Cross-layer design approach for dynamic control over Wi-Fi network slices: bringing QoS Diversity to Wi-Fi based IoT networks

DAWID KULIKOWSKI, University of Twente, The Netherlands

Wi-Fi based IoT devices are deployed for a wide variety of different applications, with each application requiring different Quality of Service (QoS) guarantees. While the IEEE 802.11e amendment to the Wi-Fi standard specifies additions that enable QoS, only 4 traffic categories are defined with hard-coded properties in the standard. This improvement is insufficient for meeting the diverse QoS requirements in IoT networks. A novel solution of Wi-Fi slicing, which allows for the network to be divided into an arbitrary number of slices, with the priority of each slice being defined by a quantum value representing the relative fraction of airtime allocated to the slice. However, the current implementation of said architecture lacks robust control over network slice configuration parameters. The aim of this paper is to ameliorate control over slices by including channel access parameters from the WiFi MAC layer in addition to quantum values. Our proposed scheme enables improved network slicing capabilities in SDN-controlled Wifi networks in addition to providing superior QoS capabilities.

Additional Key Words and Phrases: 5G-EmPOWER, EDCA, IOT, QoS, Wi-FI

1 INTRODUCTION

Due to the steady advancement of technology, the use and development of Wi-Fi enabled devices continues to increase [11]. One of the driving factors believed to be behind the rapid growth in Wi-Fi traffic are Internet of Things (IoT) devices [24]. The connectivity requirements of such devices tend to differ from use case to use case, hence considerable effort has been put into enabling Wi-Fi to be optimized for a particular purpose [28]. Not all traffic is created equal - certain traffic may require prioritization, which may refer to lower latency, jitter (variance in latency) or higher bandwidth. In such cases, service differentiation may be beneficial to the overall quality of service experienced by the user.

With the ratification of the IEEE 802.11e amendment to the original Wi-Fi standard, compliant access points and stations (clients) are enabled with quality of service (QoS) capabilities. The amendment supports the prioritization of data based on traffic classes through an updated channel access mechanism which handles Medium Access Control (MAC) - the Enhanced Distributed Channel Access (EDCA) mechanism [8]. While the IEEE 802.11e amendment successfully prioritizes data depending on its type, it has certain limitations. Namely, the traffic classes, also known as access categories, are fixed, and only 4 are defined in the standard (see Table 1) [8]. This means that all possible traffic has to be classified into 4 different categories while there may feasibly be many more. Moreover, the configuration of access categories cannot be changed on the fly to accommodate changing network conditions. The values are standardized in the same amendment [8]. Due to the limited number of access categories and the fixed nature of their parameters, the current standard lacks the capability to efficiently meet the diverse QoS requirements for various IoT use cases [12].

Table 1. EDCA parameters as defined in IEEE 802.11e [8]

AC	CW_{min}	CW_{max}	**AIFSN**
AC_BK	aCW_{min}	aCW_{max}	7
AC_BE	aCW_{min}	aCW_{max}	3
AC_VI	(aCW_{min}+1)/2-1	aCW_{min}	2
AC_VO	(aCW_{min}+1)/4-1	(aCW_{min}+1)/2-1	2

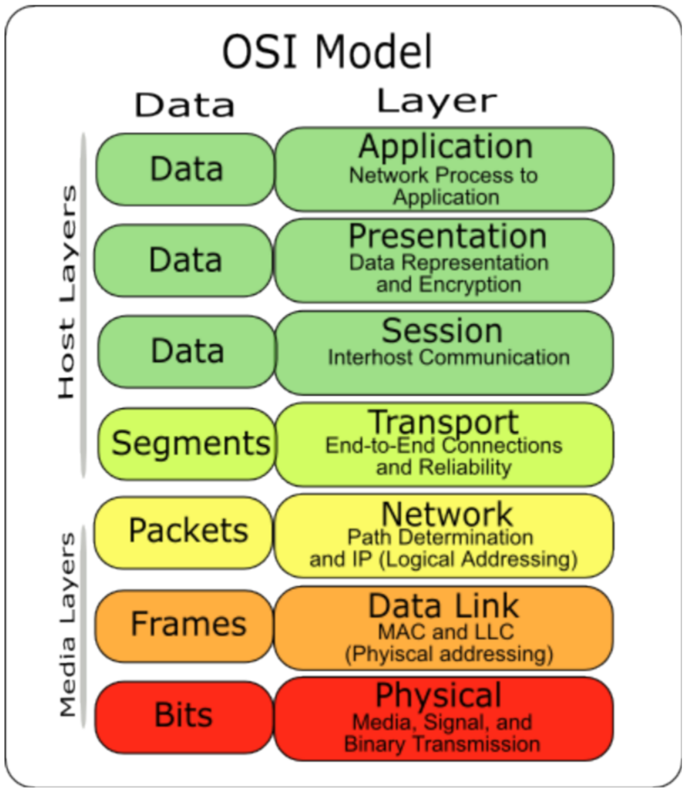
OSI Model
Data
Layer
Host Layers
Media Layers
Data
Application
Network Process to Application
Data
Presentation
Data Representation and Encryption
Data
Session
Interhost Communication
Segments
Transport
End-to-End Connections and Reliability
Packets
Network
Path Determination and IP (Logical Addressing)
Frames
Data Link
MAC and LLC (Phyiscal addressing)
Bits
Physical
Media, Signal, and Binary Transmission

Fig. 1. The OSI 7 layer model [15]

Work by Coronado et al. in [14] intends to improve upon the aforementioned amendment. The proposed alternative is to use *slices* defined and configured centrally by a software-defined networking (SDN) controller. An SDN controller allows for central control over various components of a network [27]. This alternative was implemented by the authors by expanding the *5G-EMPower* framework. The proposed architecture enables the network to be divided into multiple slices, each corresponding to a particular type of traffic.

TScIT 39, July 7, 2023, Enschede, The Netherlands

The number of slices is arbitrary and unbounded [14]. Slices are prioritized by the means of a quantum value, which is proportional to the amount of airtime that a given slice will be able to consume until it is scheduled to transmit again [14]. This prioritization takes place on the network layer of the OSI 7 layer model as presented in Fig. 1.

As part of our work, we will improve upon the aforementioned architecture by incorporating EDCA parameters as part of the slice configuration. Furthermore, we will expand EDCA as described in the standard by allowing for a 5th queue with independent parameters which can be controlled through *5G-EMPower*.

2 RELATED WORK

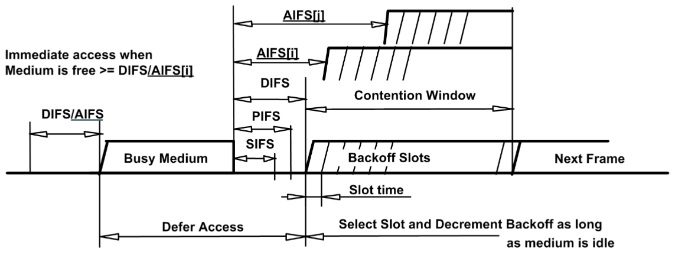
AIFS[j]
AIFS[i]
Immediate access when Medium is free >= DIFS/AIFS[i]
DIFS
Contention Window
DIFS/AIFS
PIFS
SIFS
Busy Medium
Backoff Slots
Next Frame
Slot time
Defer Access
Select Slot and Decrement Backoff as long as medium is idle

Fig. 2. Overview of the transmission mechanism of Wi-Fi [8]

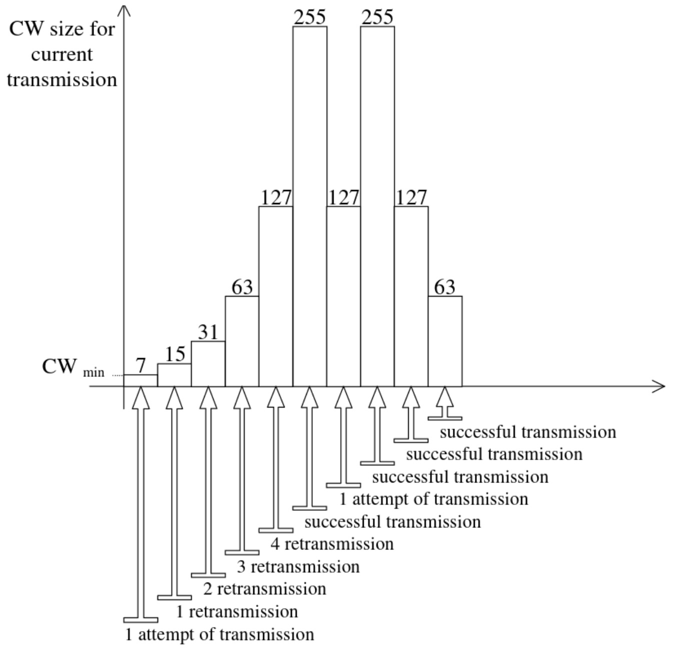

Fig. 3. A visualization of the contention window and the backoff procedure [22]

The focus of the original 802.11 W-Fi standard was fair division of resources (airtime) between a set of stations connected to a single

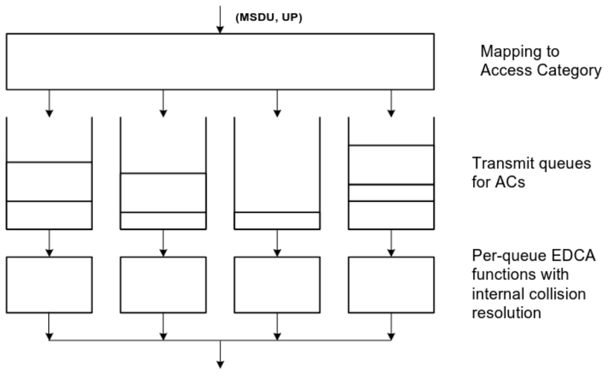
(MSDU, UP)
Mapping to Access Category
Transmit queues for ACs
Per-queue EDCA functions with internal collision resolution

Fig. 4. Per-AC queues introduced in EDCA [8]

access point. In order to satisfy the need for service differentiation, the original standard was amended with IEEE 802.11e to introduce access categories, a new contention algorithm (EDCA) and parameters that could be adjusted to differentiate different classes of traffic, namely:

(1) Minimum Contention Window (CW_{min})
(2) Maximum Contention Window (CW_{max})
(3) Arbitration Inter-Frame Spacing (AIFS)
(4) Transmit Opportunity Duration (TXOP)

These parameters work on the Medium Access Control (MAC) layer to govern the probability that an access point or a station attempts to transmit the packet [8, 12]. The physical medium through which data is sent between the access point and the clients (a range on the electromagnetic spectrum) is limited - simultaneous transmissions would make the data indiscernible to the receiver [26]. The purpose of MAC is to make decisions on when to transmit in order to avoid simultaneous transmissions (collisions) and maximize the chance of a successful transmission [26]. The most notable change introduced in EDCA is the introduction of several access categories, each with different values for parameters such as the minimum and maximum contention window (CW_{min}, CW_{max}) and arbitrary inter-frame spacing (AIFS) [8]. The AIFSN dictates the amount of time a station "should defer before either invoking a backoff or starting a transmission" [8]. The contention window parameters govern the backoff time. The way that the AIFSN value and the contention window work to change the characteristics of transmission within EDCA can be seen in Fig. 2 which demonstrate the decision-making process [8]. A station wishing to transmit a frame waits until the medium is no longer busy. It then waits for the amount of time represented by the AIFSN value, during which is senses whether the medium remains free. If the medium becomes busy during this time, the station will wait for a random amount of time (the back-off time/slots) bounded by the contention window size [8]. The contention window (CW) changes depending on the number of successful and unsuccessful transmissions as shown in Fig. 3 - for each successful transmission the contention window halves, while for each unsuccessful transmission it doubles with the maximum and minimum size bounded by CW_{max} and CW_{min} respectively [8]. The randomly selected backoff time is therefore bounded by 0 and the current size of the contention window, which in turn is bounded by the CW_{min} and CW_{max} values [8]. Together, these parameters specify which traffic type is prioritized, allowing for service differentiation.

In [25], researchers show the effectiveness of EDCA parameters in augmenting the properties of traffic classes relative to each other. Based on the empirical results, Villalón et al. concluded that CW_{min} and AIFS has the largest effect when set to be small for traffic with high priority such as packets carrying voice or video traffic while the CW_{max} does not have a large effect on the perceived performance. The researchers noted that the CW_{max} does not have a large impact because packets tend to be dropped if they are delayed too long. It should be noted however that the researchers expect the CW_{max} to have an impact on performance if it's set to be high enough for traffic classes with a lower priority as it should reduce the number of collisions [9, 25].

A large base of research exists exploring the concept of tuning EDCA parameters. Examples of such research are [18], [16] and [23]. Based on the results of the aforementioned papers, adaptive tuning of EDCA parameters improves the ability to meet QoS targets and the performance of the system. Notably, both papers test their approaches in simulations only. This is most likely partially due to the lack of suitable experimental suites supporting the dynamic control over EDCA parameters. A notable framework missing this capability is *5G-EmPOWER*. However, the results show that there may be benefits to dynamically adjust EDCA parameters based on the characteristics of traffic and conditions of the medium [16, 18, 23].

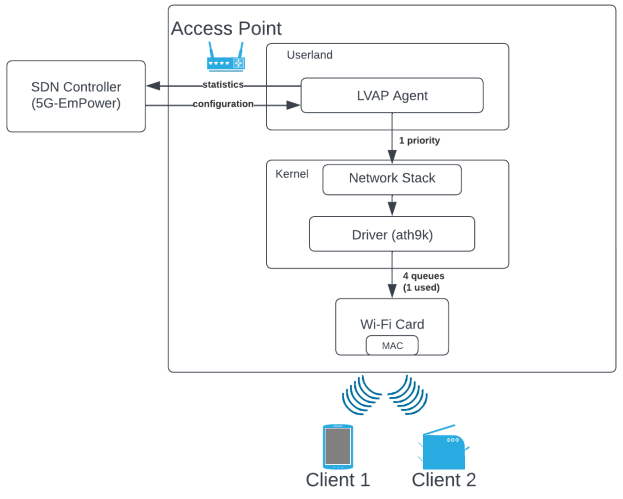

Fig. 5. The overarching architecture of *5G-EmPOWER*

Work by Coronado et al. provides a base for this project with their implementation of End-to-End Slicing. They designed a new architecture that allows for the dynamic provisioning of *slices* which correspond to a service class. Each slice has a corresponding set of *traffic rules* which can include, among other things, the fraction of airtime that can be assigned to the slice (the quantum value) and the EDCA parameters. The researchers implemented a basic version of this architecture on top of the *5G-EmPOWER* framework [14].

While the aforementioned architecture accounts for the diverse requirements of IoT networks, the service differentiation is done on the network layer of the OSI 7 layer model visible on Fig. 1. What becomes apparent is that this layer does not concern itself with the current condition of the medium - this is handled by the data link layer on which EDCA operates. Due to the nature of wireless networks, the medium conditions can vary wildly between locations and deployments, therefore being able to control prioritization on the MAC layer is favorable. However, the adopted standard which introduced EDCA is not flexible enough as it only provides 4 access categories with predefined parameters which include CW_{min}, CW_{max}, AIFSN and TXOP. In order to remediate this, we expanded the *5G-EmPOWER* framework and the architecture developed by Coronado et al. to allow for per-slice control over EDCA parameters. Furthermore, we expanded the number of available access categories to 5 from 4 in order to evaluate the feasibility and benefits of doing so, with the end goal of paving the way for expanding the number of access categories to 64. In order to evaluate the impact of our change, we look at changes in per-slice throughput and jitter when contention window parameters are changed together with the quantum compared to when only the quantum is changed.

3 CURRENT ARCHITECTURE

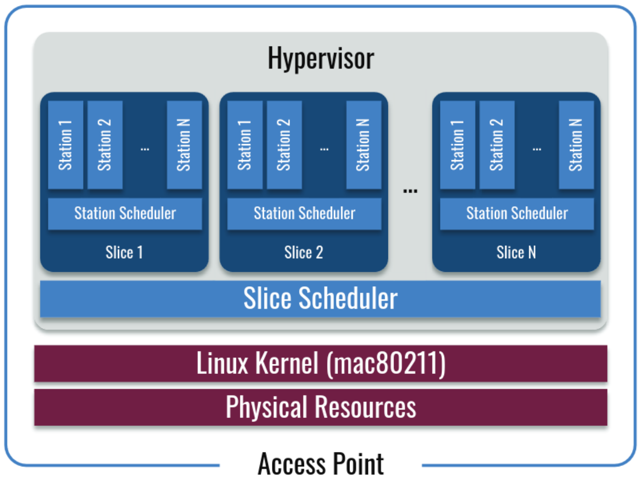

Fig. 6. The inner workings of the LVAP agent [14]

The *5G-EmPOWER* suite is made up of two main components - the Light Virtual Access Point (LVAP) agent and the controller. The LVAP agent runs on all access points while the controller fulfills the tasks of an SDN controller and can run on any device the access points can connect to. Each configured access point establishes communication with the controller, using the OpenEmpower protocol defined by authors of [13]. Using said protocol, the controller transmits various configuration parameters (most notably information regarding the configured slices) to the access point while the access point transmits relevant statistics regarding traffic on the network as visible in Fig. 5. The controller also makes decisions regarding, for example, which clients should be admitted to the network [13, 14].

The controller itself is made up of the back end which handles the actual management of associated LVAPs and the front end which can be used to manage the back end, internally using a simple REST API. The user can create or edit slices using the front end or the API, which the back end will save in a persistent database and transmit immediately to all connected LVAP agents.

The LVAP agent is more complicated than the controller as it has the task of managing the network traffic. It is based on the Click Modular Router (*Click* for short), which is a framework for "building flexible and configurable routers" [20]. The framework defines *elements* which can be joined together to process and forward packets between each other and devices, with an illustration presented in Fig. 7. The LVAP agent itself is an interconnected set of such elements [1]. The main *element* holds a queue for every slice, implementing the so-called hypervisor as illustrated in Fig. 6 [14]. When a slice is scheduled, packets are removed from the respective queue and pushed to elements to be forwarded to the kernel for sending.

When a packet is scheduled, it is forwarded to a hardware device using a built-in *Click* element (ToDevice) [20]. *Click* itself can operate in kernel mode or in user mode depending on user needs. For use with *5G-EmPOWER*, it is compiled to operate in user mode [2]. The mentioned elements that handles forwarding packets to a device/interface has many methods for doing so, including, but not limited to, Linux Sockets [20].

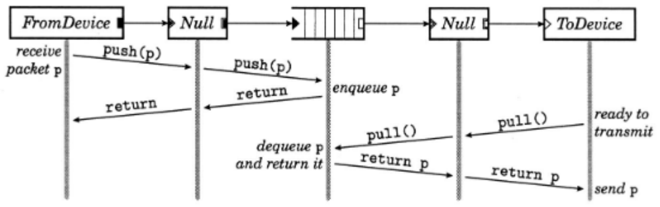

Fig. 7. The architecture of the Click Modular Router [20]

As a simplification, only Linux Sockets will be described and used. If *Click* is configured to use this method, packets will be transferred to the kernel using the **send** class of system calls [19]. The only information is can be passed to the kernel is the socket descriptor, a buffer, the length of the buffer and various pre-defined flags. In the kernel, the packet is converted to a structure called the Socket Kernel Buffer (SKB) which contains fields describing a packet, most notably a buffer containing data to be sent, the priority field and the queue mapping field [7].

As the device is operated in promiscuous (monitor) mode, packets received by the kernel bypass most of the Linux networking stack, such as the IP layer, with packets being forwarded to the driver with minimal processing [7]. 5G-EmPower supports only one driver - *ath9k* due to the fact that the *5G-EmPOWER* project made modifications to it [3].

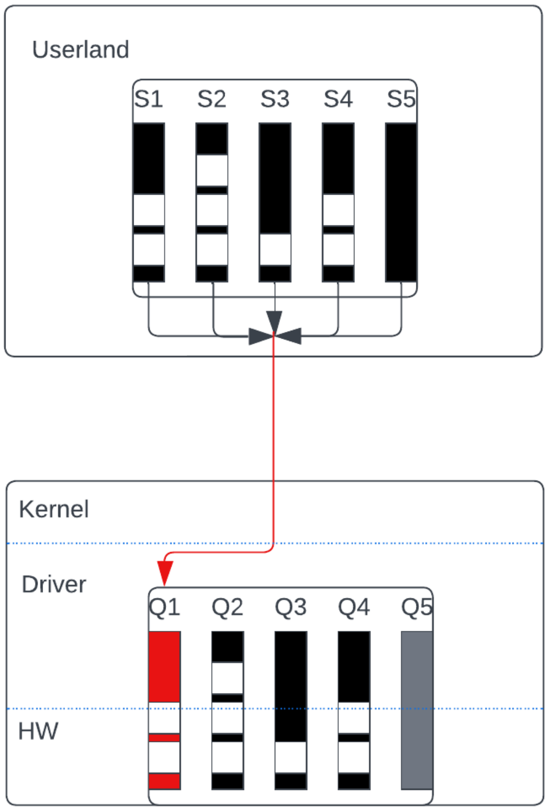

Fig. 8. An illustration of all slices being placed on the same queue

In the case of *ath9k*, decisions regarding medium access are made on the wireless card itself, with 4 total hardware queues used to forward packets to the device (1 per 802.11 access category) [7]. Each queue can be configured with custom EDCA parameters by writing to hardware registers of supported devices. When an SKB reaches the driver, the queue mapping property is used to place the packet on the appropriate queue for the device to read and transmit according to the queue configuration [7]. However, by default all packets get placed on the same queue (see Fig. 8) [7].

4 IMPLEMENTATION

In order to successfully test the impact of modifying various EDCA parameters on service differentiation, multiple components of the existing *5G-EmPOWER* framework and the Linux Kernel had to be modified.

To be able to easily experiment with various values and enable dynamic changes of the EDCA parameters, the controller had to be modified in order to add the necessary input fields to the front end and communicate said values to the LVAP. This change necessitated the modification of the HTML page to include said fields and

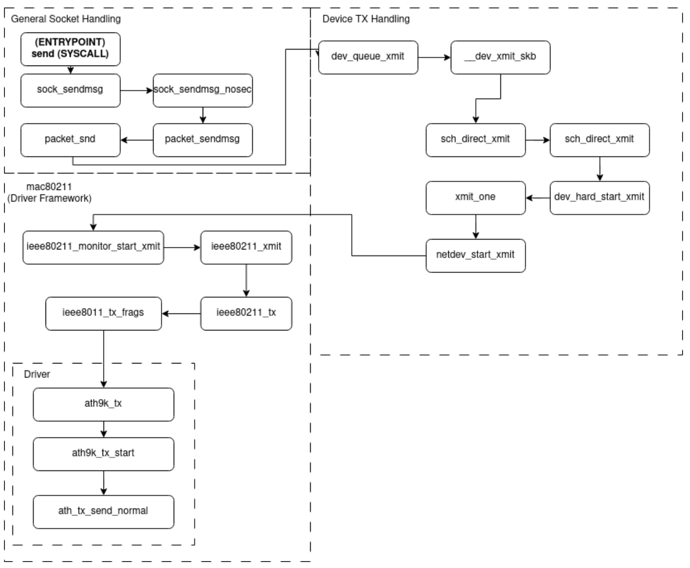

Fig. 9. The flow of a packet through the kernel [7]

changes in the JavaScript to include values inputted in the fields to be transmitted in the API calls to the controller backend. The back-end itself was modified to include this information in the slice data saved and transmitted to connected agents. As the structure of the message sent to the agent was changed, it was modified to extract said data and store it together with other slice-specific information.

While the slice configuration is known by the LVAP agent, it must be communicated to the driver as it is in charge of communicating the configuration to the device, which uses it for Medium Access Control. This is a challenge, because the configuration of EDCA parameters takes care on a lower layer as shown in Fig. 7. The *ath9k* driver is compatible with the cfg80211 component of the Linux Kernel, which is the "configuration API for 802.11 devices in Linux" [4]. Via this API, information regarding the configuration of the hardware queues can be passed to the driver which will take appropriate action. In order to take advantage of this API, the *libnl-tiny* library was used [29]. The LVAP agent was modified to use the aforementioned library to send the slice configuration to the driver after receiving a slice modification or creation command.

With the hardware queues configured, information regarding what queue each packet should be placed on had to be communicated to the Linux Kernel. The Linux Kernel will by default designate all packets to be placed on a single queue [7]. This issue is demonstrated in Fig. 8 - without the ability to communicate the hardware queue on which to place a packet, all packets will be sent with the same EDCA parameters. As mentioned above, the priority property of the SKB will be used by the kernel and the driver to determine which queue it will be placed on [7]. The kernel provides no suitable way to communicate this information via the **send** class of system calls, therefore the kernel had to be modified to accommodate this [6]. In order to ensure the stability of the kernel and preserve the ability of other applications to continue operating without modifications, simplicity and the reuse of existing facilities was preferred. The **send** system call allows for the communication of flags via a single unsigned 32 bit variable which is a part of the system call [6]. Based on existing documentation, only a part of said integer is used for existing flags [6, 7]. This leaves a sufficient number of bits to include the slice number as a part of the flag parameter, which can be read by the kernel. The code handling the system call, which is the entry point of Fig. 9 showing the flow of the packets, was modified to set the priority field of the SKB to the value of the slice passed via the system call.

Before forwarding the packet to the driver, the kernel uses the priority field of the SKB to derive the queue mapping property, which in turn is used by the driver to choose the hardware queue for the packet. By default, the kernel uses an appropriate conversion table to convert it to an access category value, with each access category corresponding to a hardware queue [7]. This logic was changed to simply write the priority field to the queue mapping field.

With these changes, *5G-EmPOWER* is able to take advantage of all 4 hardware queues, each corresponding to a single access category and in turn a single slice. While by default only 4 hardware transmission queues are used by the driver for data transmission, 10 are documented to be available in total [7, 21]. As a proof-of-concept, the driver and the kernel were expanded to support 5 hardware queues corresponding to 5 different slices. As shown in Fig. 9, an outgoing frame passes through multiple components of the kernel before entering the driver. Therefore, various supporting structures tracking the state of hardware queues were expanded to accommodate the additional queue; out-of-bounds checks were modified or removed. Lastly, as the number of buffers used by the wireless card to read outgoing packets is limited, the driver was modified to drop incoming packets once a queue reached a set cap [7]. This was done to prevent queues sending packets at a lower rate from starving higher-performing queues.

5 PERFORMANCE EVALUATION

In order to evaluate the benefits of controlling EDCA parameters in combination with the quantum value, we set out to measure the throughput and jitter experienced by slices. The aforementioned properties are key to the experienced quality of service and therefore valuable in comparing the level of service differentiation [17]. Two scenarios were tested: service differentiation using only the quantum value and service differentiation using both the quantum value and EDCA parameters. The values used for all relevant parameters such as the quantum value and the EDCA parameters can be found in Table 2 and 3. We decided to evaluate the impact of CW parameters due to research supporting its use in service differentiation [25]. The quantum was computed by distributing 10000 units among the slices using the following formula:

$$Q_{slice} = \frac{C_{slice}}{C} \cdot 10000$$

Q_{slice} represents the quantum assigned to the slice, while C_{slice} represents the bandwidth to be assigned to the slice. The total capacity of the access point is represented by C, which in our case was 24 Mbits/sec. This capacity may vary depending on the access point and medium conditions.

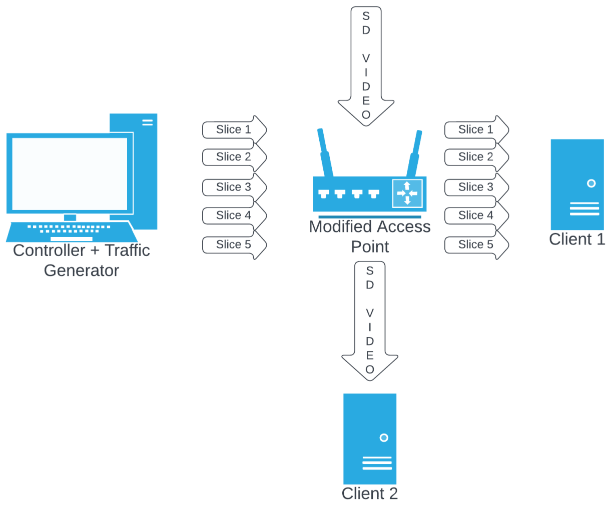

Fig. 10. The experimental setup

Table 2. Parameters for baseline measurements

Slice	CW_{min}	CW_{max}	AIFSN	Quantum
0	15	1023	2	834
1	15	1023	2	1250
2	15	1023	2	2084
3	15	1023	2	2500
4	15	1023	2	3334

Table 3. Parameters for improvement measurments

Slice	CW_{min}	CW_{max}	AIFSN	Quantum
0	31	1023	2	834
1	31	511	2	1250
2	15	127	2	2084
3	7	15	2	2500
4	3	7	2	3334

A test bed was set up including the controller, an access point and two clients as shown in Fig. 10. The controller played a minute part of the experiments and was only responsible for communicating the slice configuration to the access point as well as generating traffic on each slice to *Client 1*. *Client 2* was connected to the same access point, however it was only streaming SD video in order to cause contention on the medium. This was done because EDCA parameters being tested are expected have the biggest effect on the characteristics of transmission when contention on the medium is present. For example, with contention on the medium the contention window will grow beyond the minimum as shown in Fig. 3, worsening the performance for slices with large CW_{max} values and improving service differentiation.

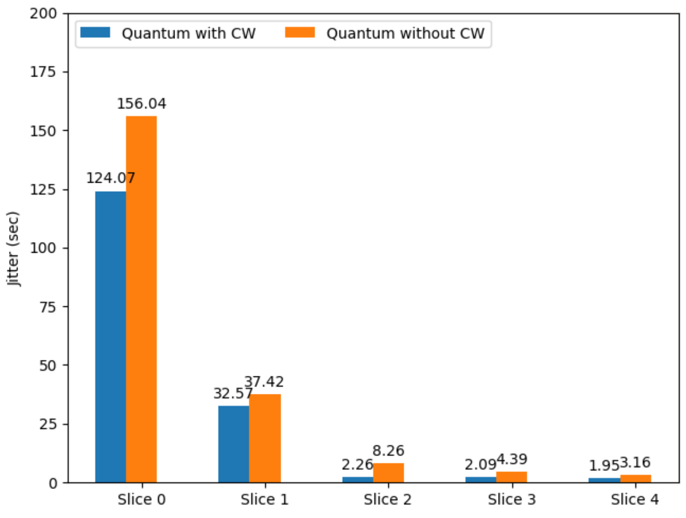

Fig. 11. Jitter of slices

The principal way of testing the level of service differentiation was using the *iperf3* command line utility to generate UDP traffic on each slice. UDP traffic was chosen as modifying the EDCA parameters on the access point primarily impacts the transmission not the receiving of frames. In order to generate traffic for each slice, the **–dscp** flag was used which sets the DSCP field of the IP header to the given value [5]. The DSCP value is a part of the IP header and is used by the *5G-EmPOWER* to discern the slice to which a packet belongs [14]. 10 Megabits of traffic were generated per slice used in each experiment (**-b 10M**) for 60 seconds (**-t 60**) [5]. This value was chosen as the sum of the traffic exceeds the capacity of the access point while the time was increased from the default of 10 seconds in order to account for the changing conditions on the network and the medium. Exceeding the capacity of the access point was important as service differentiation happens primarily in cases where the capacity of the network cannot meet the demand, necessitating prioritization.

In order to further account for changing conditions on the medium outside our control, we repeated each experiment 5 times and present the average in the Results section.

6 RESULTS & DISCUSSION

The analysis of the results can be divided into an analysis of jitter and throughput respectively.

6.1 Jitter

The jitter for both cases described in the Performance Evaluation section was plotted on Fig. 11 for ease of comparison. The notable result is that when the contention window parameters were changed in combination with the quantum value, the jitter was reduced for each slice. This result is likely due to the fact that each slice has now been given an independent queue. As shown in Fig. 8, all slices used to be placed on the same queue and transmitted with the same parameters. This poses an issue, because an influx of packets from a higher-quantum (higher priority) slice would mean more

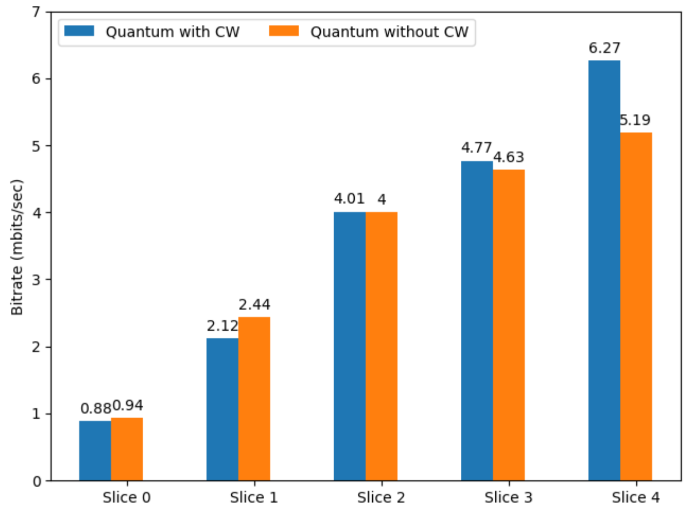

Fig. 12. Throughput of slices

latency experienced by the lower-priority slice as more higher-priority packets would get placed on the hardware queue before the lower-priority frames. However, in cases where the higher priority slice was not experiencing a lot of traffic, the low-priority slice would experience relatively low delay. The combination of these two factors mean that overall, jitter will always be higher when all slices end up in a single queue.

6.2 Throughput

Similar to the jitter results, the throughput results were plotted on Fig. 12. The results show that the throughput decreased slightly for Slice 0 & 1, remained similar for Slice 2 & 3 while increasing significantly for Slice 4. Slice 4, being the higher priority slice, has a small contention window, meaning that it would attempt to transmit most often by backing off for less time. It follows from this fact that most packets can feasibly be sent per second compared to slices with a larger upper bound on the contention window, especially in cases of a busy medium.

On the other hand, lower priority slices saw their throughput decrease slightly. This can largely be expected as the large contention window bounds mean that fewer packets can be transmitted per second. This is exacerbated by the limit placed on each hardware queue - since packets can't be sent quickly enough, the queues fill up, causing subsequent packets to be dropped and further reducing the throughput by inducing packet drops. The aforementioned issue is improved by the combination of the quantum value with EDCA parameters - packets for lower-priority slices are pushed into the kernel at a slower rate than for higher-priority slices, causing the hardware queues to fill up less rapidly. This fact supports the fact that the combination of both service differentiation mechanisms is superior to using contention window parameters alone.

7 FUTURE WORK

The limit placed on queues plays a big part in the total performance of the system and service differentiation. While the current solution is to drop incoming packets when the queue is at capacity, ideally better queue management would be employed. *5G-EmPOWER* shouldn't push packets into the kernel when the hardware queues are full or the kernel should keep a software buffers and drop the oldest packets. An alternative approach is tuning the quantum values in such a way that packets get pushed into the kernel at a pace that can be handled with the given EDCA parameters. Moreover, *5G-EmPOWER* itself can be expanded with the ability to control how many buffers a given queue can use, allowing slower queues to buffer more packets compared to fast queues which can send packets out immediately.

The driver can also be extended to support a larger number of slices. The inherent limitation of the driver and the hardware is that only up to 10 discrete hardware transmit queues are supported, meaning that some sort of software queue system will have to be used to dynamically schedule incoming packets and reconfigure queues. Such a change would necessitate far-reaching modifications of the driver and likely carry a large overhead. Nonetheless, the feasibility of such an approach should be considered.

As discussed in earlier sections of the paper, various papers aiming to dynamically tune EDCA parameters exist, however they have only been tested in simulations [16, 18, 23]. Using the techniques developed by this research paper, they may be implemented and tested in real conditions.

Furthermore, machine learning solutions for tuning the quantum values of slices exist [10]. Said models can be expanded to include EDCA parameters in order to improve their performance in service differentiation and meeting QoS targets.

Moreover, we have only evaluated the impact of CW parameters. Control over other EDCA parameters should be evaluated as they may be beneficial to service differentiation.

8 CONCLUSION

We have argued that the EDCA mechanism introduced in IEEE802.11 is not flexible enough to meet the diverse requirements of IoT networks while *5G-EmPOWER* only allows for service differentiation on the network layer. As a solution, we expanded both *5G-EmPOWER*, the Linux Kernel and the EDCA mechanism in order to allow for the dynamic adjustment of various data-link-layer parameters and introduced a 5th slice. Based on the results presented above, we have shown that the combined use of EDCA parameters and the quantum value is superior to just the quantum value in service differentiation. There are many possibilities to expand upon and use our results for future work.

REFERENCES

[1] [n.d.]. 5g-empower/empower-lvap-agent: The 5G-EmPOWER LVAP Agent. https://github.com/5g-empower/empower-lvap-agent
[2] [n.d.]. 5g-empower/empower-openwrt-packages: The 5G-EmPOWER packages for OpenWRT 19.07. https://github.com/5g-empower/empower-openwrt-packages
[3] [n.d.]. 5g-empower/empower-openwrt: This repository is a mirror of https://git.openwrt.org/openwrt/openwrt.git It is for reference only and is not active for check-ins or for reporting issues. We will continue to accept Pull Requests here. They will be merged via staging trees then into openwrt.git. All issues should be reported at: https://bugs.openwrt.org. https://github.com/5g-empower/empower-openwrt
[4] [n.d.]. cfg80211 subsystem — The Linux Kernel documentation. https://www.kernel.org/doc/html/v4.12/driver-api/80211/cfg80211.html

[5] [n.d.]. iPerf - iPerf3 and iPerf2 user documentation. https://iperf.fr/iperf-doc.php

[6] [n.d.]. send(2) - Linux manual page. https://man7.org/linux/man-pages/man2/send.2.html

[7] [n.d.]. torvalds/linux: Linux kernel source tree. https://github.com/torvalds/linux

[8] 2005. IEEE Standard for Information technology–Local and metropolitan area networks–Specific requirements–Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements. *IEEE Std 802.11e-2005 (Amendment to IEEE Std 802.11, 1999 Edition (Reaff 2003)* (Nov. 2005), 1–212. https://doi.org/10.1109/IEEESTD.2005.97890 Conference Name: IEEE Std 802.11e-2005 (Amendment to IEEE Std 802.11, 1999 Edition (Reaff 2003).

[9] Rathnakar Achary, V. Vaityanathan, Pethur Raj Chellaih, and Nagarajan Srinivasan. 2012. A New QoS Architecture for Performance Enhancement of IEEE 802.11e EDCA by Contention Window Adaption. In *2012 Fourth International Conference on Computational Intelligence and Communication Networks*. 74–78. https://doi.org/10.1109/CICN.2012.24

[10] Sri Harsh Amur, Kamran Zia, Alessandro Chiumento, and Paul Havinga. 2023. Autonomous Network Slicing and Resource Management for Diverse QoS in IoT Networks. In *2023 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*. 160–165. https://doi.org/10.1109/PerComWorkshops56833.2023.10150306 ISSN: 2766-8576.

[11] Sahba Bahizad. 2020. Risks of Increase in the IoT Devices. In *2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*. 178–181. https://doi.org/10.1109/CSCloud-EdgeCom49738.2020.00038

[12] Sunghyun Choi, J. del Prado, Sai Shankar N, and S. Mangold. 2003. IEEE 802.11 e contention-based channel access (EDCF) performance evaluation. In *IEEE International Conference on Communications, 2003. ICC '03.*, Vol. 2. 1151–1156 vol.2. https://doi.org/10.1109/ICC.2003.1204546

[13] Estefanía Coronado, Shah Nawaz Khan, and Roberto Riggio. 2019. 5G-EmPOWER: A Software-Defined Networking Platform for 5G Radio Access Networks. *IEEE Transactions on Network and Service Management* 16, 2 (June 2019), 715–728. https://doi.org/10.1109/TNSM.2019.2908675 Conference Name: IEEE Transactions on Network and Service Management.

[14] Estefanía Coronado, Roberto Riggio, José Villa1ón, and Antonio Garrido. 2018. Lasagna: Programming Abstractions for End-to-End Slicing in Software-Defined WLANs. In *2018 IEEE 19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*. 14–15. https://doi.org/10.1109/WoWMoM.2018.8449797

[15] Paul Edwards, Steven Jackson, Geoffrey Bowker, and Cory Knobel. 2007. *Understanding Infrastructure: Dynamics, Tensions, and Design.*

[16] L. Gannoune. 2006. A Non-linear Dynamic Tuning of the Minimum Contention Window (CWmin) for Enhanced Service Differentiation in IEEE 802.11 ad-hoc Networks. In *2006 IEEE 63rd Vehicular Technology Conference*, Vol. 3. 1266–1271. https://doi.org/10.1109/VETECS.2006.1683038 ISSN: 1550-2252.

[17] L. Gannoune, S. Robert, N. Tomar, and T. Agarwal. 2004. Dynamic tuning of the maximum contention window (CWmax) for enhanced service differentiation in IEEE 802.11 wireless ad-hoc networks. In *IEEE 60th Vehicular Technology Conference, 2004. VTC2004-Fall. 2004*, Vol. 4. 2956–2961 Vol. 4. https://doi.org/10.1109/VETECF.2004.1400602 ISSN: 1090-3038.

[18] Sudhanshu Gaur and Todor Cooklev. 2007. Using Finer AIFSN Granularity to Accurately Tune the Flow Ratios in IEEE 802.11e. In *2007 IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications*. 1–5. https://doi.org/10.1109/PIMRC.2007.4394345 ISSN: 2166-9589.

[19] Eddie Kohler. 2023. The Click Modular Router. https://github.com/kohler/click original-date: 2010-09-21T22:45:25Z.

[20] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. 2000. The click modular router. *ACM Transactions on Computer Systems* 18, 3 (Aug. 2000), 263–297. https://doi.org/10.1145/354871.354874

[21] Daniel Camps Mur. [n.d.]. Linux Wi-Fi open source drivers -mac80211, ath9k/ath5k-. ([n. d.]).

[22] Marek Natkaniec and Andrzej Pach. 2000. An analysis of modified backoff mechanism in IEEE 802.11 networks.

[23] C. N. Ojeda-Guerra and I. Alonso-Gonzalez. 2008. Adaptive tuning mechanism for EDCA in IEEE 802.11e wireless LANs. In *2008 14th European Wireless Conference*. 1–7. https://doi.org/10.1109/EW.2008.4623862

[24] Paulo Silva, Nuno T. Almeida, and Rui Campos. 2019. A Comprehensive Study on Enterprise Wi-Fi Access Points Power Consumption. *IEEE Access* 7 (2019), 96841–96867. https://doi.org/10.1109/ACCESS.2019.2928754 Conference Name: IEEE Access.

[25] José Villalón, Pedro Cuenca, and Luis Orozco-Barbosa. 2007. On the capabilities of IEEE 802.11e for multimedia communications over heterogeneous 802.11/802.11e WLANs. *Telecommunication Systems* 36, 1 (Nov. 2007), 27–38. https://doi.org/10.1007/s11235-007-9059-8

[26] Ping Wang. 2013. *Distributed Medium Access Control in Wireless Networks*. Springer, New York.

[27] Franciscus X. A. Wibowo and Mark A Gregory. 2016. Software Defined Networking properties in multi-domain networks. In *2016 26th International Telecommunication Networks and Applications Conference (ITNAC)*. 95–100. https://doi.org/10.1109/ATNAC.2016.7878790 ISSN: 2474-154X.

[28] Kamran Zia, Alessandro Chiumento, and Paul J. M. Havinga. 2022. AI-Enabled Reliable QoS in Multi-RAT Wireless IoT Networks: Prospects, Challenges, and Future Directions. *IEEE Open Journal of the Communications Society* 3 (2022), 1906–1929. https://doi.org/10.1109/OJCOMS.2022.3215731 Conference Name: IEEE Open Journal of the Communications Society.

[29] zorun. 2015. libnl and libnl-tiny – Technical Reference. https://openwrt.org/docs/techref/libnl Section: 2018-02-20T13:51:02-05:00.