol. The designed solution operates based on traffic priority assignment, adaptive retransmission timeout, and adaptive backoff timer. In [23], the authors have designed an adaptive CC to reduce the energy consumption in the Internet of Vehicles. The designed algorithm operates based on the two feedback bits indicating the status of buffer space and link utilization in the acknowledgment packet, appended by the receiver. The above-mentioned adaptive CC algorithms are specific to particular use cases like HTTP adaptive streaming, Internet of Vehicles, and CoAP-based IoT networks. The algorithms rely on partial feedback from the network and are unaware of other devices and their airtime utilization.

III. BACKGROUND ON UNFAIRNESS IN AIRTIME USAGE IN WI-FI NETWORKS

In a wireless network, unlike a wired network with fixed data rates, each end device connecting to an AP has a different physical data rate depending on the channel quality. For example, the latest Wi-Fi standard 802.11ax offers different physical rates ranging from 8.6 Mbps to 9.6 Gbps, i.e. the end device can choose a suitable modulation and coding scheme based on the channel quality. A few major factors that influence the channel quality in a Wi-Fi network are the signal strength and interference from other end devices. Hence, in a Wi-Fi network with multiple end devices, each device would have a different physical data rate at each instance of time.

In a wired Ethernet network, the Carrier Sense Multiple Access with Collison Detection (CSMA/CD) technique is used to ensure pseudo-random fairness in media access among multiple end devices. This would mean that a wired link of x Mbps capacity would be divided equally among n end devices, resulting in a throughput of x/n Mbps. In the case of a Wi-Fi network, though CSMA/CA [10] ensures pseudo-random fairness in channel access, because of the changing wireless channel conditions, the estimation of throughput is not as simple as in a wired network. The end device with the poor wireless channel quality experiences a lower physical data rate and hence requires more airtime to transfer the same amount of data as compared to the end device with better channel quality and higher physical data rate. Therefore, to send m byte of data, the lower physical data rate device would consume more airtime than the higher physical rate device as shown in Figure 1. Airtime unfairness is an evident problem of Wi-Fi networks that is still to be addressed.

The side effect of airtime unfairness is the significant degradation in the performance of the end devices with higher physical data rates. This has been experimentally validated by the authors of [5] and [6]. They have proven that the throughput of end devices with higher physical data rates is significantly degraded in the presence of end devices with lower physical data rates.

TABLE 2. Summary of abbreviations, their meanings and units.

Parameters at AP

Parameter	Meaning	Unit
dr	Physical data rate of an end device recorded by AP	Mbps
dc	Device count, Number of end devices using the channel	none
ts	Timeslot, time period over which the AP calculates the aggregated airtime usage	second
t_p	Airtime of packet	second
p_{ip}	IP payload	byte
sp	Percentage of ts that can be used by the device including the margin for new connection initiation	%
ar	Aggregated airtime feedback calculated by AP over ts	%
w_ar	Weighted average of aggregated airtime feedback calculated by AP over ts	%
$loss_w$	Wireless packet loss	packet

Fixing timeslot at AP

Parameter	**Meaning**	**Unit**
$ertt$	RTT estimated by AP	second
$e2a_l$	Latency between end device to AP	second

Parameters at end device

Parameter	**Meaning**	**Unit**
t_p	Airtime of packet	second
c_p	Correction factor	none
$cwnd_{up}$	Congestion window upper bound	packet
$error$	Error between the aggregated airtime feedback and target sp	none
K_p	Proportional constant of PI controller	none
K_i	Integral constant of PI controller	none
e_p	Proportional error of PI controller	none
e_i	Integral error of PI controller	none
PI_{error}	The summation of proportional and integral error in PI controller equation	none
$cwnd$	Congestion window size	packet
$cwnd_{prev}$	Congestion window size of previous data transfer	packet

Parameters from kernel using eBPF

Parameter	**Meaning**	**Unit**
$loss_p$	Packets lost due to congestion in the network	packet
$bytes_{ACK}$	Bytes acknowledged by the receiver	byte
$bytes_{SACK}$	Bytes selectively acknowledged by the receiver	byte
p_{size}	Size of the packet	byte
$flow_{timeout}$	Flow timeout notification	boolean
MSS	Maximum Segment Size	byte

Transmit Opportunity (TXOP) is a MAC feature in 802.11 that provides channel-free access to high-priority data for a certain period. TXOP limit is supported only for access categories, Voice and Video. The TXOP limit cannot be dynamically changed for each access category on the fly. Moreover, the TXOP feature is applied to an entire access category, hence it is applied to all the flows that pass by the dedicated queue for that access category. This restricts the application of different service treatments to different flows. If all the devices enable the TXOP feature, then the end device with a lower physical data rate would still slow down the end device with a higher physical data rate, hence the problem of airtime unfairness persists between the flows belonging to the same access category.

IV. GOALS AND ESSENTIAL FRAMEWORKS

This article elaborates on the design of FCCC algorithm that improves the throughput of end devices with higher physical data rates in wireless networks based on the aggregated airtime feedback from AP. To achieve this, the CC algorithm should consider **past events** and **foresee** the airtime usage, hence should be **proactive**. The CC algorithm should be **stable**, i.e. have a stable application data transfer rate to achieve better throughput. The algorithm should be **robust** to changing physical data rates and other network conditions. The actions taken by the end device are dependent on INT feedback on real-time network conditions as well as aggregated airtime feedback from AP. Since the end devices are unaware of the other devices in the network, it is the responsibility of AP to give the **aggregated airtime feedback** considering the ongoing changes in the network also ensuring the **privacy** of other end devices.

To design FCCC, the transport layer of the end device should be updated with the INT and aggregated airtime feedback from AP. The end device should be able to recalculate and assign the congestion window size based on this feedback. These operations are facilitated by the following frameworks:

1) **INT for wireless networks:** The INT framework for wireless networks designed in [1] is a low overhead, continuous network monitoring technique that collects real-time network states such as physical data rate, RSSI, buffer capacity, etc. from the intermediate network nodes. The intermediate network nodes add the network state information as an IPv6 extension header which can be exploited to add the aggregated airtime information in the designed feedback-based system.
2) **APP-NET interaction API:** The real-time network state information obtained from INT is extracted in the network layer of the end device. The APP-NET API designed in [2] provides an interface where an application can publish its service requirements and the network layer can publish the real-time network information collected using INT. This interface is extended to the transport layer in [7] resulting in application-transport-network layer interaction. In the designed system, this API is used to publish the INT and aggregated airtime feedback from AP to the transport layer of the end device.
3) **The CC plane (CCP):** The CCP framework designed in [9] allows the reconfiguration of CC algorithms. The framework allows the implementation of algorithms in Python programming language which adds additional flexibility to the design.

V. AGGREGATED AIRTIME FEEDBACK SYSTEM IN AP

Using INT, APs can now update the end devices connected to them with the real-time network information with additional aggregated information about other network flows. In this section, the design and implementation of the aggregated airtime feedback system in AP is elaborated. Fig.2 gives an overview of the network architecture with the additional processing in AP. The meaning and unit of each terminology used in the coming sections is listed in Table 2.

A. RELEVANT MONITORING PARAMETERS

Physical data rate (dr) directly affects the airtime consumed during a data transfer. Therefore, knowing the dr of the link is essential for the FCCC algorithm to predict future airtime usage. Since the goal is to adapt to the changing network conditions, the FCCC must also be aware of the number of end devices (dc) in the network. Hence from the available wireless telemetry options, dr, dc, and $loss_w$ are the monitoring parameters that are considered for the design. FCCC gets real-time information on these parameters from AP.

B. BOOKKEEPING IN AP

Since the wireless channel is continuously changing, AP calculates the statistics of airtime usage over a fixed period called timeslot (ts) to get a perspective on the total airtime usage of each end device using the channel. The methodology used to dynamically set the ts is explained in section V-D. To provide aggregated airtime feedback, AP maintains an airtime consumption table for the end devices connected to it. The end device is added to the table every time it initiates a TCP connection i.e. through the SYN packet. Every time the AP receives a packet from the sender end device, the AP calculates the airtime consumed for the data transfer ($airtime$) using equation 1:

$$\begin{aligned} &airtime \\ &= T_{PHY} + \frac{bits_s + bits_t + N * 8 * (h_{MAC} + h_{LLC} + p_{IP})}{dr} \end{aligned} \tag{1}$$

where 1 T_{PHY} indicates the time of preamble and signal, $bits_s$ and $bits_t$ are service and tail bits respectively, and N indicates the number of aggregated packets (in this case, 1), h_{MAC} and h_{LLC} indicate the MAC and LLC header sizes respectively in bytes, and p_{IP} indicates the IP packet size in bytes. The MAC layer overheads of distributed interframe spacing (DIFS) (T_{DIFS}), short interframe spacing (SIFS) (T_{SIFS}), and time for MAC layer ACK packet (T_{ACK}) are also added to $airtime$ to get the t_p.

$$t_p = airtime + T_{DIFS} + T_{SIFS} + T_{ACK} \tag{2}$$

The t_p recorded in a timeslot (ts_{t-1}, ts_t) is summed over to calculate the aggregated airtime ratio (ar) using equation 3

$$ar_t = \frac{\sum_{i=1}^{n} t_{p_i}}{ts_t}, t = |ts|, n = |t_p| \text{ recorded in } (ts_{t-1}, ts_t) \tag{3}$$

where t indicates the number of ts blocks over time.

The calculated ar is updated in the airtime consumption table for every ts period. To give a network overview to the end device, the AP assigns a setpoint (sp) value to each end device using equation 4, which is the percentage of ts that can be utilized by an end device with a 20% margin for initiation of a new connections. A margin of 20% is sufficient for an end device to start a new connection.[1]

$$sp = \frac{0.8}{dc} \tag{4}$$

Since the aim is to achieve equal airtime fairness among all the end devices, the sp is assigned equally.

C. STABILIZING THE AGGREGATED AIRTIME FEEDBACK

The ar feedback calculated by the AP at every ts is instantaneous feedback of the airtime usage. Making decisions based on instantaneous value can result in highly fluctuating outcomes. To avoid this the weighted average of ar of previous timeslots is averaged with the current aggregated airtime ratio for each end device. For ar values over t timeslots, the weighted average of ar, i.e., w_ar (in future referred as aggregated airtime feedback) is calculated using equation 5.

$$w\_ar_t = 2^{-t} * ar_1 + \sum_{i=2}^{t} 2^{(i-1)-t} * ar_i \tag{5}$$

This average value is calculated for each end device and stored in the airtime consumption table. The w_ar is sent as feedback after each timeslot.

D. DYNAMIC ASSIGNMENT OF TS IN AP

Wireless channels are continuously changing at each instant of time, therefore, it is essential to discretize and look at the airtime usage of each end device over a fixed ts to get a better outlook on the network state. Especially in wireless networks, the RTT is not fixed, and it depends mainly on the channel quality and the processing time in intermediate nodes of the network, therefore, the ts value is assigned dynamically.

The aggregated airtime information sent by AP is received by the sender end device through the ACK or Selective ACK (SACK) packet. The value of ts should be such that it includes at least one RTT of all the end devices using the channel at every instant of time. A very small ts value can result in missing feedback of certain end devices, whereas very large ts can result in delayed feedback and be ineffective to the changing network conditions. AP thus keeps track of the maximum RTT of each end device and dynamically assigns the ts to the maximum value of this range using equation 6. In equation 6 for n end devices using the wireless channel at each instant of time, $ertt$ is the RTT estimated by AP between the AP and receiver and $e2a_l$ is the one-way latency between the end device and AP. $e2a_l$ is estimated using the $rx\_timestamp$ of INT telemetry option.

$$\begin{aligned} ts = max((ertt + 2 * e2a_l)_1, (ertt + 2 * e2a_l)_2, \\ \ldots (ertt + 2 * e2a_l)_n) \end{aligned} \tag{6}$$

Thus, AP sends ts and sp (instead of dc) values as feedback in the IPv6 extension header along with other wireless telemetry fields such as dr and $loss_p$ in every TCP packet

[1]https://www.itweb.co.za/article/unwrapping-wifi-6-getting-out-of-the-traffic-jam/o1Jr5qx96jDvKdWL

received from the sender end device whereas, the aggregated airtime ratio, w_ar, information is sent only after every ts seconds. The INT information is added by the AP extracted by the receiver end device and sent as feedback through the ACK/SACK packet to the sender end device. In case of multiple INT information packets, the receiver adds the latest INT information as feedback. The aggregated airtime ratio along with ts and sp ensures that end devices get a perspective of their airtime usage and ongoing activities in the network without compromising the privacy of other end devices. After the completion of a TCP connection, the table entry of the end device is removed from the airtime consumption table in AP.

The modifications in the AP are implemented using the Click modular router framework.[2]

The airtime usage is decided on a per-end device basis and not a per flow basis. Therefore, from the AP's point of view, irrespective of number of flows from the end device, the aggregated airtime ratio is calculated for each end device. Though the current implementation focuses on achieving equal airtime fairness, the parameters monitored in AP (such as sp) and APP-NET allow priority-based airtime allocation in the future.

VI. DESIGN AND IMPLEMENTATION OF FCCC

In this section, design and implementation of the FCCC algorithm is elaborated. The designed algorithm uses INT and aggregated airtime feedback from AP to decide the congestion window size at each instant of time.

A. CONTROL LOOP-BASED FCCC DESIGN

The goal of achieving airtime fairness, thus improving the throughput of end devices with higher physical data rate is considered by AP by sending the aggregated airtime feedback (w_ar), sp, ts, and dr as feedback to the end device. It is the job of the CC algorithm in the end device to analyze the previous airtime usage, foresee the future airtime usage, and make the right correction in the congestion window size ($cwnd$) whilst maintaining a stable data transfer for higher throughput. To achieve these goals, the algorithm needs to know its performance i.e., the over- or under-usage of the airtime in a timeslot in terms of number of packets. Subtracting the amount of foreseen airtime usage by the previous airtime usage over a given timeslot (dividing the difference by the timeslot value) gives the correction factor (c_p), i.e. the number of packets to be added to or subtracted from previous data transfer to achieve intended airtime usage as follows:

$$c_p = cwnd * \frac{t_p}{ts} - cwnd_{prev} * \frac{t_p}{ts} \tag{7}$$

where t_p is the time per packet calculated in the end device using equation 2. (The assumption is $ts_t \approx ts_{t-1}$, since the difference between the two values over a period of time in negligible.)

[2]https://github.com/kohler/click

To achieve the above mentioned goals, the FCCC has been designed as a loop-based Proportional Integral (PI) controller. Therefore, the PI_{error} is the outcome of the PI controller (equation 8):

$$PI_{error} = K_p * error + K_i * \int_0^t error(t)dt \tag{8}$$

where $error = sp - w\_ar$ and $K_p = sp, K_i = 0.1 * sp$. The $error$ in equation 8 indicates the deviation in the airtime consumed in the previous timeslot as compared to the allocated airtime, sp, sent as feedback by AP. w_ar is the outcome of previous congestion window size ($cwnd_{prev}$). Hence c_p can be derived also as the fraction of $cwnd_{prev}$ by calculating the PI_{error} from the sp:

$$c_p = \frac{PI_{error}}{w\_ar} * cwnd_{prev} \tag{9}$$

The proportional error with proportional gain K_p determines the rate of direct response to the error, whereas integral error is the summation of the errors since the beginning of the connection, with integral gain K_i, eliminating the steady-state error linked with proportional response. The gains achieved with K_p and K_i should be in sync with the sp set by AP and thus proportional to the sp. The values of K_p and K_i were tuned to sp and $0.1 * sp$ respectively. Substituting the values, the congestion window size of the FCCC algorithm can be obtained using equation 10. Since the $cwnd$ is the unknown factor, combining and restructuring equations 7 and 9, the congestion window size of the current data transfer can be calculated.

$$\begin{gathered} cwnd = (PI_{error} * \frac{ts}{t_p}) * \frac{cwnd_{prev}}{w\_ar} + cwnd_{prev} \\ [PI_{error} * \frac{ts}{t_p}] \in [-sp, sp] \end{gathered} \tag{10}$$

By calculating the $error$ for each timeslot, the CC algorithm foresees future airtime usage. Since the $error$ is calculated from sp, the algorithm considers the other flows in the network. These features make the algorithm proactive in nature. On starting the connection, the proportional gain in the equation gives the instant gain of the $cwnd$ value thus achieving the highest possible throughput. The integral error of the equation makes the algorithm respond quickly by considering the previous airtime usage and the duration of the error. The fast response of the integral gain can also result in instability and oscillation of the overall outcome, thus, the e_i of the $cwnd$ equation is clamped between -1 and 1. Since the value of $error$ lies in the range of $-sp$ and sp, $PI_{error} * ts/t_p$ is also clamped, which in turn helps to maintain stable data transfer. The value of the congestion window size varies with the changing physical data rate through t_p, thus the algorithm is robust to the changing physical data rate. Though the outcome of the algorithm is stable and robust, and foresees the future, the design does not ensure whether the airtime used by the calculated congestion window size is under the limitation imposed by the AP. Therefore, to achieve the last

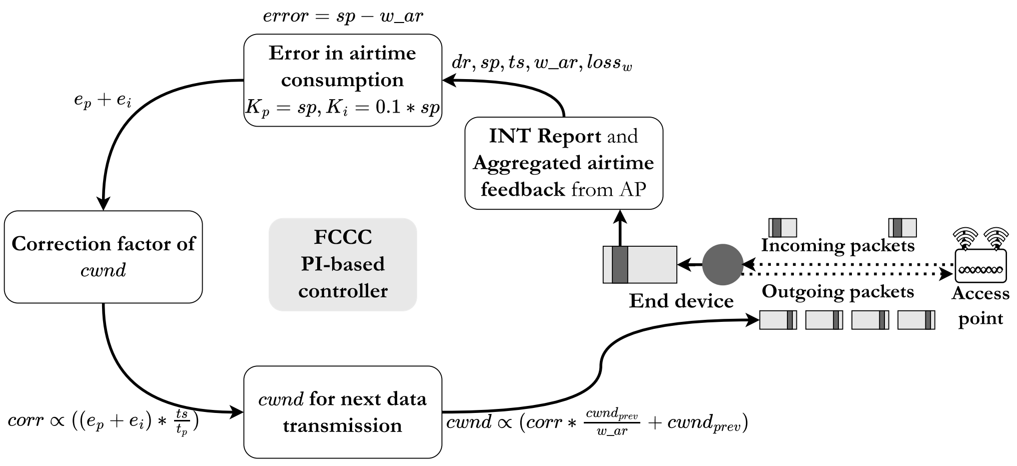

$error = sp - w\_ar$
Error in airtime consumption
$K_p = sp, K_i = 0.1 * sp$
$dr, sp, ts, w\_ar, loss_w$
$e_p + e_i$
INT Report and Aggregated airtime feedback from AP
Correction factor of $cwnd$
FCCC PI-based controller
Incoming packets
End device
Outgoing packets
Access point
$cwnd$ for next data transmission
$corr \propto ((e_p + e_i) * \frac{ts}{t_p})$
$cwnd \propto (corr * \frac{cwnd_{prev}}{w\_ar} + cwnd_{prev})$

FIGURE 3. Overview of control-loop based CC algorithm based on network feedback.

step of proactiveness, boundary condition to the congestion window size is applied. Fig. 3 gives an overview of the FCCC algorithm.

1) BOUNDS TO THE CONGESTION WINDOW SIZE

The airtime used by an end device should be less than or equal to the *sp* feedback from the AP, where airtime used over one timeslot is obtained by the product of congestion window size and time per packet divided by the timeslot.

$$w\_ar \leq sp$$
$$cwnd \leq \frac{ts}{t_p} * sp$$

Writing the above relation in an equation gives the upper bound of the congestion window size.

$$cwnd_{up} = \lfloor 1 + \frac{ts}{t_p} * sp \rfloor \tag{11}$$

The algorithm ensures that the value of the congestion window size is lower than the upper bound in equation 11. The added upper bound foresees the airtime usage, ensures stability and the dependency on *sp* allows it to adapt to the changing network conditions such as increase in the number of end devices in the network.

B. IMPLEMENTATION OF THE FCCC

The designed FCCC algorithm is represented in the algorithm 1. On starting a new TCP connection, the FCCC sends a SYN packet with *cwnd* value equal to $cwnd_{init}$. The value of $cwnd_{init}$ is equal to the *ssthresh* value set in conventional CC algorithms. The algorithm receives TCP end-to-end information such as total bytes acknowledged ($bytes_{ACK}$), total bytes selectively acknowledged ($bytes_{SACK}$), number of packet losses (p_{loss}), size of the payload sent in a packet (p_{size}) and whether the connection is lost, and TCP enters connection timeout (*flow_timeout*). In case of a timeout, the algorithm resets the variables as mentioned in *reset_function*(). If the connection is successful, the device also receives feedback from AP such as *sp*, *ts*, *w_ar* and real-time network information such as *dr* and packet loss due to wireless conditions ($loss_w$) through INT.

On receiving every ACK packet, the algorithm calculates the time per packet and congestion window upper bound based on the feedback from AP and enters the *on_ack*() function. The algorithm calculates the *cwnd* value using equation 10 for every *w_ar* feedback received from the AP. The algorithm also checks for the upper boundary condition based on the current feedback from the network. In case of no feedback, the *cwnd* value is kept the same as the previous congestion window size.

In the case of a p_{loss} notification from the kernel, the algorithm checks whether the packet loss is due to wireless conditions such as interference, poor channel quality, obstacles in the channel, etc. This is obtained by the $loss_w$ variable published by APP-NET. In case of packet loss due to a wireless channel, the congestion window value is kept constant. In case of a packet loss due to congestion in the network, the conventional CUBIC CC algorithm reduces the value to 70% of the previous value [24]. But in the case of FCCC, the performance of the algorithm was tested by reducing the value by 70%, 80%, 90%, and 95%. The FCCC performed better when the congestion window value was reduced to 90% of its previous value. Therefore, the value of the decrease factor β is set as 0.9.

The algorithm is implemented using Python programming language 3.8 with the CCP framework [9]. The end-to-end TCP information from the Linux kernel is made available to the algorithm by the CCP framework.

VII. RESULTS AND DISCUSSION

The performance of the designed FCCC algorithm was validated by defining key performance indicators (KPIs) and conducting several tests to measure the improvement

Algorithm 1 The Designed FCCC Algorithm

```
1:  Initialization:
2:  $reset\_function()$
3:  $cwnd \leftarrow cwnd_{init}$,
4:  Value reported by eBPF: $MSS$
5:  $cwnd_{out} \leftarrow MSS * cwnd$
6:
7:  $on\_report()$:
8:  Values updated from INT feedback:
9:  $dr$, $sp$, $w\_ar$, $ts$, $loss_w$
10: Values reported by eBPF:
11: $bytes_{ACK}$, $bytes_{SACK}$, $loss_p$, $p_{size}$, $flow\_timeout$, $MSS$
12: if $bytes_{ACK}$ or $bytes_{SACK}$ then
        $t_p \leftarrow \frac{bits_s + bits_t + 8 * (h_{MAC} + h_{LLC} + p_{size})}{dr} + T_{PHY} + T_{DIFS} + T_{SIFS} + T_{ACK}$
        $cwnd_{up} \leftarrow \lfloor 1 + \frac{ts}{t_p} * sp \rfloor$
13:     $on\_ack(sp, w\_ar, ts, t_p, cwnd_{up}, time_{prev}, cwnd_{prev}, MSS)$
14: else if $loss_p$ then
15:     $on\_loss(loss_w, MSS)$
16: else if $flow\_timeout$ : then
17:     $on\_timeout()$
18: end if
19:
20: $on\_clamp(value, lower, upper)$:
21: if $value \leq lower$ then
22:     $value \leftarrow lower$
23: else if $value \geq upper$ then
24:     $value \leftarrow upper$
25: end if
26: return $value$
27:
28: $on\_ack(sp, w\_ar, ts, t_p, cwnd_{up}, time_{prev}, cwnd_{prev}, MSS)$:
29: if $w\_ar \neq 0$ then
        $error \leftarrow sp - w\_ar$
        $dt \leftarrow current.time() - time_{prev}$
        $e_p \leftarrow sp * error$
        $e_i \leftarrow on\_clamp(e_i + (0.1 * sp * error * dt), -1, 1)$
        $corr \leftarrow on\_clamp(((e_p + e_i) * \frac{ts}{t_p}), -sp, sp)$
        $cwnd \leftarrow corr * \frac{cwnd_{prev}}{w\_ar} + cwnd_{prev}$
30:     if $cwnd \geq cwnd_{up}$ then
31:         $cwnd \leftarrow cwnd_{up}$
32:     end if
33:     $cwnd_{prev} \leftarrow cwnd$, $time_prev \leftarrow current.time()$
34: else
35:     $cwnd \leftarrow cwnd_{prev}$
36: end if
37: $cwnd_{out} \leftarrow MSS * cwnd$
38:
39: $on\_loss()$:
40: if $loss_w$ then
41:     $cwnd \leftarrow cwnd_{prev}$
42: else
43:     $cwnd \leftarrow cwnd_{prev} * \beta$
44: end if
45: $cwnd_{out} \leftarrow MSS * max(cwnd, cwnd_{init})$
46:
47: $flow\_timeout()$:
48: $reset\_function()$
49:
50: $reset\_function()$:
51: $cwnd_{init} \leftarrow 10$, $\beta \leftarrow 0.9$, $time_{prev} \leftarrow current.time()$,
    $cwnd_{prev} \leftarrow cwnd_{init}$,
52: $bytes_{ACK} \leftarrow 0$, $bytes_{SACK} \leftarrow 0$, $p_{loss} \leftarrow 0$, $p_{size} \leftarrow 0$,
    $dr \leftarrow 0$, $sp \leftarrow 0$,
53: $ts \leftarrow 0$, $loss_w \leftarrow 0$, $error \leftarrow 0$, $e_p \leftarrow 0$, $e_i \leftarrow 0$,
    $corr \leftarrow 0$,
```

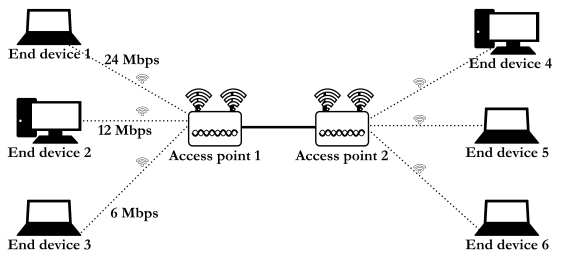


FIGURE 4. Wireless network setup used to test the sensitivity to *ts* length and compare the performance of FCCC and CUBIC.

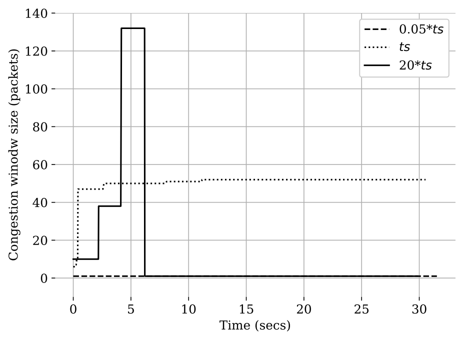


FIGURE 5. *cwnd* for different *ts*.

in throughput and airtime fairness achieved. The values of *cwnd* and instantaneous airtime consumed are used as KPIs to measure the sensitivity to timeslot length. The improvement in the throughput and airtime fairness between the wireless end devices are the KPIs to validate the performance of the FCCC algorithm. The behavior of *cwnd* and $cwnd_{up}$ for varying physical data rates are the KPIs to evaluate the robustness of FCCC algorithm to changing data rates. The instantaneous airtime consumption and *cwnd* are the KPIs for the use case test with different *sp* values from AP. This

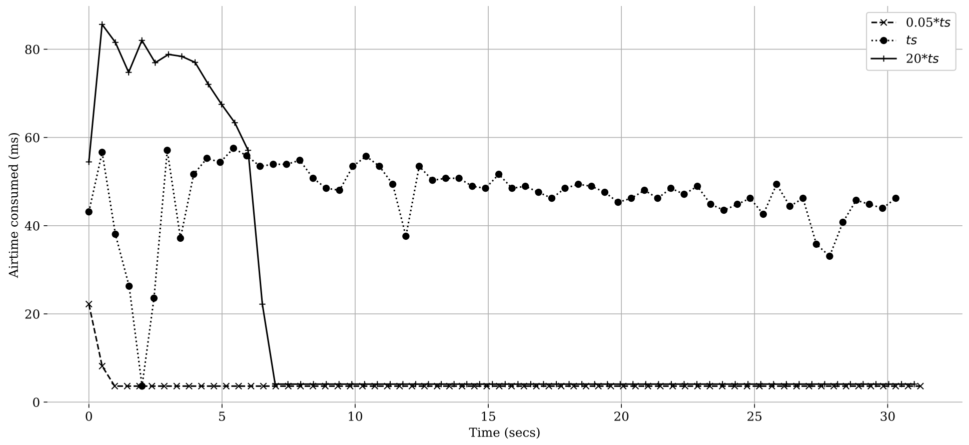


FIGURE 6. Instantaneous airtime consumption for different *ts* values.

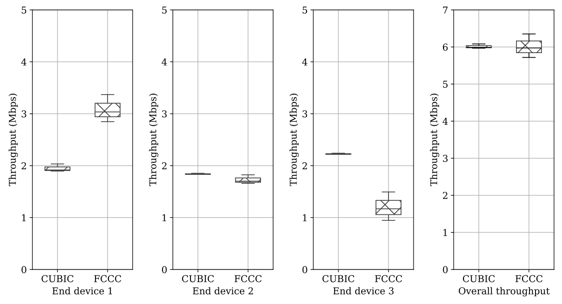


FIGURE 7. Average throughput of the FCCC algorithm as compared to the CUBIC.

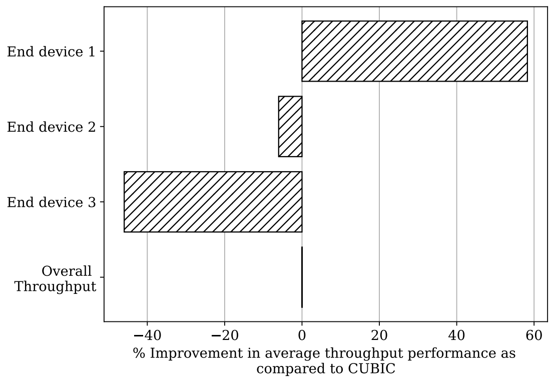


FIGURE 8. Percentage change in average throughput of the FCCC algorithm as compared to the CUBIC.

section will elaborate on the test setup and the tests carried out to validate the performance.

A. TEST SETUP

One of the drawbacks that is evident in the state of the art of CC algorithms innovation on top of WiFi is the lack of implementation and experimentation in the commercially available wireless devices. Most of the research works are built and validated in a simulated environment. Therefore, the designed FCCC algorithm is implemented on a commercially available wireless node and the experiments are conducted in a WiLab2 testbed.[3] The experimental configuration includes a wired connection linking two APs who serve six end devices in total, with three end devices linked to each AP. The APs are configured for 802.11n WiFi standard and the aggregation is disabled. The airtime distribution is done on per-device basis. Allocation of airtime allotment among different flows in an end device is the task of the end device itself and is out of the scope of this work. Therefore, each end device has been restricted to one flow. To clearly see the performance of the designed algorithm, the physical data rates of the end devices have been fixed to 6, 12, and 24 Mbps using `iwconfig` command for different tests. The INT is enabled in every packet to get the clear picture of ongoing changes in the network and evaluate the impact of FCCC algorithm accordingly. Computational overhead added by INT and the age of information is previously measured in [26] and it is shown to be limited.

B. SENSITIVITY OF FCCC TO TIMESLOT LENGTH

The *ts* length after which the AP sends the *w_ar* feedback has a major impact on the performance of the FCCC algorithm, hence the sensitivity of the FCCC algorithm to different *ts* lengths was measured. The AP dynamically calculates the *ts* period based on the maximum end-to-end packet latency of different traffic flows. The performance of the algorithm was also tested when the *ts* value was 20 times less and 20 times more than this calculated period. The test was conducted in a wireless network setup using two end devices, End device 1 and End device 3, with the physical data rate values fixed to 24Mbps and 6Mbps respectively as shown in Fig. 4. End devices 4 and 6 are used as receiver en devices. The values of *cwnd* and instantaneous airtime consumed are used as KPIs to measure the sensitivity to timeslot length. The test was run

[3]https://doc.ilabt.imec.be/ilabt/wilab/

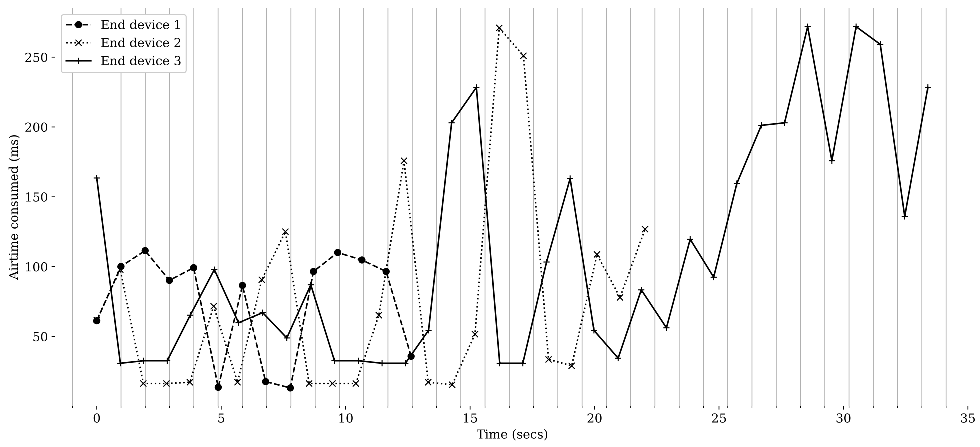


FIGURE 9. Instantaneous airtime consumption of the FCCC algorithm on sending a fixed data payload of 5MB (minor grid lines indicate *ts*).

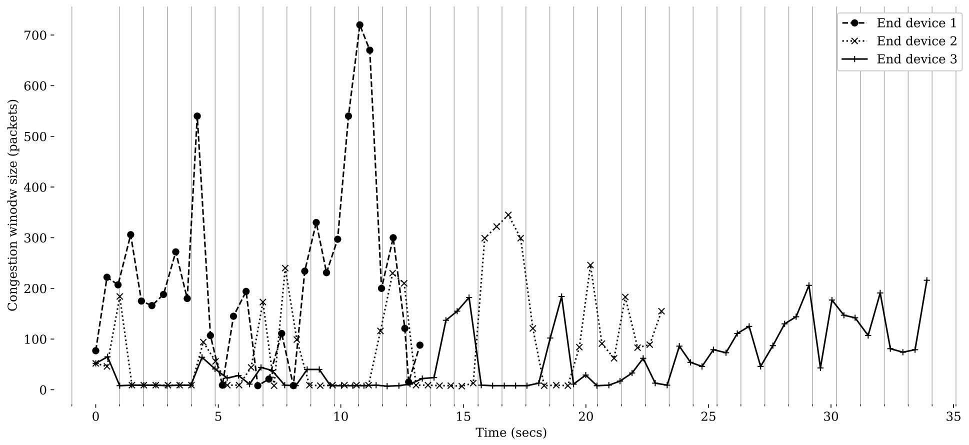


FIGURE 10. Behaviour of *cwnd* of the FCCC algorithm on sending a fixed data payload of 5MB (minor grid lines indicate *ts*).

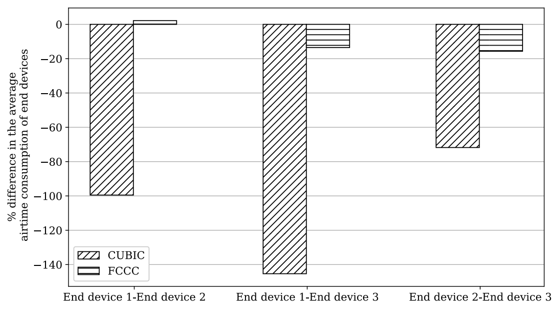


FIGURE 11. Percentage difference between average airtime consumption of end devices.

for a fixed period and the outcome of *cwnd* and instantaneous airtime consumption value used by the algorithm for different *ts* periods by End device 1 is plotted in Fig. 5 and Fig. 6 respectively.

For the case when *ts* is 20 times less, the *ts* period is approximately 48ms and it is very short, hence the AP cannot consider the airtime usage of all the end devices (even in this toy case with only two end devices) using the network for such a short period. Hence it appears to AP as if End device 1 is overusing the network. Therefore, it asks the End device 1 to reduce the airtime usage in the form of feedback with every ACK packet. The same effect can be seen in Fig. 5 where the *cwnd* value is kept to the minimum possible value, the impact of which can be seen in Fig. 6 where the airtime usage is very low. Though the airtime usage is very low, because of the very small *ts* value, the airtime usage appears to be relatively high compared to the setpoint *sp* (in this case, 0.4).

For the case when *ts* is 20 times higher, the airtime consumption of the end device is accumulated over a large period and the feedback on airtime usage is sent relatively late. Meanwhile, if the end device exceeds or subceeds the allocated usage, it will be reduced or increased after the late feedback due to the large *ts*. Additionally, the *cwnd* value is significantly decreased or increased based on the

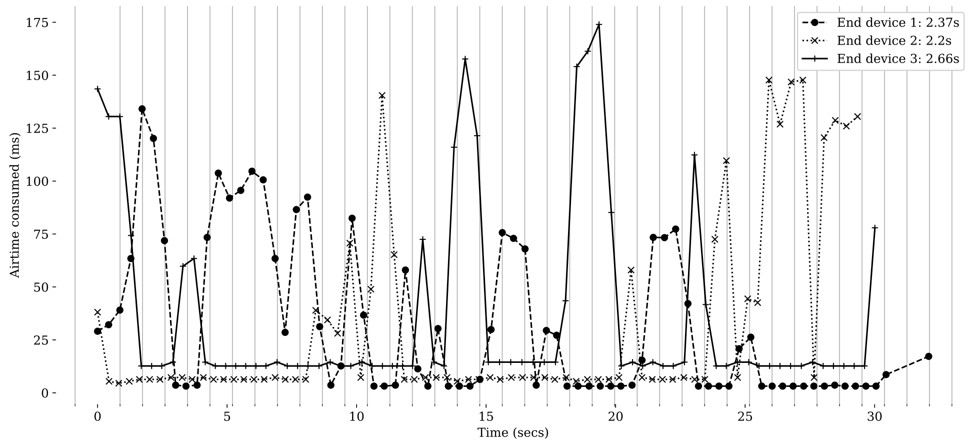
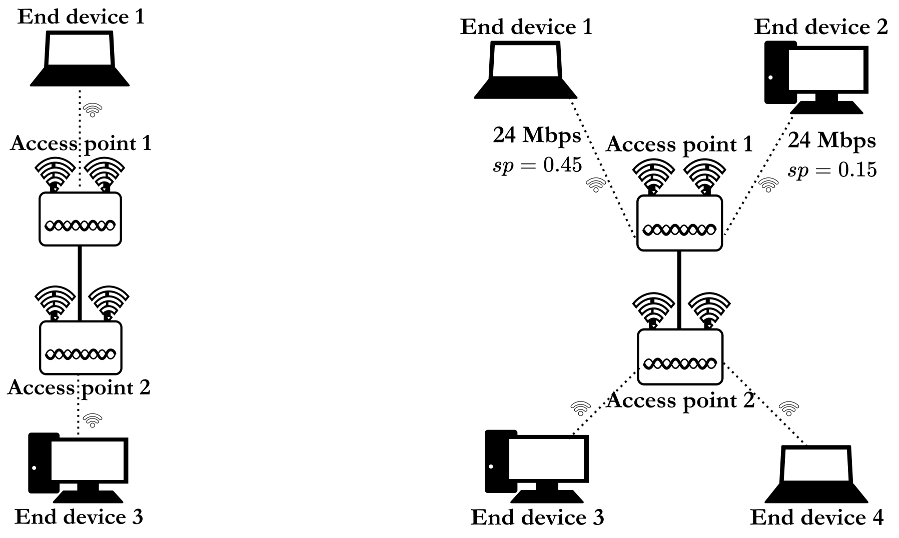


FIGURE 12. Instantaneous airtime consumption of the FCCC algorithm on sending data for 30 secs (minor grid lines indicate *ts*).



(a) Wireless network setup to validate the robustness of the FCCC algorithm.

(b) Wireless network setup to test the use case of varying sp value.

FIGURE 13. Wireless network setup for test cases D and E.

feedback and large *ts* value and has a negative impact over the next *ts*. Similarly, the cycle continues over the next timeslots and the throughput of the end device is either too high with several packet losses or too low with a very low data transfer rate. The application data transfer rate will always remain in the extremes as shown in Fig. 5. In Fig. 6, the inital *ts* value was approximately 6 secs and the airtime consumption of the end device is accumulated till the 6th second and the device receives the feedback to reduce the window size extensively. This resulted in large amount of data in the network thus increasing the RTT, therefore the *ts* was increased to 22 secs. On reducing the size, though the device underuses the timeslot since the timeslot is very large, the feedback is not received immediately.

C. BENCHMARKING AGAINST TRADITIONAL CC ALGORITHM

The performance of the FCCC algorithm was benchmarked against the CUBIC CC algorithm which is traditionally used by default in all the networking devices. The KPIs used to validate the performance of FCCC are improvement in the throughput and airtime fairness between the wireless devices. To measure these values, the tests were conducted in a wireless network setup in Fig. 4 with three end devices, End device 1, End device 2, and End device 3, with physical data rates fixed to 24 Mbps, 12 Mbps, and 6 Mbps respectively acting as sender end devices. The other end devices End device 4, 5, and 6 act as receiver end devices.

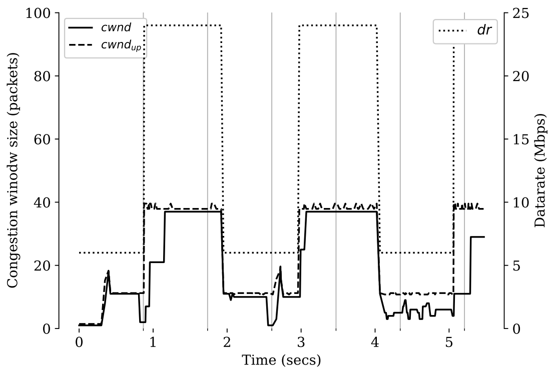


FIGURE 14. The $cwnd_{up}$ and *cwnd* for changing physical data rates (minor grid lines indicate *ts*).

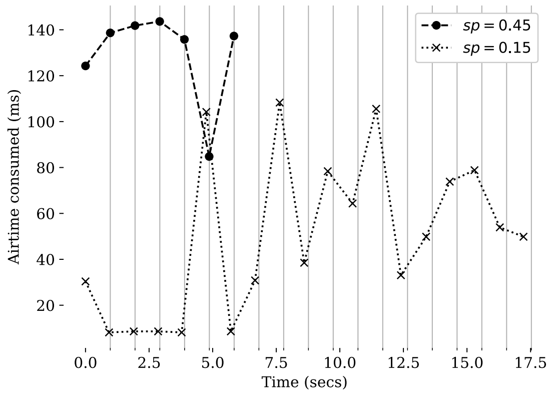


FIGURE 15. Instantaneous airtime consumption for different *sp* values (minor grid lines indicate *ts*).

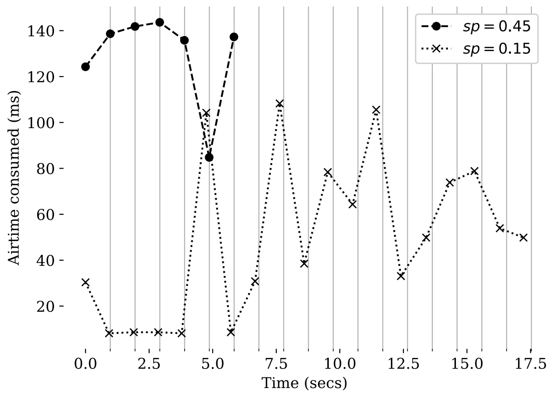


FIGURE 16. Results of *cwnd* for different *sp* values from AP.

To evaluate the improvement in throughput, a fixed payload of 5 megabytes (MB) of data was sent from End devices 1, 2, and 3. The measured average throughput values of all the end devices using FCCC and CUBIC algorithms are plotted in Fig. 7. The percentage of average throughput improvement per end device when using the FCCC algorithm as compared to CUBIC CC is also measured and plotted in Fig. 8. From the plots it can be noticed that all the end devices have the same throughput while using the CUBIC CC algorithm, irrespective of the physical data rates of the end devices. Whereas in the case of FCCC, the End device 1 with a higher physical data rate has 58% better throughput as compared to the CUBIC and other end devices. Though the throughput of End device 3 is 43% lower while using FCCC, the overall efficiency of the network bandwidth is the same for both algorithms. From the instantaneous airtime consumption plot in Fig. 9, it can be noticed that along with the improved throughput, FCCC also achieves airtime fairness among the end devices. This is achieved by utilizing *cwnd* accordingly as per the FCCC algorithm, as shown in Fig. 10. On starting the TCP connection simultaneously in all the end devices, the End device 1 has higher *cwnd* at 0th second. Once the data transfer of End device 1 is ended at arund 13th second, the *cwnd* values of other end devices are updated accordingly. When all the end devices are active, we can see an unequal cyclic pattern in the application data transfer rate resulting in equal airtime usage.

To measure the improvement in airtime consumption of the end devices with higher physical data rates, the test was conducted where all the end devices continuously sent data for 30 seconds for FCCC and CUBIC algorithms. The percentage difference between the average airtime consumed by the end devices is plotted in Fig. 11. Results indicate that in the case of CUBIC, the End device 1 with a lower physical data rate is dominating in airtime consumption, therefore the percentage of the average difference in airtime consumption among the end devices is higher than 100% and negative in the case of CUBIC. In contrast, the FCCC aims at airtime fairness, thus all the end devices consume equal airtime with a maximum difference of 18%. A similar effect can be seen in the instantaneous airtime consumption of the end devices plotted in Fig. 12.

In summary, the results prove that using FCCC, the throughput of end devices with higher physical data rates is better in comparison to the throughput from using the CUBIC algorithm. This is the outcome of airtime fairness achieved by FCCC among the end devices. Though there is a small degradation in the throughput of those devices with lower physical data rates this is the penalty that end devices with low data rates must pay to achieve airtime fairness and improve the overall performance of higher physical data rates.

D. ROBUSTNESS OF FCCC ALGORITHM TO CHANGING DATA RATES

The robustness of the designed FCCC algorithm to changing physical data rates was tested in a wireless setup shown in Fig. 13(a). The physical data rate of End device 1 is switched between 24 Mbps and 6 Mbps every second. The End device 3 acts as receiver end device. The behavior of *cwnd* and $cwnd_{up}$ for varying physical data rates are the KPIs for this test and the outcome is plotted in Fig. 14. $cwnd_{up}$ is dependent on the data rate and acts as an upper bound to the congestion window size. Therefore, when the physical data rate goes down at 2nd second in Fig. 14, the $cwnd_{up}$ instantly goes down according to the physical data rate and thus the data transfer rate *cwnd*. Similar effects can be seen when the physical data rate goes up at 3rd second. The robustness of FCCC helps in maintaining airtime fairness in case of improvement or deterioration in the physical data rate and thus avoiding the biased decision at the AP among the end devices.

E. USE CASES: DIFFERENT SP VALUES FROM AP

The *sp* value shared by AP has a significant impact on deciding the application data transfer. Varying the *sp* values for different end devices adds additional functionality of priority-based airtime fairness to the designed network architecture. Tests were conducted in the wireless network setup in Fig. 13(b), while sending a fixed amount of data from each end device, to see the behavior of application data transfer rates and the airtime consumption when different *sp* values are sent as feedback to different end devices. End device 1 has fixed physical data rate of 24 Mbps and *sp* value of 0.45 and the End device 2 has a fixed physical data rate of 24 Mbps and *sp* value of 0.15. End devices 3 and 4 act as receiver end devices. The measured instantaneous airtime consumption and the behavior of application data transfer rate i.e. *cwnd* being the KPIs, are plotted in Fig. 15 and 16 respectively. Results indicate that for the *sp* value of 0.45, the *cwnd* value is higher, and thus the airtime consumption of End device 1. The total airtime consumption of the end device with *sp* value 0.45 is three times greater than that of the one with *sp* value 0.15. Unlike the previous cases, after the end of the data transfer of End device 1, the *cwnd* value of the other end device doesn't increase significantly, and the device keeps the airtime consumption as low as possible because of the feedback of 0.15 *sp* value from AP. This experiment has proven that priority-based airtime fairness can be implemented by AP by varying the *sp* values. This opens the door for use in multiple application scenarios.

Even with the increase in the number of nodes, each end device will get a certain amount of airtime allocated by the AP, and in the worst-case scenario, the algorithm is designed to send at least $cwnd_{init}$ number of packets to sustain the connection and avoid starvation. Also, AP monitors the airtime consumption of each end device and hence the end device that is already registered but has not been sending packets in each timeslot will be given an opportunity and the other end devices will be notified to reduce the application data transfer rate to balance the airtime fairness. The transport layer buffers are designed to be large enough to cater to the TCP needs such as re-transmissions and adjusting to the receiver buffer size. Hence no significant buffer overflow situations are noticed in transport layer buffers. Since the number of packets reaching the MAC layer of the device is monitored by the transport layer, the MAC layer buffer does not experience excessive packet queuing or buffer overflow.

VIII. CONCLUSION AND FUTURE WORK

Wi-Fi is an essential tool for connectivity in today's and future private professional networks, offering various data rates based on channel quality. Despite Wi-Fi protocol advancements, throughput degradation remains a problem for higher data rate devices when lower data rate devices are present, due to airtime unfairness. Emerging applications with diverse requirements highlight the need for network protocols serving these demands. Innovations like INT, APP-NET, and eBPF allow real-time monitoring, but CC algorithms, designed for wired networks, need adaptation for efficient data transfer in wireless networks. Therefore, there is a need for a suitable algorithm to determine optimal data transfer rates without sacrificing throughput in the presence of lower data rate devices in wireless networks. In this work, we address this problem by designing a feedback-based control-loop CC algorithm, FCCC. The system implements additional bookkeeping of the airtime consumption of each end device in AP, which is further added to the INT header as aggregated airtime feedback to each end device. The FCCC algorithm considers the real-time aggregated airtime feedback from AP and network state information obtained through INT to decide the suitable congestion window size. The designed algorithm was benchmarked against the existing CUBIC CC algorithm, whereby using the FCCC the end device with a higher physical data rate had a 58% improvement in the throughput as compared to using the CUBIC algorithm without compromising the overall network efficiency. The airtime difference between the end devices was as low as 18%. The algorithm was validated for its sensitivity to the varying timeslot length, and robustness to the changing physical data rate. A potential use case of priority-based airtime fairness was also tested for future use cases.

The implemented FCCC algorithm improves the throughput of the higher physical data rate devices in the presence of lower physical data rates in wireless networks. The additional features in the algorithm and the monitoring and feedback system in AP, enable priority-based airtime fairness in wireless networks and are an outlet for multiple such use cases that can be explored in the future. Another aspect that is out of the scope of this work, but a potential future work is the division of wireless airtime among different flows in an end device and the scalability of the designed algorithm. The myriad data collected during the design and experimentation of this work can be used in the future to study and design adaptive application layer protocols.

REFERENCES

[1] J. Haxhibeqiri, P. H. Isolani, J. M. Marquez-Barja, I. Moerman, and J. Hoebeke, "In-band network monitoring technique to support SDN-based wireless networks," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 1, pp. 627–641, Mar. 2021.

[2] J. Haxhibeqiri, A. Seferagic, R. V. Bhat, I. Moerman, and J. Hoebeke, "Tighter application-network interfacing to drive innovation in networked systems," in *Proc. ACM SIGCOMM Workshop Netw.-Appl. Integr.*, Aug. 2021, pp. 53–57.

[3] A. M. Abdul-Hadi, O. Tarasyuk, A. Gorbenko, V. Kharchenko, and T. Hollstein, "Throughput estimation with regard to airtime consumption unfairness in mixed data rate Wi-Fi networks," *Commun. Sci. Lett. Univ. Zilina*, vol. 16, no. 1, pp. 84–89, Feb. 2014.

[4] F. Peng, B. Peng, and D. Qian, "Performance analysis of IEEE 802.11e enhanced distributed channel access," *IET Commun.*, vol. 4, no. 6, p. 728, 2010.

[5] F. Safdari and A. Gorbenko, "Experimental evaluation of performance anomaly in mixed data rate IEEE802.11ac wireless networks," in *Proc. 10th Int. Conf. Dependable Syst., Services Technol.*, Jun. 2019, pp. 82–87.

[6] F. Safdari and A. Gorbenko, "Theoretical and experimental study of performance anomaly in multi-rate IEEE802.11ac wireless networks," *Radioelectronic Comput. Syst.*, no. 4, pp. 85–97, Nov. 2022.

[7] R. V. Bhat, J. Haxhibeqiri, I. Moerman, and J. Hoebeke, "Adaptive transport layer protocols using in-band network telemetry and eBPF," in *Proc. 17th Int. Conf. Wireless Mobile Comput., Netw. Commun.*, Oct. 2021, pp. 241–246.

[8] R. V. Bhat, J. Haxhibeqiri, I. Moerman, and J. Hoebeke, "Network- and application-aware adaptive congestion control algorithm," *J. Commun. Netw.*, vol. 26, no. 3, pp. 344–355, Jun. 2024.

[9] A. Narayan, F. Cangialosi, D. Raghavan, P. Goyal, S. Narayana, R. Mittal, M. Alizadeh, and H. Balakrishnan, "Restructuring endpoint congestion control," in *Proc. Appl. Netw. Res. Workshop*, Jul. 2018, pp. 30–43.

[10] G. Bianchi, L. Fratta, and M. Oliveri, "Performance evaluation and enhancement of the CSMA/CA MAC protocol for 802.11 wireless LANs," in *Proc. PIMRC 96 - 7th Int. Symp. Pers., Indoor, Mobile Commun.*, 1996, pp. 392–396.

[11] J. Hoblos, "New adaptive 802.11 MAC protocol to enhance throughput and fairness in multihop wireless networks," in *Proc. Int. Wireless Commun. Mobile Comput. (IWCMC)*, Jun. 2021, pp. 1908–1913.

[12] J. Kim, H. Jin, J.-B. Seo, S. H. Kim, and D. K. Sung, "Adaptive transmission control for uplink/downlink fairness in unsaturated CSMA networks," *IEEE Commun. Lett.*, vol. 23, no. 10, pp. 1879–1882, Oct. 2019.

[13] A. Baiocchi, "Maximizing the stable throughput of heterogeneous nodes under airtime fairness in a CSMA environment," *Comput. Commun.*, vol. 210, pp. 229–242, Oct. 2023.

[14] J. Lei, J. Tao, J. Huang, and Y. Xia, "A differentiated reservation MAC protocol for achieving fairness and efficiency in multi-rate IEEE 802.11 WLANs," *IEEE Access*, vol. 7, pp. 12133–12145, 2019.

[15] S. I. Yu and C. Y. Park, "A responsible airtime approach for true time-based fairness in multi-rate WiFi networks," *Sensors*, vol. 18, no. 11, p. 3658, Oct. 2018.

[16] Z. Mao and Y. Jiang, "Fair airtime allocation for content dissemination in WiFi-direct-based mobile social networks," in *Proc. IEEE 28th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, Oct. 2017, pp. 1–7.

[17] Y. Fang, B. Doray, and O. Issa, "A practical air time control strategy for Wi-Fi in diverse environment," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, Mar. 2017, pp. 1–6.

[18] T. Høiland-Jørgensen, M. Kazior, D. Taht, P. Hurtig, and A. Brunstrom, "Ending the anomaly: Achieving low latency and airtime fairness in WiFi," in *Proc. USENIX Annu. Tech. Conf.*, Santa Clara, CA, USA, 2017, pp. 139–151.

[19] D. J. Kulenkamp and V. R. Syrotiuk, "Support for differentiated airtime in wireless networks," in *Proc. IEEE Conf. Comput. Commun. Workshops*, May 2021, pp. 1–6.

[20] T. Liubov and Y. Krasnozheniuk, "Analysis of airtime fairness technology application for fair allocation of time resources for IEEE 802.11 networks," in *Proc. Int. Sci.-Practical Conf.*, 2022, pp. 635–655.

[21] C. Ben Ameur, E. Mory, and B. Cousin, "Combining traffic-shaping methods with congestion control variants for HTTP adaptive streaming," *Multimedia Syst.*, vol. 24, no. 1, pp. 1–18, Feb. 2018.

[22] J. Jayaudhaya, S. Supriya, V. A. Kandaswamy, S. Kamatchi, and C. P. Priya, "ACoCo: An adaptive congestion control approach for enhancing CoAP performance in IoT network," in *Proc. 2nd Int. Conf. Appl. Artif. Intell. Comput.*, May 2023, pp. 189–194.

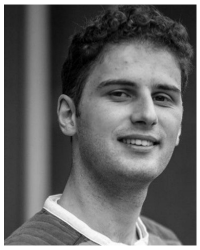
[23] T. K. Mishra, K. S. Sahoo, M. Bilal, S. C. Shah, and M. K. Mishra, "Adaptive congestion control mechanism to enhance TCP performance in cooperative IoV," *IEEE Access*, vol. 11, pp. 9000–9013, 2023.

[24] I. Rhee, L. Xu, S. Ha, A. Zimmermann, L. Eggert, and R. Scheffenegger, *CUBIC for Fast Long-Distance Networks*, document RFC 8312, 2018.

[25] S. Patel, Y. Shukla, N. Kumar, T. Sharma, and K. Singh, "A comparative performance analysis of TCP congestion control algorithms: Newreno, westwood, veno, BIC, and cubic," in *Proc. 6th Int. Conf. Signal Process. Commun. (ICSC)*, Mar. 2020, pp. 23–28.

[26] J. Haxhibeqiri, R. V. Bhat, I. Moerman, and J. H. Idlab, "Age-of-Information aware in-band network telemetry for better network predictability," in *Proc. 17th Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, Oct. 2021, pp. 42–47.

RAMYASHREE VENKATESH BHAT received the B.E. degree in electronics and communications engineering from the PES Institute of Technology (now PES University), Bengaluru, India, in 2017, and the M.Sc. degree in communication systems from the Technical University of Munich, Munich, Germany, in 2020. She is currently pursuing the Ph.D. degree in engineering computer science with Ghent University, Ghent, Belgium.



JETMIR HAXHIBEQIRI (Member, IEEE) received the master's degree in engineering, information technology, and computer engineering from RWTH Aachen University, Aachen, Germany, in 2013, and the Ph.D. degree in engineering computer science from Ghent University, Ghent, Belgium, in 2019. He is currently a Postdoctoral Researcher with the Internet Technology and Data Science Laboratory, Ghent University, and IMEC, Leuven, Belgium. His research interests include wireless communications technologies (IEEE 802.11, IEEE 802.15.4e, LoRa) and their application, the IoT, wireless time-sensitive networking, in-band network monitoring, and wireless network management.

INGRID MOERMAN (Senior Member, IEEE) received the degree in electrical engineering, in 1987, and the Ph.D. degree from Ghent University, in 1992. She became a part-time Professor with Ghent University, in 2000. She is currently a Staff Member with IDLab, Core Research Group, imec, with research activities embedded with Ghent University and the University of Antwerp. Her research interests include cooperative and intelligent radio networks, real-time software-defined radio, time-sensitive networks, dynamic spectrum sharing, coexistence across heterogeneous wireless networks, vehicular networks, open-source prototyping platforms, software tools for programmable networks, next-generation wireless networks (5G/6G), and experimentally supported the research.

JEROEN HOEBEKE is currently an Associate Professor with the Internet Technology and Data Science Laboratory, Ghent University–imec. He is conducting and coordinating research on wireless (IoT) connectivity, embedded communication stacks, deterministic wireless communication, and wireless network management. This expertise has been applied in a variety of application domains, such as logistics, industry 4.0, building automation, healthcare, and animal monitoring. He is particularly active in national funded projects as well as in defining, executing, and managing such projects. He has also been involved in several EU research funded projects and is the author or co-author of more than 150 publications in international journals or conference proceedings.