2018 IEEE 26th International Conference on Network Protocols

If you can't beat them, augment them

Improving Local WiFi with Only Above-driver Changes

Ahmed Saeed*, Mostafa Ammar*, Ellen Zegura*, Khaled A. Harras†
*Georgia Institute of Technology
{amsmti3, ammar, ewz}@cc.gatech.edu
†Carnegie Mellon University
kharras@cs.cmu.edu

***Abstract*—The basic MAC mechanisms in IEEE 802.11 (WiFi) have remained largely unchanged for over 20 years. In this paper, we argue that the prevalence of WiFi makes it almost impossible to improve its performance through changes that require modifying hardware, firmware, or drivers. New applications, however, continue to exert novel performance demands. We suggest that changes should be developed as *augmentation-only* solutions through above-driver, kernel-level software modifications. An augmentation-only solution needs to maintain inter-operability and afford transparency in performance to existing WiFi devices, as well as enable minimum overhead upgradability. Our goal is to demonstrate the feasibility of MAC augmentation according to these principles. To this end, we leverage *soft scheduling*, where nodes are asked for a best-effort attempt to adhere to a given schedule. We allow the soft scheduler to *coexist with* and *work at a different time scale from* WiFi's Distributed Coordination Function (DCF); allowing it to reduce the time nodes spend contending for the medium while allowing DCF to handle only missed schedule slots and schedule divergence. We present a new Soft Token Passing Protocol (STPP) as an instance of this family of Soft Scheduling Protocols. We then show how STPP can be made part of a MAC protocol with specific performance improvement goals by developing the Wireless Low-Latency Local Links (WL4) system. We evaluate WL4 on a five node microbenchmark and quantify the system's overhead on network throughput and latency. We show that soft scheduling, via STPP, enables WL4 to adhere to our augmentation principles while improving the latency within the system.**

I. Introduction

The basic medium access coordination mechanisms in IEEE 802.11 (WiFi) have remained largely unchanged for over 20 years, with the exception of IEEE 802.11e [1] introduced in 2002 to incorporate QoS (compare IEEE 802.11 standards [2] circa 1997 and [3] circa 2016). During that time, a plethora of protocols were proposed to improve on WiFi's performance with little to no adoption. With billions of devices that are expected to be compatible and interoperable [4], [5], [6], [7], there is no clear path to the adoption of significant changes to WiFi or of a new standard.

New applications, however, continue to exert novel demands for low latency and elaborate bandwidth allocation. Augmented-reality (AR) SDK for Android [8] and iOS [9] are promoting multiplayer gaming over WiFi, which requires very low latency communication between players (e.g., [10]). Additionally, mirroring applications have been widely used and require seamless interaction between the mobile or hand held device and TV (e.g., [11]). Finally, the emerging trend of edge computing systems that leverage idle compute cycles on clusters of mobile and IoT devices within close proximity acts as another motivating domain [12], [13], [14], [15].

In this paper, we argue that the prevalence and ubiquity of WiFi makes it almost impossible to improve WiFi performance through changes that require modifying hardware, firmware, or drivers. We suggest that rather than thinking about such dramatic changes to the protocol, changes should be developed as *augmentation-only* solutions through above-driver, kernel-level software modifications that are independent of hardware. Such an approach makes improvements as simple as a kernel patch rather than a hardware upgrade, thus, increasing the chance of wide adoption.

We identify the following design principles for an augmentation-only solution:

1) it should be backward compatible and interoperable with standard WiFi devices,
2) it should improve on WiFi's performance for certain applications and should not degrade its performance for traditional use cases,
3) the existence of augmented devices should be transparent to non-augmented devices and should not harm their performance, and
4) existing WiFi devices should be upgradable with minimal overhead through above-driver, OS patches.

Our goal in this paper is to demonstrate the feasibility of MAC augmentation according to the four principles above. To this end, and with these principles in mind, let us consider the question of how one can augment WiFi's medium access control (MAC) protocol. WiFi relies on the Distributed Coordination Function (DCF) or Enhanced Distributed Channel Access (EDCA) for medium access coordination[1] [16]. DCF employs a CSMA/CA algorithm with exponential backoff. EDCA provides four priority levels through four DCFs each with different backoff parameters. DCF and EDCA are both based on *random access* techniques which are known to introduce significant and sometimes unpredictable latencies [17].

A meaningful augmentation to WiFi cannot simply tweak random access mechanisms or their parameters, and needs

[1]Point Coordination Function (PCF) is a third algorithm but it is rarely implemented or used.

978-1-5386-6043-0/18/$31.00 ©2018 IEEE
DOI 10.1109/ICNP.2018.00053

to enable additional mechanisms with significantly different features and capabilities. For MAC protocols, scheduled operation has long been considered as an alternative to random access [16], [18]. The main challenge is that typical scheduled transmission mechanisms require strict coordination and scheduling. As such, they are designed as replacements for WiFi. We, therefore, consider how one might augment WiFi with some form of less strict scheduled operation while preserving the augmentation principles highlighted above.

Throughout the long history of research and development on WiFi, several systems and protocols have attempted such augmentation (Section II). However, we find that they remain unrealistic and prohibitive due to one or more of the following requirements:

1) Making slight modifications to the driver assuming that such modifications can be easily adapted by *all* manufacturers [19].
2) Strict time synchronization that requires both processing and networking overhead. This overhead is unnecessary, especially because WiFi is increasingly becoming the main interface for weak devices with limited power [18], [20].
3) Operating in promiscuous mode to listen to or detect certain types of packets, which leads to power overhead to keep the network card on and listening. It also requires a large overhead to process all sniffed packets [19].

In this paper, we propose an augmentation scheme that adheres to the four principles introduced, while avoiding the above requirements. To enable such augmentation and interoperability, we adopt the notion of *soft scheduling* where upgraded nodes are informed of the network schedule and are asked for a best-effort attempt to adhere to the schedule. We allow the soft scheduler to *coexist with* and *work at a different time scale from* WiFi's DCF (i.e., milliseconds instead of microseconds). This allows the soft scheduler to introduce coordination and reduce the time nodes spend contending for the medium, while allowing DCF to handle only missed schedule slots and schedule divergence.

We introduce our new Soft Token Passing Protocol (STPP) that operates on top of 802.11's DCF to coordinate medium access (Section III). STPP is based on our key insight that *to provide performance that is better than random access, a protocol should balance schedule synchronization and idle time minimization.* STPP relies on token passing to synchronize a Time Division Multiple Access (TDMA) schedule. The protocol uses a combination of *softer* rules for token-passing and TDMA. For instance, while it attempts to keep the number of tokens at one, it allows for multiple tokens. Moreover, while it attempts to keep the schedule synchronized, it allows for overlapping transmission slots. This combination helps STPP avoid idle time while forcing at least a subset of nodes to adhere to the TDMA schedule, leading to improved average network performance. WiFi deployments are diverse, and thus, produce different requirements and constraints on the system.

We show how STPP can be made part of a MAC protocol with specific performance improvement goals by developing the Wireless Low-Latency Local Links (WL4) system (Section IV). STPP and WL4 are developed specifically for local communication settings where improving latency while maintaining high throughput is the main objective. This objective is motivated by the aforementioned use cases of screen mirroring and augmented reality applications. In those settings, a small number of devices are typically contending for the medium in the same room, and low latency and high throughput are key requirements. Therefore, WL4 relies on a central controller that allocates medium access shares based on predefined share requests made by local links between such devices.

We evaluate WL4 on a five node microbenchmark (Section V) and quantify the system's overhead on network throughput and latency. We evaluate WL4's impact on both synthetic traffic and traffic generated by an Augumented Reality (AR) application. We show that STPP enables WL4 to adhere to our augmentation principles, while improving the latency within the system. Our results show that our STPP enabled WL4 MAC improves latency in local links by 40% with a proportional impact on their throughput. We also show that the coordination overhead does not incur more than a 9% loss in terms of throughput while improving the average latency in the network by 38%. We also find that as the number of nodes increases, WL4's impact on latency increases (i.e., reduces latency) while its effect on throughput diminishes (i.e., smaller loss in throughput).

Overall, we make the following contributions:

1) We develop a protocol, STPP, that enables the improvement of 802.11 and adheres to the principles of augmentation we introduced without any violations.
2) We integrate the protocol into our WL4 system to create an operational MAC protocol in the linux kernel that improves the latency of local communication in WiFi.
3) We deploy WL4 in an experimental setup and show that it can greatly improve performance of high priority links without impacting legacy WiFi systems and applications.

II. Related Work

WiFi has a very long history that is hard to capture in a single paper. Surveys have been written to capture work done on different aspects of WiFi, from scheduling to power efficiency, and deployed in different settings, from home to enterprise to sensor and vehicular networks. We focus here on the directions we find to be most related to our approach, namely, attempts to schedule WiFi to improve performance with respect to both latency and throughput.

TDMA on Commodity Devices: There are several protocols that have been proposed over the years that rely on Time Division Multiple Access (TDMA) to improve the performance of WiFi. We pick two examples that can be deployed on commodity devices. Clock synchronization-based protocols like Soft-TDMAC [18] and OpenTDMF [20] attempt to allow nodes to reach microsecond level synchronization. This level of synchronization requires significant CPU and messaging overhead. Soft-TDMAC implements an NTP-like protocol using fine grain timers, which also consumes considerable

CPU. On the other hand, OpenTDMF focuses on scheduling WiFi as an access network to the Internet (i.e., synchronization between access points in an enterprise network). This does not provide a clear solution for local and direct communication over WiFi (e.g., two nodes in a multiplayer game). OpenTDMF achieves coordination between access points through out of band time synchronization. Furthermore, both OpenTDMF and Soft-TDMAC require changes to driver and kernel implementation. Our own STPP uses TDMA-like schedules without introducing significant CPU overhead while relying on above driver changes only.

Overlay MAC Protocols: Several protocols over the years have been proposed as "layer 2.5" or as overlay MAC protocols. Such protocols are the closest to STPP. However, earlier approaches violate one or more of the unrealistic requirements we discussed earlier. For instance, OML [19] is an overlay MAC protocol that provides extra medium access control functions above those provided already by the hardware and driver. OML requires editing the driver to control the number of packets enqueued in a layer below OML. OML also requires operation in promiscuous mode, which as discussed earlier leads to power and processing overhead.

"Look-Who-is-Talking" Protocols: Distributed-Centralized hybrid MAC protocols have been proposed aiming to schedule either uplink only [21], downlink only [22], or relying on relative scheduling for both uplink and downlink flows [23], [24]. Domino [23] and LWT [24] rely on relative scheduling between nodes. In particular, each node listens for who is currently transmitting to know its own turn to transmit. This process is coordinated by a central node that transmits the schedule to everyone. This approach makes these protocols applicable in local communication scenarios which resemble WL4's approach. However, Domino and LWT make several non-practical assumptions.For instance, both Domino and LWT require nodes to overhear the medium to gain any benefits which causes high processing and power overhead. Furthermore, both LWT and Domino require significant changes to how WiFi hardware and software operate, which renders it not practical and thus difficult to deploy. Significant changes to WiFi operations and hardware are required to allow devices to detect the node currently transmitting and synchronize schedules efficiently. STPP simplifies this approach by relying on token passing to determine the node that should be transmitting.

QoS Support in MAC protocols: These protocols provide differentiation of service between Access Categories (ACs). For instance, 802.11e [1] provides coarse grain level statistical differentiation of service by differentiating between the DCF parameters of different ACs. This approach gives high priority classes improved likelihood of medium access. SlickFi [17] provides differentiation of service by adapting channel width based on application demand. This is achieved by relying on two radios in the access point. Another example is SoftMAC [25] which provides QoS aware queuing and admission control above the driver. These approaches can be considered orthogonal to STPP as they provide differentiation of service within the random access platform. STPP can leverage this as well.

Low Latency WiFi: Recently, WiFi has been singled out as the major source of latency in WiFi-based Internet access [26], [27]. Several approaches have been proposed to lower the latency including better queuing mechanisms in access points [26], relying on the redundancy of wireless links [28], and changing access point parameters including transmission power, location, and bit rate adaptation mechanisms [29], [27]. This line of work focuses on improving WiFi induced latency for Internet access by improving access point characteristics. However, for local communication, going through the access point remains a major source of latency and adds one hop to the data path. Furthermore, most of the suggested improvements target latencies induced by current designs of access points. This makes them not useful in handling latency of direct links, where nodes are talking directly to each other not through the AP. In this paper, we focus on minimizing latency for local communication. STPP is orthogonal to presented work on low latency WiFi. For instance, direct local links are not impacted by improvements in the access point. On the other hand, they can benefit from techniques like latency aware adaptive rate. Moreover, networks discussed in earlier work can benefit from scheduling offered by STPP.

Soft Real-time Scheduling: This approach stems from work on soft real-time systems that require statistical guarantees for meeting deadlines as opposed to hard real-time systems that require meeting all deadlines [30]. However, in this paper, we push the idea by further relaxing the guarantee requirements on the soft scheduling algorithm. This is enabled by allowing the soft scheduler to operate at a coarse time granularity while allowing the WiFi DCF to resolve conflicts at small time granularities.

Multi-hop, Ad-hoc, and Sensor Networks: Improving the performance of local wireless communication is a well-studied problem in the context of multi-hop wireless networks [31], ad-hoc and mobile ad-hoc networks [32], and sensor networks [33]. These proposals develop MAC protocols that achieve some network objective (e.g., high throughput or low latency) while sometimes dealing with issues related to dynamic network topologies (e.g., a mobile ad-hoc network). The resulting solutions are by necessity complex, do not necessarily target low latency, and do not directly map to the simple WiFi scenarios we consider in this paper.

III. Soft Token Passing Protocol

We now describe the Soft Token Passing Protocol (STPP) as an instance of the soft scheduling approach. In this approach, nodes are informed of the network schedule and are asked for a best-effort attempt to adhere to the schedule. Later, in Section IV, we will show how STPP can be used as part of an overall MAC protocol architecture that meets the augmentation principles discussed in Section I.

A. Problem Context

STPP operates in a WLAN with a single access point and where random access using 802.11's distributed coordination

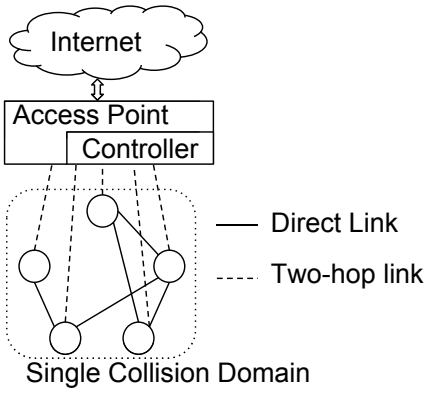

Fig. 1. Example of an STPP network.

functions (DCF) is available as a medium access control protocol. Direct links between mobile devices within range of each other are enabled using TDLS [34]. The devices within the WLAN are divided into separate collision domains, as shown in Figure 1. In a single collision domain any two simultaneous transmissions cause a collision.

STPP assumes the presence of a controller that coordinates its schedule. This controller can reside in an access point or in one of the participating nodes within a collision domain. Relying on the access point as the controller is the most straightforward approach. However, this requires upgrading the software of access points. If this is not possible, controllers can be deployed on one of the wireless devices. In this case, one would need to deploy procedures for leader selection and leader migration in case of controller failure (e.g., controller node shutting down or walking away). Solutions for such scenarios have been well-researched in the distributed systems literature [35]. For our purposes, we assume a fixed and reliable controller and focus on the details of STPP itself.

To identify collision domains, nodes exchange periodic beacons to construct a list of nodes they can directly overhear. These lists are sent to the controller which performs clustering to group nodes in collision domains. Note that these beacons allow for handling node mobility by keeping an updated list of nodes within a specific collision domain. A node in the WLAN will send and receive traffic of three types: 1) traffic from/to nodes outside the WLAN, 2) traffic from/to nodes in the same WLAN but at a different collision domain, and 3) traffic from/to nodes within the same collision domain. The first two types of traffic have to go through the access point.

The focus of this work is on local communication, be it through direct links between nodes within the same collision domain (e.g., using TDLS), or between nodes and the access point. Single-collision-domain local traffic is handled by STPP by augmenting the network stack on the sender. Importantly, STPP does not introduce any changes to the receiver. STPP controls medium access by constructing schedules that allocate time slots for each link. The length of a link's slot can be determined in multiple ways (e.g., relative shares).

In our design and implementation, we determine slot duration and shares as follows. Each node probes the length of the queue at its transport layer and sends it to the AP. We choose to probe transport queues as they are more representative of application demand. For instance, in the Linux networking stack, TCP Small Queue maintains a small number of packets in the queues of lower layers [36]. The AP uses the relative lengths of the queues to determine slot shares. We have a maximum cap for demand where all demand equal to or higher than the cap is set to the cap. Note that this maximum demand can be set based on a node's weight or priority. We note that shares can be translated into bits per second allocations based on observed rates. Slots in the schedule span a few milliseconds and their lengths depends on the allocated share of each node. Hence, STPP coordinates medium access at the millisecond level and delegates to DCF coordinating medium access at the microsecond level to handle missed slots and occasional schedule divergence (e.g., when collision occurs due to nodes missing their slot).

B. Protocol Overview

A typical approach to increased coordination is Time Division Multiple Access (TDMA) where each local link is allocated a time slot over which it can transmit without contention. One of TDMA's problems is that the time allocated to a link may go unused if there is no data to transmit on the link. Another problem is that TDMA schedules require strict synchronization within the network [37]. Correct behavior of TDMA protocols is contingent on avoiding collisions. Hence, synchronization mechanisms exist to keep the slots from overlapping during which no node can transmit. These problems can be addressed by using a token passing protocol [38] in which tokens representing exclusive permission to transmit are passed among nodes in some order. With token passing, every node has a predecessor from which it receives a token and a successor to which it transmits this token. In a WLAN setting, token loss can be frequent [39]. Correct behavior of typical token-passing protocols is contingent on having one token in the network. Hence, complex and high overhead mechanisms are implemented for token recovery [38]. Because of their complexity and strict synchronization requirement, neither TDMA nor token passing allow WiFi augmentation according to the principles we laid out in Section I.

To achieve our objectives we develop the Soft Token Passing Protocol (STPP) with a design philosophy that attempts to improve on contention-based medium access by *increasing coordination* and *avoiding idle time*. The protocol is built on top of WiFi's DCF functions thus providing backwards compatibility with WiFi-only devices. Relying on a centrally constructed, distributedly enforced schedule, STPP imposes *softer* rules compared to typical token-passing and TDMA protocols. It allows for multiple tokens, while attempting to reduce them to only one. It also allows for overlapping slots, while attempting to synchronize the schedule. In other words, links are never forced to stop transmitting by the protocol when it is trying to recover from an error. STPP operates on a

per-link basis where a link is a communication flow between two nodes. Note that due to timed slots, nodes can detect token loss through timeouts. This approach allows for controlled allocation to links and significantly reduces protocol overhead. STPP's softness can result in giving permission to multiple links in the WLAN, thus resulting in potentially overlapped transmissions. These are handled by the underlying 802.11 DCF functions.

C. Protocol Operation

The key to STPP operation is the scheduling algorithm that controls access to the channel for different links. A link is defined as a connection between two nodes within a single collision domain. We describe the basic operation of the algorithm for direct links, followed by the mechanisms used for robustness in the face of frame loss or node failure. We then discuss how unscheduled traffic is handled.

1) Basic Operation: We now discuss the most common features of the STPP protocol in standard operational mode.

Schedule Construction: The algorithm takes as input the per-link requests for shares of the channel, indicated in multiples of a minimum allocation time. Similar to a TDMA system, the controller uses these requests to create a schedule comprising an order for links to take turns accessing the channel. The schedule is shared with all STPP nodes so that each link sender knows when to take its turn. In each cycle through the full schedule, each link gets to access the channel for the requested allocation time.

Unlike TDMA, STPP is augmented with token passing. The source node for the currently active link holds a token. When the source node either completes transmitting all its data or reaches the end of its allocation, whichever comes first, it transmits the token to the source of the next link in the schedule. This simple and old idea allows more efficient use of the channel than a strict TDMA schedule. It also relaxes the need for strictly synchronized clocks at each node since token passing is used to indicate a turn taking transition. The schedule is shared with all nodes via reliable TCP connections between the controller and each node in the collision domain. The token is broadcast so that all nodes, including the controller, overhear the token transmission.

Token Structure: The token has two parameters, the ID of the link sending the token and the ID of the link receiving the token, used to specify which node should pick up the token. In steady state, each link gets one turn per cycle to transmit up to its requested share. If a link does not need the full share, it passes the token to the next link.

Schedule Membership Management: When a new link is needed, the source node notifies the controller, which updates and transmits the new schedule. The same steps occur when a link is no longer needed: the source node notifies the controller; the controller recomputes the schedule and transmits the new version to all nodes. On the creation of a new schedule, all nodes update their shares and order. The schedule message is treated as a token for the first node on the schedule with the rest of the nodes pausing transmission.

Algorithm 1 Soft Token Passing Functions.

```
1:  procedure OnTokenReceived()
2:      if TokenExpiryPeriod > Now &&
        Token.Src != TxNode then
3:          Discard Token
4:      else if Token.Dst == ID&&QueueHasPackets() then
5:          Refresh Share
6:          Resume Transmission
7:      else
8:          TxNode = Token.Dst
9:          TokenExpiryPeriod = Now + Share[Token.Dst]
10:     end if
11: end procedure
12: procedure OnDataFinishedOrShareFinished()
13:     Generate and Send Token
14:     Pause Transmission
15:     Start Max Token Passing Timer
16: end procedure
17: procedure OnMaxTokenPassingTimer()
18:     Refresh Share
19:     Resume Transmission
20: end procedure
```

2) Protocol Robustness: The protocol has additional mechanisms to make it robust to the following issues.

Multiple Tokens: If a currently active link holds a token and a second token appears on the channel, the second token will be discarded by its receiver. To achieve this, each link estimates the time when the destination link of a token should finish and keeps track of that transmitting link. During that period, referred to as the Token Expiry Period, all tokens are discarded except for the token sent by the transmitting link, which also handles scenarios where the links finishes earlier than planned (i.e., that token is not discarded). This ensures that only one token survives, as the token sent by the transmitting link will start a similar period of token discarding. However, a lost token in a perfectly synchronized schedule can cause idle periods of the length of the schedule. Hence, we allow the Token Expiry Period to be a system parameter that controls how strict the system is about having multiple tokens.

Token Loss: Each link estimates the time when it should next get the token, based on knowledge of the schedule. After completing a turn, the source node sets a timer estimating the Max Token Passing Time. If the timer expires without the return of the token, the source node assumes it has the token and takes its normal turn. When timers are accurate, this mechanism allows the next source node in the schedule to quickly recover the lost token. If a timer fires on a late, but not lost token, the token will end up reaching the node while it is transmitting. In that case, the node assumes that its share just started on the reception of the token and ignores the data transmitted before the token. This helps keep the schedule synchronized at the expense of giving that node more than its allocated share.

Node Failure: If a node fails then any links it participates in also fail. If the node is the source of a link then failure will be evident when the token is passed to the link and there is neither data traffic on the medium nor a passing of the token to the next link. The mechanisms that recover from a lost

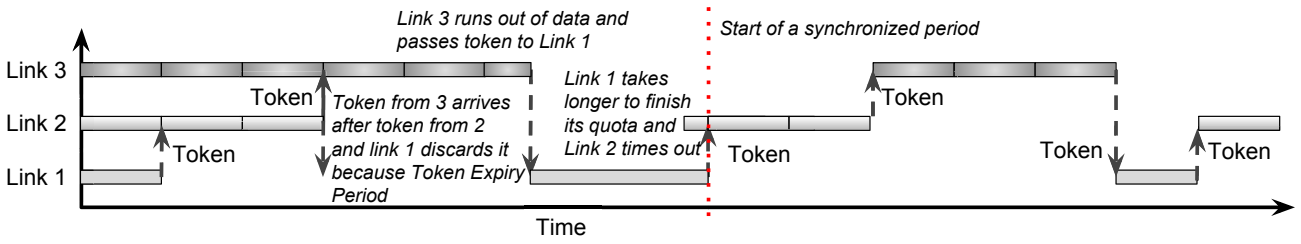

Fig. 2. Example of Soft Token Passing handling three special cases of operation. In this scenario, token passing sequence is Link1 to Link 2, Link 2 to Link 3, and Link 3 to Link 1.

token will recover the token; the controller will also observe the node silence, remove all links involving the node from the schedule, and transmit the new schedule. This is also useful to avoid having long schedules with idle nodes. Hence, STPP follows a policy of removing nodes if they are silent (i.e., not transmitting) for more than two seconds.

3) Handling Unscheduled Traffic: Nodes that do not implement STPP, yet share the channel, will have unscheduled traffic. To maintain backward compatibility and co-exist with non-local traffic, the controller monitors the channel usage by non-scheduled traffic and predicts the nominal rate available for scheduled traffic. This prediction of the nominal rate is calculated as a weighted moving average of the rate achieved during the two most recent schedules. The controller keeps track of nominal rates, and when change occurs this nominal rate, it sends it to all nodes. Note that changes in nominal rate are triggered at the start or the end of transmission of unscheduled traffic, which is typically captured by the access point. We rely on such triggers to transmit the nominal rate from the controller to the nodes. The task of calculating a nominal rate is not an overhead on the controller as it is tasked with overhearing the medium for scheduling purposes as well. Given a predicted nominal rate, the links translate their time share allocation into a byte limit, and use this byte limit to determine the end of a turn.

4) Two-hop Links: A two-hop local link is used when its performance is expected to be better than the direct link. A background monitoring process provides measurements to determine which to use (§III-F). In addition to the direct link share requests described above, STPP also takes requests for shares of two-hop links. Each two-hop link is converted into two "direct" links, with the AP as an endpoint in each. These two direct links are incorporated into the turn taking schedule and treated as direct links with respect to token passing.

The operations described above are encoded in three main procedures `OnTokenReceived`, `OnDataFinishedOrShareFinished`, and `OnMaxTokenPassingTimerExpiry` as shown in Algorithm 1.

D. Soft Token Passing Scenario

Figure 2 shows an example of a typical STPP schedule. In this example, we assume for simplicity that all links are transmitting at the same rate whether they are using the link exclusively or contending for it. Note that because of this example, we can use shares in terms of time or rate interchangeably. We also assume that each link belongs to a different pair of nodes (i.e., this scenario involves a total of 6 nodes).

In this example, we present a worst case scenario where all links timeout and start transmitting at the same time causing collisions which are handled by DCF. Links get to a contention period where they timeout due to multiple token losses at time zero. Links 1, 2, and 3 are allocated shares of 10Kb, 20Kb, and 30Kb respectively. Link 1 finishes its share first and passes the token to Link 2. On receiving the token, Link 2 refreshes its share and Link 3 starts a token expiry period of length 20Kb. Recall that the Token Expiry Period is a system parameter that controls how strict the system is about having multiple tokens. In particular, during a Token Expiry Period, all tokens are discarded except for the one generated by the transmitting link according to the schedule. Using the Token Expiry Period, STPP reduces the number of competing links. Then, links 2 and 3 finish at the same time. Links 3 and 1 receive tokens from both Links 2 and 3. The token from Link 2 is transmitted first, forcing a Token Expiry Period at Link 1 which discards the token generated by Link 3. This is resolved independently and Link 3 refreshes its share. Link 3 runs out of data and transmits its token before it finishes its planned share. Link 1 receives the token and starts transmitting. However, due to interference, Link 1 takes a longer time to finish its share. Link 2 times out and starts transmitting. Link one finishes and passes the token to Link 2. Link 2 refreshes its share. Afterwards, tokens are passed as expected.

E. Soft Token Passing vs DCF

The load on the network is a critical factor in determining the impact of STPP on network performance in comparison with DCF. There are two parameters controlling the load: 1) the number of nodes, and 2) the traffic load at each node. STPP is most useful for a small to medium scale network (2 to 10 nodes) with medium to high traffic loads. In this case, STPP reduces contention and results in better average network latency and negligible effect on throughput. As the number of nodes increase, the chances of two or more nodes contending increases (e.g., due to the underestimation of nominal rates). This will lead to degradation in the performance of STPP bringing it closer to DCF.

On the other hand, as the load on each node decreases, STPP coordination is turned into overhead. This means that DCF will be more suitable. This is the behavior that STPP attempts to achieve by removing nodes that are silent for more than two seconds from the schedules. This removal leads to a reduction in the number of nodes on the schedule as load decreases, leading to smaller overhead of the coordination algorithm. Also, note that under very light loads, traffic can fit in one share allowing a node to transmit immediately once it has traffic.

F. Direct vs two-Hop

We note that using a direct link is not guaranteed to provide low-latency local communication. We now focus on how to determine which path to choose: direct communication, or communication through the AP. Nodes have to compare the characteristics of the direct link to the AP-based link. Several metrics have been presented in literature to compare wireless routes in terms of latency, RTT, and throughput per hop [40].

Since throughput is mostly determined through shares in the STPP schedule, we focus on latency as the main metric for selecting between direct and two-hop links. STPP relies on periodic pings between an actively communicating pair of nodes. STPP sends two consecutive ping packets through different paths by modifying the From DS and To DS fields in the MAC header of the ping packet. This determines whether the packet goes through the access point or directly to the destination. The response packets mirror the markings of the ping packets.

Based on our measurements, we observe that the main cause of deterioration of latency in direct links, compared to two-hop links, is distance. This means that if latency on a direct link was found to be better than latency of the two-hop link, this can only change at a rate corresponding to human walking speed (which is the main way of increasing distance between two nodes). Hence, we send ping packets at 100ms intervals. If 10 consecutive pings determine one approach to be superior to the other, we consider this as a sign of mobility and we switch to the mode with lower latency. Note that switching between direct and AP-based communication is performed through the TDLS protocol.

IV. Building a MAC Architecture around STPP

We now demonstrate how Soft Scheduling, as instantiated by STPP, can be used as part of a full MAC architecture that adheres to the augmentation principles in Section I. The MAC protocol we describe, Wireless Low-Latency Local Link (WL4), aims at reducing MAC latency for local communication (i.e., where all nodes belong to the same collision domain) using STPP. An overview of the WL4 system architecture is shown in Figure 3. In this section, we present the system's two major components and their implementation.

WL4 Nodes each have two components: *WL4 Client* and the *STPP* module. The *WL4 Client* handles several control functions. It creates the beacons that allow for the construction of the collision domain. This allows it to determine whether

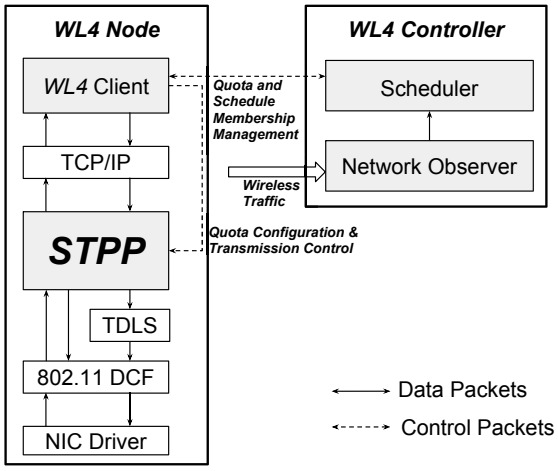

Fig. 3. WL4 system architecture.

a direct or an AP-based link should be used for each local link. It also manages the membership of local links in the STPP schedule by communicating with the controller. It uses the share obtained from the controller to configure *STPP*. It also handles token processing and the decision of resuming transmission. The *STPP* module implements the STPP protocol discussed in the previous section.

WL4 Controller has two components: *Scheduler* and *Network Observer*. The *Scheduler* determines the share of each link in the STPP schedule. This is determined based on the link's Access Category (AC) or its requested share. The scheduler also handles membership in the schedule. In particular, a new link requests to join the schedule and indicates its AC. This messaging is implemented in the application layer over TCP. This approach provides reliability. The *Network Observer* constantly hears traffic belonging to the collision domain, and determines idle nodes. We find that if a node is silent for more than 2 seconds, it should be off the schedule because otherwise the performance of the rest of the nodes starts to deteriorate.

WL4 Implementation takes place by integrating *STPP* by modifying the `mac80211` module in the Linux kernel with a total of 307 lines of code. *STPP* is implemented in the wireless transmission path and performs share accounting and TCP queues checking. This component also has the interface needed to pause and resume queues which is used by other components. We implement the *WL4 Client* as two components: 1) token processing in a Click Modular Router element [41]. 2) A python program that reacts to control messages. This split corresponds to the token being a MAC layer control packet, while the rest of the control messages are treated as application layer messages over TCP. We implement control messages over TCP to ensure reliability and avoid loss. We implement the interface between *STPP* and the *WL4 Client* by extending `wpa_supplicant` to provide WL4 primitives in order to control adding peers, changing their shares, and changing the timeout. The controller is implemented as a simple python program that communicates with Python clients

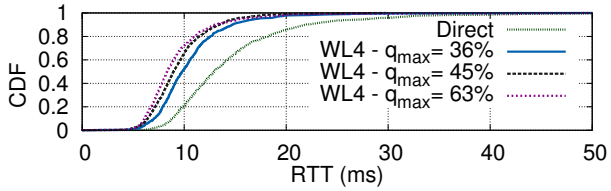

Fig. 4. CDF of RTT observed by the highest priority link of 5 links under three different WL4 share settings compared to the fastest link in an uncoordinated direct setting.

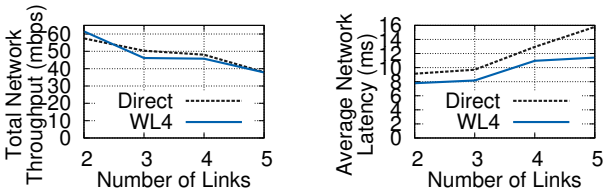

Fig. 5. Effect of number of links on average network throughput (left) and latency (right) for WL4 compared to uncoordinated direct links.

that invoke `wpa_supplicant`.

V. Evaluation

A. Experimental Setup

We conduct experiments on an indoor wireless network deployed in an office on campus. This network coexists with several other networks and is not isolated. The testbed is composed of five desktops with PCIe ath9k AR93xx wireless cards. The testbed also included two Raspberry Pis and a laptop each equipped with a USB ath9k AR9002U wireless cards. We perform all of our experiments in the 2.4Ghz band. However, WL4 does not constrain the band.

We use `netperf` to generate TCP traffic and use the report of `netperf` as achieved throughput. Background traffic for uncoordinated links is generated the same way. In the application evaluation, we implement an augmented reality application using ArUco library [42]. A client node captures images through an attached camera and sends it to a server node for processing. The networking overhead is mainly in the transmission of the 1.7 Mbytes images from the camera to the server. The server replies with 100 byte markers that the node can impose on the captured images which represents minimal networking overhead. We use RTT reported by `ss` (socket statistics) command which represents the average RTT seen by the socket.

We collect RTT measurements every 100 ms. All reported results are the average of 4 experiments of length 100 seconds each. We find this experiment size to be sufficient as variance tends to be very small. We use a Token Expiry Period of zero for all experiments. We discuss the effect of Token Expiry Period values later in this section. We refer to a share in figures using variable q, where q_{max} is the maximum share. We refer to cases where DCF is used without WL4 as *Direct*.

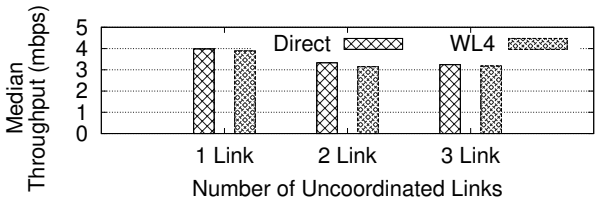

Fig. 6. Effect of number of uncoordinated links on median throughput of a WL4 network of four nodes.

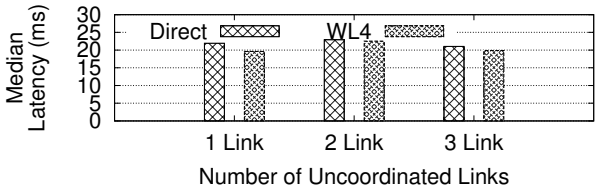

Fig. 7. Effect of number of uncoordinated links on median latency of a WL4 network of four nodes.

B. WL4 Microbenchmark

Impact on high priority links: Figure 4 compares the CDFs of the RTT for the fastest link among five nodes using TDLS and the highest priority link using WL4 using different shares. As the share increases, the latency decreases. WL4 improves median latency by 24%, 30%, and 35% respectively. Moreover, WL4 improves tail latency at the 90th percentile by 32%, 38%, and 40%. It also reduces the 99th percentile by up to 53%.

Impact on average network behavior: We compare the behavior of WL4 and uncoordinated direct communication as we increase the number of links in Figure 5. We compare total network throughput to illustrate utilization of the medium for both approaches and average network latency to show the impact of coordination. Results show that the impact of WL4 ranges from 7% improvement in the case of two links to 9% reduction in the case of three links. However, it is consistently improving latency by at least 17% and up to 38%. This implies that WL4 improves latency with minimal impact on medium utilization.

Impact of non-WL4 devices on WL4 gains: WL4 aims at providing benefits while retaining backward compatibility. This means that while some links will be coordinated using WL4, some other links within the same network can be uncoordinated. Figures 6 and 7 show the effect of uncoordinated links on the performance of four WL4 links compared to their effect on TDLS links. We note that the first uncoordinated link is a two-hop link and the rest are all TDLS. We find that WL4 improves latency by 10%, 1%, and 5% respectively while reducing throughput by 2%, 5%, and 2%. WL4 can improve latency while having negligible effect on throughput in the presence of uncoordinated links. We observe that average network throughput for these cases does not change as the number of links is constant, this requires very few exchanges of nominal rate values.

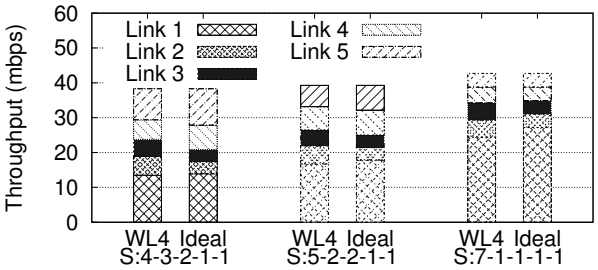

Fig. 8. Division of shares under WL4 compared to ideal share for three division strategies.

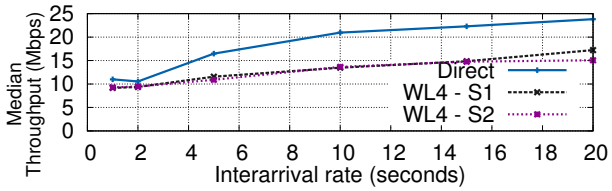

Fig. 9. Effect of arrival rate of a 10 seconds download Poisson processes on median throughput of WL4 network of four nodes.

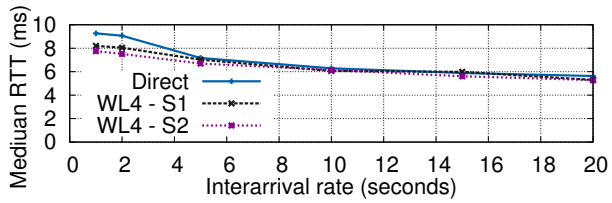

Fig. 10. Effect of arrival rate of a 10 seconds download Poisson processes on median latency of a WL4 network of four nodes.

Share Division: Figure 8 compares the throughput achieved by five links under different share assignments, where the rate is configured for the experiment and not based on demand. It shows the actual achieved rate compared to the ideally achievable rate under the corresponding sharing assignment. The figure also shows that WL4 provides accurate assignment with an average deviation from the assigned share of 1.15 Mbps. This result is achieved by setting slot sizes to correspond to the demand of individual nodes. This sets the rate of operation for each node.

Impact of traffic patterns: We model traffic as a Poisson process with varying the rate of interarrival time. Each traffic event is of length 10 seconds. Experiments here are conducted for four nodes. The process on each node is independent but they all have the same value of the rate of the interarrival time. We compare uncoordinated direct communication with WL4 under two share assignments (S1:4-4-4-4 and S2:4-3-2-1). Figures 9 and 10 show the impact of traffic interarrival time on median throughput and latency. We use median throughput rather than total throughput, as the total throughput in this case does not represent actual medium utilization since links' transmission do not necessarily overlap. As expected, for small

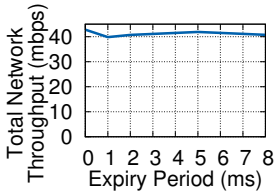

Fig. 11. Effect of Token Expiry Period on median latency of a WL4 network of four nodes.

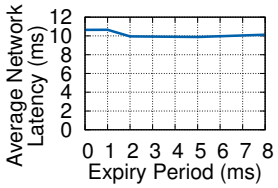

Fig. 12. Effect of Token Expiry Period on median latency of a WL4 network of four nodes.

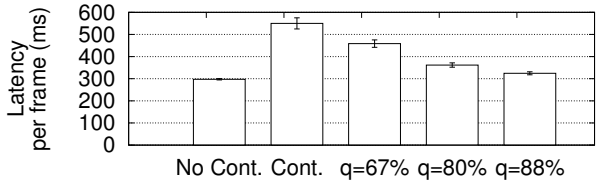

Fig. 13. Latency per frame for an augmented reality application when no other nodes compete for the medium with it, with uncoordinated contention with one link, and for coordinated contention such that the application gets 2x, 3x, 4x, and 8x medium access compared to background traffic.

interarrival times of 2 seconds, contention is high and WL4 provides significant latency benefits of 17% reduction with 10% reduction in throughput in case of S2. However, as the interarrival time increases, contention is reduced and token passing represents only overhead. Hence, we recommend that if a node is not transmitting for 2 seconds, the controller should take it off the schedule.

Effect of Token Expiry Period: In Section III, we introduced Token Expiry Period as the means to move the system into synchronized coordination. Figures 11 and 12 show the effect of changing the Token Expiry Period from 0 to 8 which is the maximum share. We find that, while it improves latency (due to better coordination), it hurts throughput very little (due to added idle time). We recommend using a value of zero which we used in all other experiments. This allows WL4 to improve latency while having negligible effect on throughput.

C. WL4 and a Low Latency Application

Figure 13 shows the performance of an augmented reality application in terms of latency per frame. We first measure the performance of the application with no background traffic in the network and all traffic was transmitted through a direct link (No Cont.). This provide a baseline for performance of the application that cannot be exceeded of 297 msec per frame. We also measure the performance of the application when it competes with background traffic through CSMA/CA (Cont.). This provides an upper bound on the latency that WL4 should not exceed. Figure 13 shows that when WL4 is used to give the application a higher share and hence more access to the medium, the application can reach a latency that is only 10% worse than its best performance while allowing background traffic to operate at a reasonable rate.

Figure 14 shows the value of having WL4 periodically compare AP-based links to direct links to optimize application

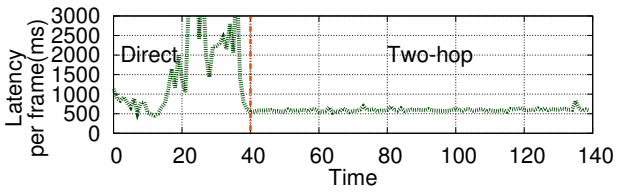

Fig. 14. Latency per frame for an augmented reality application when the client and server are placed far apart. WL4 disables the direct link and communicates through the AP which significantly improves performance.

performance. As the client moves away from the server, the channel conditions between the two nodes degrade, which forces a lower bit rate and higher loss. However, the link to the access point remains of good quality. As shown in Figure 14, realizing such situations and switching back to AP-based communication can improve the performance by 3x.

VI. Discussion

WL4 Security: There are two types of attacks that WL4 can face: 1) attacks on the controller, and 2) malicious nodes. Protection of the controller is similar to the protection of access points. Recall that WL4 relies on TDLS, and inherits all of its security features. As for malicious nodes, such nodes will use the medium aggressively beyond their allocated slots. However, this behavior can be detected by the controller and those nodes can be taken off the schedule. We have shown that WL4 can still result in performance gains in the presence of such uncoordinated nodes. Another attack by malicious nodes is falsely declaring their Access Category. Such violations can be detected by the controller through occasional checking of port numbers used by the transport protocol and comparing them to known or reserved port numbers.

WL4 Multihop Communication: Communication through the access point was shown to incur high latency even with a lightly loaded access point. Our initial experiments with a university campus enterprise wireless network show that latency experienced on such networks for local communication can reach hundreds of milliseconds. The work in Section III-F can be extended to accommodate choices with more that two hops. We envision that WL4 can be used to lower latency of enterprise wireless networks by forming a mesh network of devices connected to the enterprise network. WL4 can make use of earlier work on mesh network and mobile adhoc networks to find the best routes between source and destination and maintain a connected network.

Integration with Wireless SDN Systems: WL4 can be integrated into Wireless SDN systems (e.g., OpenSDWN [43]) by allowing such systems to communicate their schedules to the WL4 controller which allows full control on the behavior of all L4s. Services provided by OpenSDWN like service differentiation and participatory networking can be communicated to all nodes and executed in a distributed fashion. We also note that other services offered by OpenSDWN are orthogonal to WL4 and can operate normally without the need to be modified or interact with WL4 (e.g., handling node mobility and roaming between access points).

Scheduling Transport Layer Flows within a WL4 Link: WL4 focuses on scheduling multiple links in the MAC layer. However, it assumes that this link either contains a single transport (TCP or UDP) flow, or multiple flows with the same access category. Differentiation of service between multiple flows within a single link can be achieved through traffic control at the source. Queuing disciplines such as HTB can be used to provide differentiation of service between multiple flows.

VII. Concluding Remarks

In this paper, we argue that simple high level augmentation of WiFi is the only feasible, deployable solution for large scale improvement of MAC layer performance. This improvement is needed to meet application demands. We develop a set of principles that this augmentation needs to adhere to. We argue that an introduction of scheduling transmission mechanisms has the potential to endow WiFi with significant additional functionality to allow it to respond to these rising application demands. We introduce the idea of soft scheduling as a means to maintain interoperability, transparency, and ease of deployment. We propose and develop the Soft Token Passing Protocol (STPP) and demonstrate how it can be used to construct a full MAC protocol, WL4, that we implement to reduce link latencies in WLANs.

Further research is needed to better understand the fundamentals of soft scheduling and its use to construct full MAC protocols. Are there other techniques that can be used to provide additional MAC features? How can one use these techniques to target other performance improvements for WiFi deployments? Are there limits to our ability to augment WiFi protocols and still adhere to the principles of Section I?

References

[1] S. Mangold, S. Choi, P. May, O. Klein, G. Hiertz, and L. Stibor, "IEEE 802.11 e Wireless LAN for Quality of Service," in *European Wireless*, 2002, pp. 32–39.

[2] B. P. Crow and et al., "IEEE 802.11 wireless local area networks," *IEEE Communications magazine*, vol. 35, no. 9, pp. 116–126, 1997.

[3] "IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pp. 1–3534, Dec 2016.

[4] A. Essameldin and K. A. Harras, "The hive: An edge-based middleware solution for resource sharing in the internet of things," in *Proceedings of the 3rd Workshop on Experiences with the Design and Implementation of Smart Objects*. ACM, 2017, pp. 13–18.

[5] A. Saeed, A. Abdelkader, M. Khan, A. Neishaboori, K. A. Harras, and A. Mohamed, "Argus: realistic target coverage by drones," in *Information Processing in Sensor Networks (IPSN), 2017 16th ACM/IEEE International Conference on*. IEEE, 2017, pp. 155–166.

[6] M. Khan, K. Heurtefeux, A. Mohamed, K. A. Harras, and M. M. Hassan, "Mobile target coverage and tracking on drone-be-gone uav cyber-physical testbed," *IEEE Systems Journal*, 2017.

[7] H. Abdelnasser, K. A. Harras, and M. Youssef, "Ubibreathe: A ubiquitous non-invasive wifi-based breathing estimator," in *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*. ACM, 2015, pp. 277–286.

[8] ARCore - Google Developers, https://developers.google.com/ar/, 2017.

[9] ARKit - Apple Developer, https://developer.apple.com/arkit/, 2017.
[10] The Machines on the App Store - iTunes - Apple, https://itunes.apple.com/us/app/the-machines/id1280682965, 2017.
[11] chromecast, https://www.google.com/chromecast, 2017.
[12] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, pp. 14–23, 2009.
[13] A. Saeed, M. Ammar, K. A. Harras, and E. Zegura, "Vision: The case for symbiosis in the internet of things," in *Proceedings of the 6th International Workshop on Mobile Cloud Computing and Services*. ACM, 2015, pp. 23–27.
[14] K. Habak, C. Shi, E. W. Zegura, K. A. Harras, and M. Ammar, "Elastic mobile device clouds: Leveraging mobile devices to provide cloud computing services at the edge," *Fog for 5G and IoT*, p. 159, 2017.
[15] K. Habak, E. W. Zegura, M. Ammar, and K. A. Harras, "Workload management for dynamic mobile device clusters in edge femtoclouds," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*. ACM, 2017, p. 6.
[16] "Ieee standard for information technology–telecommunications and information exchange between systems–local and metropolitan area networks–specific requirements–part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," *IEEE Std 802.11ac(TM)-2013*, pp. 1–425, Dec 2013.
[17] K. Nishat and et al., "Slickfi: A service differentiation scheme for high-speed wlans using dual radio aps," in *CoNEXT*, 2016.
[18] P. Djukic and P. Mohapatra, "Soft-tdmac: A software tdma-based mac over commodity 802.11 hardware," in *INFOCOM '09*. IEEE, 2009, pp. 1836–1844.
[19] A. Rao and I. Stoica, "An overlay mac layer for 802.11 networks," in *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '05. New York, NY, USA: ACM, 2005, pp. 135–148. [Online]. Available: http://doi.acm.org/10.1145/1067170.1067185
[20] Z. Yang and et al., "Enabling tdma for today's wireless lans," in *INFOCOM '15*, 2015.
[21] Z. Zeng, Y. Gao, K. Tan, and P. Kumar, "Chain: Introducing minimum controlled coordination into random access mac," in *INFOCOM '11*, 2011, pp. 2669–2677.
[22] V. Shrivastava, N. Ahmed, S. Rayanchu, S. Banerjee, S. Keshav, K. Papagiannaki, and A. Mishra, "Centaur: realizing the full potential of centralized wlans through a hybrid data path," in *MobiCom '09*, 2009, pp. 297–308.
[23] W. Zhou and et al., "Domino: relative scheduling in enterprise wireless lans," in *CoNEXT '13*, 2013, pp. 381–392.
[24] C.-F. Shih and et al., "Look Who's Talking: A Practical Approach for Achieving Scheduled WiFi in a Single Collision Domain," in *CoNEXT '15*, 2015.
[25] H. Wu, Y. Liu, Q. Zhang, and Z. l. Zhang, "Softmac: Layer 2.5 collaborative mac for multimedia support in multihop wireless networks,"
[32] M. Hadded, P. Muhlethaler, A. Laouiti, R. Zagrouba, and L. A. Saidane, "TDMA-based MAC protocols for vehicular ad hoc networks: a survey, qualitative analysis, and open research issues," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2461–2492, 2015.
IEEE Transactions on Mobile Computing, vol. 6, no. 1, pp. 12–25, Jan 2007.
[26] T. Høiland-Jørgensen and et al., "The good, the bad and the wifi: Modern aqms in a residential setting," *Computer Networks*, 2015.
[27] C. Pei, Y. Zhao, G. Chen, R. Tang, Y. Meng, M. Ma, K. Ling, and D. Pei, "Wifi can be the weakest link of round trip network latency in the wild," in *INFOCOM '16*, 2016, pp. 1–9.
[28] R. Kateja, N. Baranasuriya, V. Navda, and V. N. Padmanabhan, "Diversifi: Robust multi-link interactive streaming," in *CoNEXT '15*, 2015, pp. 35:1–35:13.
[29] C.-Y. Li, C. Peng, S. Lu, X. Wang, and R. Chandra, "Latency-aware rate adaptation in 802.11 n home networks," in *INFOCOM '15*, 2015, pp. 1293–1301.
[30] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM (JACM)*, vol. 20, no. 1, pp. 46–61, 1973.
[31] S. Lee, S. Banerjee, and B. Bhattacharjee, "The case for a multi-hop wireless local area network," in *Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, vol. 2. IEEE, 2004, pp. 894–905.
[33] P. Huang, L. Xiao, S. Soltani, M. W. Mutka, and N. Xi, "The evolution of MAC protocols in wireless sensor networks: A survey," *IEEE communications surveys & tutorials*, vol. 15, no. 1, pp. 101–120, 2013.
[34] "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," *IEEE Std 802.11z-2010*, pp. 1–96, Oct 2010.
[35] I. Cidon and O. Mokryn, "Propagation and leader election in a multihop broadcast environment," in *International Symposium on Distributed Computing*, 1998, pp. 104–118.
[36] E. Dumazet and J. Corbet, "TCP small queues," https://lwn.net/Articles/507065/, 2012.
[37] I. Rhee and et al., "Drand: distributed randomized tdma scheduling for wireless ad-hoc networks," in *MobiHoc '06*, 2006.
[38] W. Bux and et al., "Architecture and design of a reliable token-ring network," *IEEE JSAC*, vol. 1, no. 5, pp. 756–765, 1983.
[39] A. Balasubramanian, R. Mahajan, and A. Venkataramani, "Augmenting mobile 3g using wifi," in *MobiSys '10*, 2010, pp. 209–222.
[40] R. Draves, J. Padhye, and B. Zill, "Comparison of routing metrics for static multi-hop wireless networks," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 133–144, 2004.
[41] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The click modular router," *ACM Transactions on Computer Systems (TOCS)*, pp. 263–297, 2000.
[42] R. Munoz-Salinas, "Aruco: a minimal library for augmented reality applications based on opencv," *Universidad de Córdoba*, 2012.
[43] J. Schulz-Zander, C. Mayer, B. Ciobotaru, S. Schmid, and A. Feldmann, "Opensdwn: Programmatic control over home and enterprise wifi," in *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*. ACM, 2015, p. 16.