

In-band Network Telemetry-based Congestion Control Algorithm for Industrial Wireless Networks

Ramyashree Venkatesh Bhat, Jetmir Haxhibeqiri, Ingrid Moerman, Jeroen Hoebeke
IDLab, Ghent University – imec, Ghent, Belgium
Email: [name.surname]@ugent.be

Abstract—Recent technologies of Wi-Fi are being widely incorporated in smart factories with an expectation of real-time, flexible, reliable, and quality services. The current Wi-Fi offers a range of physical data rates based on channel quality but is limited to certain applications. With the traditional application and transport layer protocols being designed for wired networks, using them in bottleneck situations of industrial wireless networks poses certain drawbacks and one such drawback is the throughput degradation due to airtime unfairness. Using innovations like in-band network telemetry, application-network interaction, and the programmability of the kernel and intermediate devices, a reactive airtime feedback-based congestion control algorithm, RACC, for wireless networks is introduced in this paper. The designed algorithm reactively adapts the application data transfer rate based on the feedback from the access point and was implemented on commercial off-the-shelf devices. Using this algorithm, the throughput was improved 72% and the airtime unfairness was reduced to as low as 15% as compared to the CUBIC. The designed congestion control algorithm addresses the issue of airtime unfairness and throughput degradation in industrial wireless networks based on real-time network state.

Index Terms—airtime fairness, congestion control algorithm, industrial wireless networks, cross-layer interaction

I. INTRODUCTION

IEEE 802.11, known commercially as Wi-Fi, is one of the crucial tools for connectivity in smart factories. Industries are moving towards wider incorporation of Wi-Fi, and the protocols are expected to provide real-time, reliable, and quality data transfer services¹. They must also address some of the existing wireless challenges like flexibility, interference, and reliability. Unlike the present Wi-Fi protocols being limited to a few applications, the future Wi-Fi protocols are expected to be flexible and cater to the needs of a wide variety of industrial applications.

The channel access mechanism for all IEEE 802.11 standards is based on Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) to avoid collisions between different devices. It provides pseudo-random fairness in channel access between different devices. The latest Wi-Fi standard, IEEE 802.11ax, supports a wide range of physical data rates (from 8.6 Mbps to 9.6 Gbps)², giving the freedom to devices to select the physical data rate based on the channel quality. As the network is composed of devices experiencing different channel quality, this results in unfairness of airtime usage

between devices. Though the recent Wi-Fi technologies offer significantly higher data throughput, the presence of a low physical data rate device brings down the throughput of a high physical data rate device. In a Wi-Fi network, all the devices have equal throughput irrespective of their physical data rate and the throughput is equal to the throughput of the device with the least physical data rate [1]. Even with Enhanced Distributed Channel Access (EDCA), there is airtime unfairness within the same Quality of Service (QoS) class. The Transmit Opportunity (TXOP) feature of 802.11 is supported only for a few access categories and is applied to all the flows belonging to the same access category. When the TXOP is enabled in all the end devices, the flows of the same access category end up facing airtime unfairness during the presence of an end device with a lower physical data rate. Therefore, it is essential to address the issue of airtime unfairness in industrial wireless networks.

Transmission control protocol (TCP) is a widely used transport protocol as it provides ordered, reliable, and error-free data transfer utilizing its congestion control and flow control algorithms. However, the congestion control algorithms used in TCP do not cope well with wireless links [4]. Innovations such as Application-Network Integration (APP-NET) [2] and In-band Network Telemetry (INT) [3] have enabled the possibility of applications communicating their requirements to the network, whilst learning the current network state and performing continuous, low-overhead, network monitoring respectively. This brings application-transport-network layer interactions where the transport layer becomes aware of applications and real-time network conditions, in addition to end-to-end network information. Utilizing extended Berkeley Packet Filter (eBPF), modifications on existing transport layer protocols are possible [5] [6].

Utilizing these innovations, the issue of airtime unfairness leading to throughput degradation, and lack of congestion control algorithms suitable for industrial wireless networks is addressed by designing a reactive airtime feedback-based congestion control (RACC) algorithm. The designed algorithm immediately reacts to the airtime feedback from the access point (AP) by adapting the application data transfer rate, hence the name. The design was implemented in commercial off-the-shelf (COTS) devices. The performance of the designed algorithm was validated in terms of throughput, airtime consumption, and efficiency and is proven to achieve airtime fairness among the end devices in a mixed physical data rate

¹<https://iebmedia.com/technology/industrial-5g-wireless/industrial-wireless-technology-2022-special-report/>

²<https://www.intel.com/content/www/us/en/gaming/resources/wifi-6.html>

wireless network.

The rest of the paper is structured as follows: Section II discusses the existing works in the field of airtime fairness; Section III discusses the challenges and problem statement addressed in the paper; Section IV explains the design and implementation of the RACC algorithm. Section V consists of results and observations. The conclusions and future work are addressed in Section VI.

II. RELATED WORK

This section discuss different existing research works that focus on achieving airtime fairness in wireless networks and adaptive congestion control algorithms.

In [7], the author has designed an adaptive transmission algorithm that maximizes the throughput achieved by allowing each station to adjust its transmission probability, under airtime fairness constraints. The designed algorithm maximizes the throughput by adjusting the contention window size of CSMA/CA. The authors of [8] have designed a distributed resource allocation protocol called REACT which negotiates the airtime among nodes in a wireless network. This protocol is implemented for an ad hoc setting, where nodes bid their required airtime. The design has been extended for differentiated airtime. In [9], the authors have exploited the access point to influence the behavior of connected nodes. The access point communicates the changes in the scheduling of transmission opportunities of different nodes, thus achieving airtime fairness. The same has been extended to the PoliFi system which acts as an access point policy enforcer for various instances like prioritization of nodes, load balancing, and performance improvement for streaming applications. The authors of [10] have designed a new queuing scheme based on the FQ-CoDel queue management system, to eliminate the buffer bloat in the wireless networks. Utilizing this scheme, the authors have designed an airtime fairness scheduler operating in an access point.

The authors of [11] have combined two traffic shaping methods, the Hierarchical Token Bucket shaping Method and the Receive Window Tuning Method, and four congestion control algorithms, NewReno, Vegas, Illinois, and CUBIC to improve the quality of experience of HTTP adaptive streaming in the same home network. In [12], the authors have designed an adaptive congestion control approach for CoAP (ACoCo), which uses a reinforcement learning-based congestion control algorithm to reduce end-to-end transaction delays in 6TiSCH networks. The authors of [13] have designed a novel congestion control algorithm that operates on one-bit signaling feedback from the base station/router to achieve strict queuing delay bounds in dynamic cellular network environments. In [14] authors have implemented a cross-layer technique for Internet of Vehicles networks that receives information on the status of buffer space and link utilization in ACK packet from the receiver, which is used by the sender to modify the application data transfer rate. The designed approach has improved buffer utilization, link utilization, throughput, and packet loss.

The above-mentioned works achieve airtime fairness either by modifying the MAC layer contention window size or by enforcing different scheduling policies in the AP. Though the MAC layer contention window size modification may seem like an ideal method, implementing the design in COTS devices is challenging. In case of scheduling policies in the APs, when the number of flows in the network increases, more than one node have to use the same schedule, resulting in airtime unfairness. The implemented adaptive congestion control algorithms in the above works are specific to different use cases like video streaming and network environments like 6TiSCH. In this paper, the transport layer congestion window size is dynamically adjusted based on the INT feedback from APs. The design is implemented in COTS devices by using the recent innovations in the field of wireless networking.

III. CHALLENGES AND PROBLEM STATEMENT

This section gives an insight into the challenges of wireless networks and the problem statement of this paper.

A. Challenges

One of the reasons for industrial networks moving towards Wi-Fi is higher data throughput in the recent versions. The latest Wi-Fi technology offers physical data rate as high as 9.6Gbps. Though the device may have a higher physical data rate, the actual throughput of the data transfer is eventually decided by the channel quality and other competing devices in the wireless network. The existing channel access mechanism hinders the throughput of faster devices in a wireless network. Since all the devices in the network have a fair chance to access the channel, a low physical data rate device consumes more airtime of the network. This makes the high physical data rate device decrease the overall throughput to the level of the throughput of the low physical data rate device. Given a scenario of a wireless network with two devices operating on 2.4Gbps and 8.6Mbps, the approximate airtime consumed to transfer a payload of 10MB is equal to 0.033s and 9.3s respectively, and the resulting throughput of both the devices is 8.569Mbps.

The aim of the congestion control algorithm is to decide the right congestion window size to achieve higher throughput whilst avoiding network congestion. The existing algorithms are designed to exploit the performance of high-bandwidth wired links. Along with network congestion, wireless networks pose additional challenges because of lossy medium, thick walls, mobility, and interference in industrial environments. Hence it is essential to continuously monitor the network state to achieve the intended performance. The congestion window size determined by the transport layer protocol is the number of packets through which the application layer data is sent for wireless transmission. Therefore, in addition to the physical data rate, channel quality, and the number of other end devices in the network, the congestion window size decided by the congestion control algorithm also influences the airtime utilized. Hence, the issue of throughput degradation can be tackled by a better design of the transport protocol's

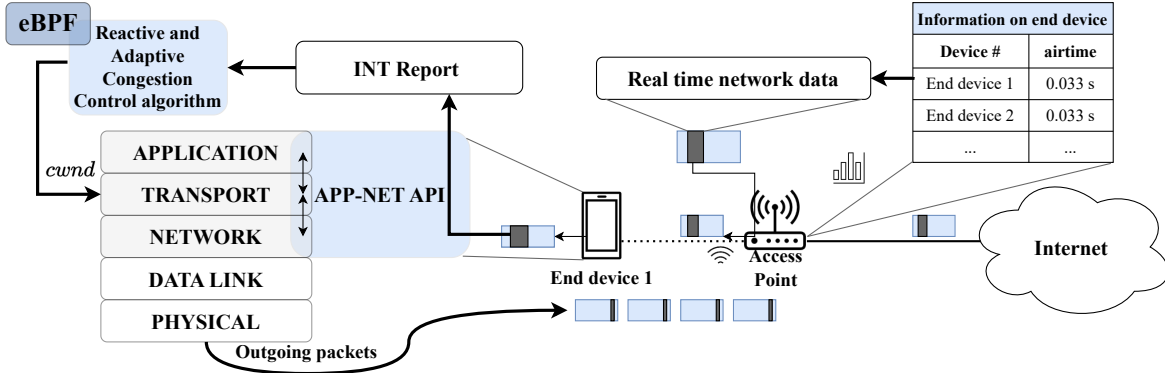


Fig. 1: The block diagram of implemented wireless system with underlying frameworks.

congestion control algorithm which can give distinguished treatment of the traffic flows from devices with different airtime usage.

B. Problem statement

This paper aims to solve the issue of airtime unfairness in industrial wireless networks by designing a congestion control algorithm that adapts the application data transfer rates by reacting to the feedback from the network and AP. The performance of the designed algorithm is validated by evaluating the improvement in the throughput of a high physical data rate device when the RACC algorithm is used. The difference in the airtime consumption of a high physical data rate device when the RACC algorithm is used, as compared to the CUBIC congestion control algorithm, is also verified.

IV. UNDERLYING FRAMEWORKS AND MONITORING PARAMETERS

This section explains the underlying frameworks that facilitate the design of the RACC algorithm. Then discuss the essential monitoring parameters that are obtained as feedback from AP through INT.

A. Underlying frameworks

The design of the reactive airtime feedback-based congestion control algorithm is based on three concepts that have been recently presented in literature. The Fig. 1 indicates the workflow of implemented design along with underlying frameworks. This subsection gives a short explanation of each of the such enablers.

- 1) For the applications to react to the changes in the network it is essential to continuously receive real-time feedback from the network. For this, the algorithm is built upon the idea proposed by the authors of [3] regarding a multi-plane design capable of performing real-time monitoring. The wireless INT is a low overhead, real-time monitoring that enhances data packets with monitoring data on the fly. Monitoring data include end-to-end information and per-hop information such as physical data rate, number of flows in the path, received signal strength (RSSI), etc.

- 2) In the current OSI model, application and transport layers only have an end-to-end view of the communication and are uninformed of the underlying network. The APP-NET API designed by the authors of [2] has facilitated the interaction between application, transport, and network layers, resulting in application-transport-network layer interaction. Hence, this framework is used to communicate the application requirements from the application to the network layer as well as the network layer information to the application and transport layers of OSI model.
- 3) The congestion control plane (CCP) API, designed by researchers at MIT, is a framework based on eBPF, that allows the reconfiguration of congestion control algorithms [15]. This framework is used to implement the reactive airtime feedback-based congestion control algorithm.

B. Monitoring parameters

The key network parameters that directly affect the airtime consumption of the end devices are physical data rate (*data_rate*, unit: Mbps) and the number of end devices in the wireless network (*device_count*, unit: number of end devices). The packet loss in an industrial wireless network is not only caused by network congestion but can be due to wireless conditions like interference. Hence, to differentiate the packet losses due to wireless conditions, the per-hop counter value (terminology from [3]) is also an essential parameter.

Since the AP has an overall view of the connected end devices, the designed algorithm depends on the feedback from the AP. To achieve airtime fairness, the AP maintains a table of airtime allocated to each end device, by calculating the *airtime_i* (unit: second/device) of each end device using Equation 1:

$$airtime_i = \frac{timeslot}{device_count} \quad (1)$$

where *timeslot* (unit: second) is fixed by the AP and gives the period over which the AP calculates the airtime allocation per end device, and *device_count* indicates the number of end devices in the network seen by the AP. The *airtime_i*

indicates the amount of airtime that can be utilized by the device i . With the increase in the number of end devices, the airtime allocated to each device is decreased, therefore the $airtime_i$ is inversely proportional to and dependent on the number of end devices. Since the aim is to achieve airtime fairness, the equation does not have any weighted component, and the given airtime is equally distributed among different flows.

Therefore the monitoring parameters that are sent as INT feedback are $data_rate$, $wireless_loss$, and $airtime_i$.

V. REACTIVE AIRTIME FEEDBACK-BASED CONGESTION CONTROL ALGORITHM

This section describes the design and implementation of the reactive airtime feedback-based congestion control algorithm.

A. Design of RACC algorithm

This paper aims to achieve airtime fairness among the end devices by adapting the application data transfer to the network and AP feedback. $data_rate$, $wireless_loss$, and $airtime_i$ are the monitoring parameters received as feedback from AP by the end device. For a given $data_rate$ and allotted $airtime_i$, the number of packets that can be sent for each $timeslot$, i.e., the congestion window size ($cwnd$, unit: packet) is given by the Equation 2:

$$cwnd = \lfloor \frac{(data_rate * airtime_i)}{MSS * 8} \rfloor \quad (2)$$

The multiplication of $data_rate$ and $airtime_i$ gives the amount of payload, and dividing it by maximum segment size (MSS) gives the number of packets that can be transferred for the given $timeslot$. The obtained $cwnd$ equation is dependent on $data_rate$ and $airtime_i$ feedback from AP. Since the $cwnd$ is directly proportional to the $data_rate$, the high physical data rate device sends more packets during each data transfer as compared to the low physical data rate device. The increase in the number of end devices in the network, decreases the $airtime_i$, thus decreasing the number of packets sent at each data transfer by the device. Thus the designed algorithm aims to achieve airtime fairness by making the application data transfer rate dependent on the physical data rate and the changing network conditions.

B. Implementation of RACC algorithm

The designed RACC algorithm is represented graphically in Algorithm 1. On starting a new TCP connection, the RACC algorithm first enters the *Initialization* phase where it receives the value of total *payload* length (unit: byte) to be sent, via APP-REQ interface. It sets the value of $init_cwnd$ (unit: packet), i.e., the $cwnd$ of the first data transfer, and β , where β is a $cwnd$ decrease factor after packet loss event. Once the initialized values are set the $reset_function()$ is called as shown in Algorithm 1. The value of $init_cwnd$ is set to 1, and the device sends an synchronize (SYN) packet, where the $cwnd$ is set to $init_cwnd$.

On receiving this SYN packet, the AP registers the device and increases the $device_count$. The AP then calculates the

Algorithm 1 The designed RACC algorithm

```

1: Initialization:
2: Value updated from APP_REQ:  $payload$ ,  $\beta \leftarrow 0.9$ ,
3:  $init\_cwnd \leftarrow 1$ ,  $cwnd \leftarrow init\_cwnd$ ,
4:  $cwnd\_out \leftarrow MSS * cwnd$ ,  $reset\_function()$ 
5:
6: on_report:
7: Values updated from INT feedback:
8:  $wireless\_loss$ ,  $airtime_i$ ,  $data\_rate$ 
9: Values reported by eBPF:
10:  $bytes\_acked$ ,  $bytes\_sacked$ ,
11:  $packet\_loss$ ,  $flow\_timeout$ ,  $MSS$ 
12: if  $bytes\_acked$  or  $bytes\_sacked$  then:
13:    $on\_ack()$ 
14: else if  $packet\_loss$  then:
15:    $on\_loss()$ 
16: else if  $flow\_timeout$  then:
17:    $on\_timeout()$ 
18: end if
19:  $on\_ack()$ :
20: if  $airtime \neq 0$  then:
       $cwnd \leftarrow \lfloor \frac{(data\_rate * airtime_i)}{MSS * 8} \rfloor$ 
21: else:
22:    $cwnd \leftarrow init\_cwnd$ 
23: end if:
24:  $cwnd\_out \leftarrow MSS * cwnd$ 
25:
26:  $on\_loss()$ :
27: if  $wireless\_loss$  then:
28:    $cwnd \leftarrow cwnd$ 
29: else:
30:    $cwnd \leftarrow cwnd * \beta$ 
31: end if
32:  $cwnd\_out \leftarrow MSS * \max(cwnd, init\_cwnd)$ 
33:
34:  $flow\_timeout():reset\_function()$ 
35:
36:  $reset\_function()$ :
37:  $wireless\_loss \leftarrow 0$ ,  $airtime_i \leftarrow 0$ ,  $data\_rate \leftarrow 0$ ,
38:  $bytes\_acked \leftarrow 0$ ,  $bytes\_sacked \leftarrow 0$ ,  $packet\_loss \leftarrow 0$ 

```

$airtime_i$ of the current device using equation (1). For experimental purposes, the $timeslot$ is fixed to 0.066 s (explained in Section VI-A). The AP reports the monitoring parameters in Section IV-B as INT header. This information is received by the device as feedback with the synchronize-acknowledgment (SYN-ACK) packet. This $airtime_i$ feedback is further sent at every $timeslot$, hence updating the device with ongoing changes in the network, for example beginning of a new connection from an end device.

If the connection is successful, the end device receives a SYN-ACK packet. The RACC algorithm then enters the *on_report* function through which it gets the information

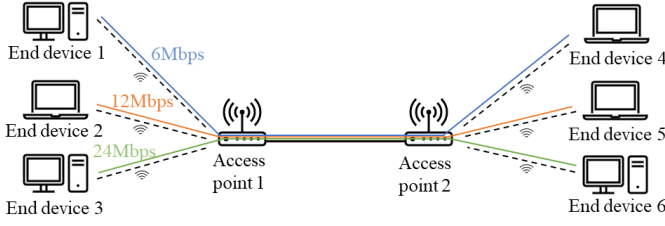


Fig. 2: Wireless network setup

on $airtime_i$ allotted and other wireless network parameters. The algorithm also receives feedback about the end-to-end parameters of the ongoing connection such as $bytes_acked$ (total bytes acknowledged, unit: byte), $bytes_sacked$ (total misordered bytes sent, unit: byte), $packet_loss$ (total number of packets lost, unit: packet) and $flow_timeout$ (if the TCP connection of the flow itself was lost due to TCP timeout, Boolean variable) from Linux kernel using eBPF.

If the device receives an acknowledgment (ACK) packet, the RACC algorithm enters the $on_ack()$ function. The number of packets that can be sent after each ACK packet is calculated using equation (2). If there is no $airtime_i$ feedback, the $cwnd$ value is set to $init_cwnd$ to sustain the TCP connection. The same function is executed on receiving every ACK packet.

If the RACC algorithm receives a notification of $packet_loss$, the algorithm enters the $on_loss()$ function. It first checks if the packet loss is due to the wireless medium ($wireless_loss$, unit: packet), if so, it keeps the $cwnd$ constant. If the packet loss is due to the congestion in the intermediate nodes, then the $cwnd$ value is multiplied by β , thus decreasing the value by 10%. The conventional CUBIC congestion control algorithm uses the β value of 0.7 [16]. The performance of the RACC algorithm was tested for different β values of 0.7, 0.8, 0.9, and 0.95 in a congested network. The performance of the algorithm was proven to be better for the β value of 0.9. If the RACC algorithm receives the notification of connection timeout, it enters the $reset_function()$. The $reset_function()$ resets the values of $wireless_loss$, $airtime_i$, $data_rate$, $bytes_acked$, $bytes_sacked$, and $packet_loss$ to zero.

The INT monitoring and airtime feedback in AP are implemented in the Click modular router framework³. The designed RACC algorithm was implemented in Python 3.8 programming language, using the CCP framework [15].

VI. PERFORMANCE OF THE DESIGNED RACC ALGORITHM

This section explains the test setup used to evaluate the performance of RACC algorithm. It then elaborates on the performance of the RACC algorithm in terms of throughput improvement and percentage airtime difference as compared to the CUBIC algorithm.

A. Test setup

The RACC algorithm aims to mitigate the airtime unfairness in wireless networks. To validate the performance of the

algorithm, problem statements have been formulated in terms of airtime consumption, throughput, and overall efficiency as stated in Section III-B. The experiments were conducted in Wilab2 testbed⁴. The experimental setup consisted of two APs connected by a wired connection and six end devices, with three end devices connected to each AP. Both the APs operate in different wireless channels and are far enough to avoid interference. To clearly see the impact of the designed algorithm, each end device is restricted to one flow (indicated in different colors in Fig. 2). To emulate different physical data rates in a network, the physical data rate of end devices 1, 2, and 3 are set to 6, 12, and 24 Mbps respectively, using the `iwconfig` command (only a few physical data rate values can be fixed using the `iwconfig` command and these are chosen for testing). The end devices 4, 5 and 6 connected to AP2 serve only as data receivers whose physical data rates are not fixed. The $timeslot$ fixed for $airtime_i$ allocation should be large enough to include the round trip times (RTT, unit: second) of all the end devices using the wireless link. Therefore, the ping tests were run in all the end devices and the $timeslot$ in AP was set to the value of average RTT, 0.066 s, as shown in table I.

TABLE I: Ping results of end devices

End device	<i>min</i> (s)	<i>avg</i> (s)	<i>max</i> (s)
End device 1	0.019491	0.067910	0.162247
End device 2	0.015527	0.066231	0.193776
End device 3	0.014046	0.063360	0.111578
Average (chosen timeslot)	-	0.066	-

A fixed payload of 5 mega bytes (MB) was sent from end devices 1, 2, and 3 at the same time to evaluate the airtime consumption and overall throughput achieved for CUBIC and RACC algorithms. The obtained results are discussed in the following section.

Since the performance of CUBIC congestion control algorithm is better than other existing conventional congestion control algorithms in terms of throughput and round trip time fairness, the performance of designed RACC algorithm is validated against CUBIC algorithm only [17].

B. Results and discussion

On sending the fixed payload of 5MB from the end devices using the designed RACC algorithm, the $cwnd$ of each end device was recorded and plotted in Fig. 3. From the plot, it can be seen that the $cwnd$ is dependent on the physical data rate of the end device. On starting the TCP connection at the same time, the end device 3 with a higher physical data rate has a higher $cwnd$ value. Further at the 15th second on the plot, it can be noticed that the end device 3 finishes its data transfer and the $cwnd$ of end device 1 and end device 2 are updated. A similar effect can be seen at the end of the transfer from end device 3, where $cwnd$ of end device 2 gets increased to

³<https://github.com/kohler/click>

⁴<https://doc.ilabt.imec.be/ilabt/wilab/>

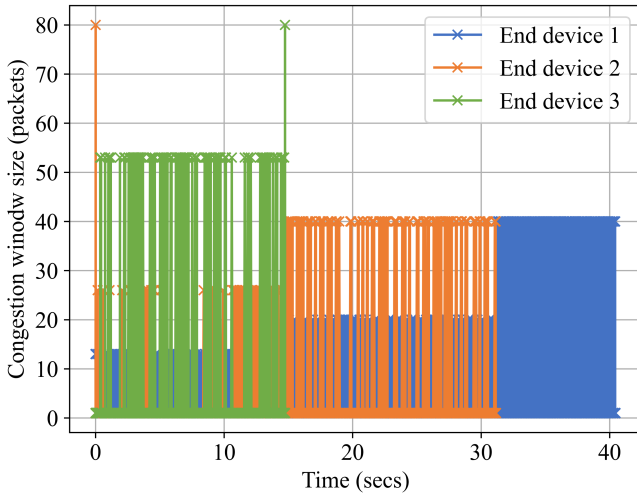


Fig. 3: The behavior of *cwnd* of the RACC algorithm

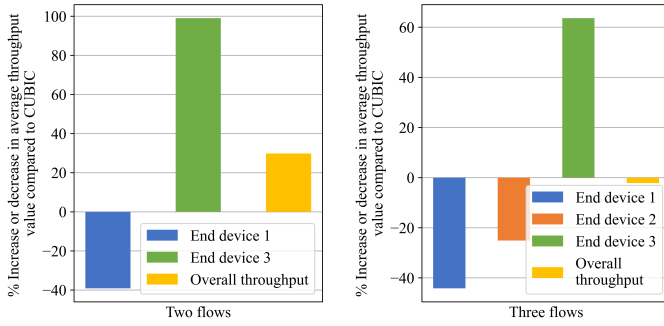


Fig. 4: Percentage increase or decrease in average throughput of the RACC algorithm as compared to the CUBIC

utilize the released capacity. Further, when the transfer from end device 2 is finished, the *cwnd* of end device 1 increases to utilize the whole capacity left.

For the same test of sending a fixed 5MB payload, the throughput of each end device was recorded. The test was conducted for two flows (end devices 1 and 3) and three flows (end devices 1, 2, and 3). The obtained results in Fig. 4 indicate the percentage increase or decrease in throughput of each end device as compared to the CUBIC congestion control algorithm. For the case of two flows, it can be noticed that there is a reduction in the throughput of end device 1 as compared to the throughput of the CUBIC algorithm. At the same time, there is around 99% improvement in the throughput of end device 3 as compared to the CUBIC algorithm. Thus the overall efficiency has been improved by 25% by using the RACC algorithm. For the case of three flows, the throughput of end devices 1 and 2 is lower than that of the CUBIC algorithm. End device 3 with a high physical data rate has increased its throughput by 45% as compared to the performance in the CUBIC algorithm. The overall efficiency of the RACC algorithm in this scenario is 2% lower than the CUBIC's performance. The overall throughput for three flows is lesser

than that of two flows, the reason being, that in the case of two flows, the number of devices contending for the channel is lesser. Hence, when one of the devices finishes its transfer, the other device can fully utilize the network. Whereas, in the case of three nodes, after one of the devices finishes its transfer, the other two nodes still contend for the channel access. This results in more white spaces as compared to the case of two nodes. This effect can be seen in both CUBIC and RACC algorithms, but the effect is higher in the case of RACC. Therefore, to answer the first question of the problem statement, the high physical data rate device has 99% and 45% better throughput on using the RACC algorithm than the CUBIC algorithm when there are two and three flows in the network respectively. On the other hand, the low physical data rate end devices can be penalized for their throughput when using RACC, as was the case for end devices 1 and 2. However, this is a trade-off for better airtime fairness usage between end devices.

Further, the airtime consumed by each end device is calculated and plotted the instantaneous airtime by summation of the packet air times over a 500ms timeslot. From Fig. 5, it can be noticed that airtime consumption is inversely proportional to the physical data rate of the end device. Since the CUBIC algorithm does not consider airtime fairness in the design, all the end devices finish the data transmission of a fixed 5MB payload at the same time. Whereas, since the RACC algorithm is designed with the intention to achieve airtime fairness, this was tested by continuously sending the data for a fixed time of 40 seconds and plotting the airtime calculated by AP in Fig. 6. From the plot, it can be noticed that the airtime consumption of all three end devices lies in the range of 5-30 ms. The calculated airtime on sending the fixed payload of 5 MB is plotted in Fig. 7 and the airtime of the three devices lie in the range of 10-40 ms, with the added advantage of improved performance in terms of throughput. When end device 3 finishes its data transfer, the airtime of end devices 1 and 2 is slightly increased and lies in the range of 20-40 ms. Finally, when end device 2 finishes its transfer, the airtime consumption of end device 1 is increased further.

Using these values, the percentage difference between the airtime consumption was calculated for the devices with low and high physical data rates, respectively. Since end device 1 has the lowest physical data rate, the calculated percentage difference between the airtimes of end devices 1 and 2 and the percentage difference between the airtimes of end devices 1 and 3 for CUBIC and RACC algorithms and plotted in Fig. 8. From the plot, it can be seen that in the case of CUBIC, the percentage difference between the airtimes is as high as 119%. This percentage difference is between end devices 1 and 3, i.e. between the lowest physical data rate and highest physical data rate devices of the network. Whereas, in the case of the RACC algorithm, the percentage difference is as low as 15% and 30% on average. Therefore, to answer the second question of the problem statement, the percentage difference in the airtime consumption of a high physical data rate device is 30% on average when the RACC algorithm is used.

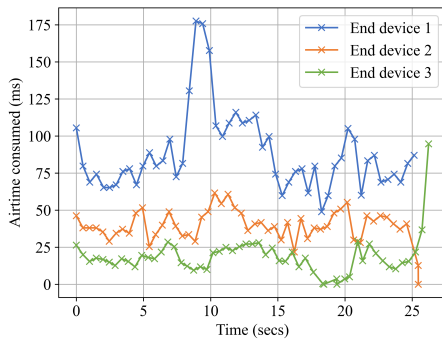


Fig. 5: Instantaneous airtime of the CUBIC algorithm on sending a fixed data payload of 5MB

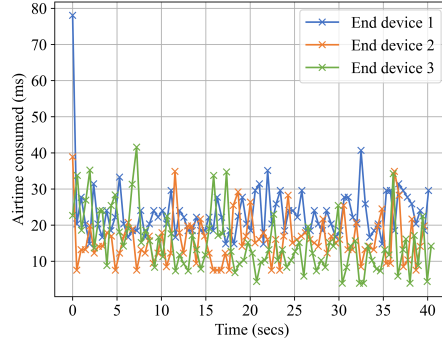


Fig. 6: Instantaneous airtime of the RACC algorithm on sending data for 40 secs

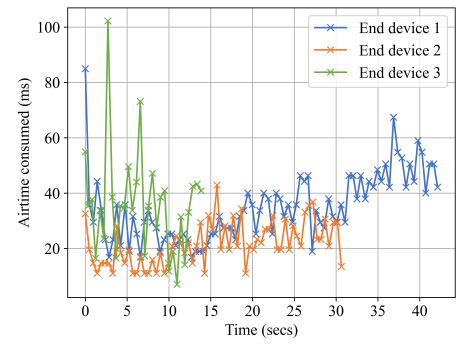


Fig. 7: Instantaneous airtime of the RACC algorithm on sending a fixed data payload of 5MB

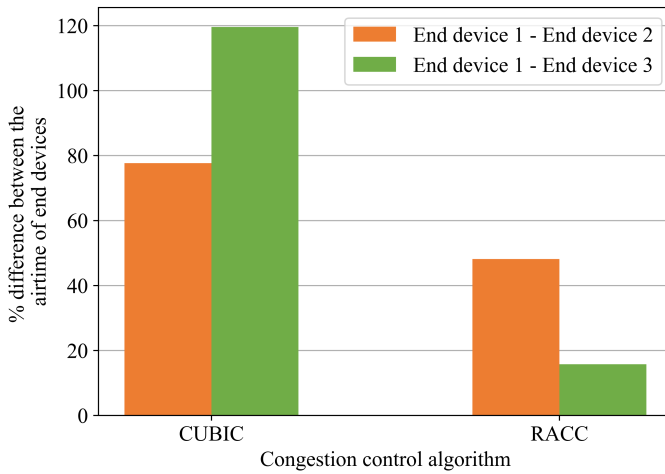


Fig. 8: Percentage difference in airtimes of low and high physical data rate devices

Though the percentage difference in airtime of low and high physical data rate devices is as low as 15%, there is still room for improvement. From Fig. 3 it can be noticed that there are still a few gaps between the congestion window size of end devices 2 and 3. Also, the end device 1 does not fully utilize the available bandwidth when there are no other flows in the network. Since the feedback from AP is explicit, the designed RACC reacts immediately to this and does not take into account the events in the past nor predict the future. Hence there is a possibility of underutilization or overutilization of the wireless bandwidth by certain devices.

VII. CONCLUSION AND FUTURE WORK

With a large number of industrial applications moving towards Wi-Fi, there is a requirement for flexible, real-time, reliable services from Wi-Fi protocols. Since existing application and transport layer protocols are designed for wired networks, it is important to address these drawbacks in wireless scenarios. One such important drawback is throughput degradation in mixed physical data rate networks caused by

airtime unfairness between low and high physical data rate devices. The designed RACC algorithm adapts the application data transfer rate based on the real-time network and airtime feedback, thus solving the issue of throughput degradation in mixed physical data rate industrial wireless networks. The implemented design was benchmarked against the CUBIC congestion control algorithm. The high physical data rate device had an average of 72% improvement in its throughput as compared to CUBIC when the RACC algorithm was used. The percentage difference between the airtime of the lowest and the highest physical data rate devices was reduced to as low as 15%, with an average of 30%. The overall efficiency of the RACC algorithm was 2% lesser than the CUBIC in the case of three flows. One of the main advantages of this design is that it was implemented on COTS devices as compared to the other works that were carried out in simulators. Since energy harvesting is crucial in industrial networks, as a next step, the designed algorithm will be investigated for harvesting energy in industrial wireless networks. The implemented algorithm will be modified and tested to balance the network's energy consumption by adapting the application data transfer rates.

ACKNOWLEDGMENT

This research was partially funded by the FWO-Flanders under grant agreements #S003921N and #G055619N and by the Flemish Government under the “Onderzoeksprogramma Artificiële Intelligentie Vlaanderen” program. This work was also partially supported by the CHIST-ERA grant SAMBAS (CHIST-ERA-20-SICT-003), with funding from FWO, ANR, NKFIH, and UKRI.

REFERENCES

- [1] Safdari, F. and Gorbenko, A., 2019, June. Experimental Evaluation of Performance Anomaly in Mixed Data Rate IEEE802.11ac Wireless Networks. In 2019 10th International Conference on Dependable Systems, Services and Technologies (DESSERT) (pp. 82-87). IEEE.
- [2] Haxhibeqiri, J., Seferagic, A., Bhat, R.V., Moerman, I. and Hoebeke, J., 2021, August. Tighter application-network interfacing to drive innovation in networked systems. In Proceedings of the ACM SIGCOMM 2021 Workshop on Network-Application Integration (pp. 53-57).

- [3] Haxhibeqiri, J., Isolani, P.H., Marquez-Barja, J.M., Moerman, I. and Hoebeke, J., 2020. In-band network monitoring technique to support SDN-based wireless networks. *IEEE Transactions on Network and Service Management*, 18(1), pp.627-641.
- [4] Dias, M.C., Caldeira, L.A. and Perkusich, A., 2015, August. Traditional TCP congestion control algorithms evaluation in wired-cum-wireless networks. In *2015 International Wireless Communications and Mobile Computing Conference (IWCMC)* (pp. 805-810). IEEE.
- [5] Tran, V.H. and Bonaventure, O., 2019. Making the Linux TCP stack more extensible with eBPF. In *Proc. of the Netdev 0x13, Technical Conference on Linux Networking*.
- [6] Bhat, R.V., Haxhibeqiri, J., Moerman, I. and Hoebeke, J., 2021, October. Adaptive transport layer protocols using in-band network telemetry and eBPF. In *2021 17th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)* (pp. 241-246). IEEE.
- [7] Baiocchi, A., 2023. Maximizing the stable throughput of heterogeneous nodes under airtime fairness in a CSMA environment. *Computer Communications*, 210, pp.229-242.
- [8] Kulenkamp, D.J. and Syrotiuk, V.R., 2021, May. Support for Differentiated Airtime in Wireless Networks. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (pp. 1-6). IEEE.
- [9] Høiland-Jørgensen, T., Hurtig, P. and Brunstrom, A., 2019, April. Polifi: Airtime policy enforcement for wifi. In *2019 IEEE Wireless Communications and Networking Conference (WCNC)* (pp. 1-6). IEEE.
- [10] Høiland-Jørgensen, T., Kazior, M., Täht, D., Hurtig, P. and Brunstrom, A., 2017. Ending the Anomaly: Achieving Low Latency and Airtime Fairness in WiFi. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)* (pp. 139-151).
- [11] Ben Ameer, C., Mory, E. and Cousin, B., 2018. Combining traffic-shaping methods with congestion control variants for HTTP adaptive streaming. *Multimedia Systems*, 24, pp.1-18.
- [12] Jayaudhaya, J., Supriya, S., Kandaswamy, V.A., Pandi, S., Kamatchi, S. and Priya, C.P., 2023, May. ACoCo: An Adaptive Congestion Control Approach for Enhancing CoAP Performance in IoT Network. In *2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC)* (pp. 189-194). IEEE.
- [13] Kim, J., Im, Y. and Lee, K., 2023. Enabling Delay-Guaranteed Congestion Control With One-Bit Feedback in Cellular Networks. *IEEE/ACM Transactions on Networking*.
- [14] Mishra, T.K., Sahoo, K.S., Bilal, M., Shah, S.C. and Mishra, M.K., 2023. Adaptive congestion control mechanism to enhance TCP performance in cooperative IoV. *IEEE Access*, 11, pp.9000-9013.
- [15] Narayan, A., Cangialosi, F., Raghavan, D., Goyal, P., Narayana, S., Mittal, R., Alizadeh, M. and Balakrishnan, H., 2018, August. Restructuring endpoint congestion control. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication* (pp. 30-43).
- [16] Rhee, I., Xu, L., Ha, S., Zimmermann, A., Eggert, L. and Scheffenegger, R., 2018. RFC 8312: CUBIC for Fast Long-Distance Networks.
- [17] Patel, S., Shukla, Y., Kumar, N., Sharma, T. and Singh, K., 2020, March. A comparative performance analysis of tcp congestion control algorithms: Newreno, westwood, veno, bic, and cubic. In *2020 6th International Conference on Signal Processing and Communication (ICSC)* (pp. 23-28). IEEE.