Automatic Quality of Experience Management for WLAN Networks using Multi-Armed Bandit

Henrique D. Moura, Daniel Fernandes Macedo, Marcos A. M. Vieira

Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte, MG – Brazil
E-mail:{henriquemoura, damacedo, mmvieira}@dcc.ufmg.br

***Abstract*—Providing acceptable *Quality of Experience* (QoE) in *Wireless Local Area Network* (WLAN) is very difficult: home networks are managed by non-technical people, and the proprietary management solutions of enterprise networks usually do not incorporate QoE mechanisms. Due to these difficulties, automatic QoE management mechanisms are welcome. This paper presents control loops capable of changing the power and transmission channels in the WLAN, based on Software Defined Wireless Networks and reinforcement learning, in order to improve user satisfaction for Web applications. A prototype evaluates our proposal in three case studies with a web browsing application, in which several access points are controlled by a central controller or by independent controllers. Our results show that the control loop can improve the *Mean Opinion Score* (MOS) by at least 4 % in the worst case, and 167 % in the best case, thus benefiting the user. Further, the control loop also reduced in page load time by 25 % in the worst case, and 233 % in the best case.**

***Index Terms*—Wireless Networks, Software Defined Networks, Reinforcement Learning, Multi-Armed Bandit, Quality of Experience**

I. INTRODUCTION

Recent estimates suggest that more than 10 billion Wi-Fi devices have been sold and more than 4.5 billion of those are in use today [1]. WLAN is the most common method of Internet access for home stations [2], but its transmission medium is subject to performance problems, and the home user is a non-technical person. Many home WLAN *Access Point* (AP) are provided by *Internet Service Provider* (ISP) to home clients, and ISPs have the expertise to analyze the data generated by these devices, however, most of the time they avoid monitoring home traffic because of privacy and cost issues [3]. Companies use commercial wireless controllers that provide WLAN central configuration and management, but they are closed platforms, which rely on the vendor's initiative to provide new features. Faced with so many challenges, there is a need to propose new and automatic control mechanisms to wireless devices that improve the user's quality of experience.

Auto-configuration and auto-optimization capabilities can be added to the wireless network using *Software Defined Networking* (SDN) approaches [4, 5] with minimal or no human administration [4]. To achieve that, two prerequisites must be fulfilled: a control loop algorithm that gathers information from the environment and acts on the devices, and some metric that is representative of the satisfaction perceived by the user when using the network application, called QoE.

978-3-903176-15-7

In the literature, the solutions use metrics obtained at the stations, the servers, or the network nodes to estimate the QoE perceived by the user. Hora et al. [6] studied the QoE of a web browsing application on WLANs. They created a regressor that relates WLAN metrics perceived at the AP to the QoE perceived by the wireless user, so that ISPs do not infrige the user's privacy. Proposals that use QoE, focus on local decisions [7–12], or use some communication mechanism between the agents in the network [13–15]. To the best of our knowledge, no proposal does active control of the wireless network considering interfering APs, using metrics capable of estimating user's satisfaction. This is important because it is the trending scenario addressed by operators or network administrators nowadays.

This paper proposes two closed control loops for WLAN using SDN and a machine learning technique called *Multi-armed Bandit* (MAB), which optimizes the Web QoE proposed in [6]. We evaluated the control loop using *Ethanol* [16], an SDN architecture for IEEE 802.11 networks. To the best of our knowledge, our work is the first that uses SDN to control 802.11-based wireless networks using MAB with a QoE metric as feedback, and also performs real life experiments. The control loops are evaluated in a prototype under three scenarios: (i) a single controller manages an AP with stations connected to it; (ii) the APs are managed by independent controllers; and (iii) one controller coordinates all the APs. Results show that our control loops improve the QoE metric, reducing the average regret at least by 45 % in the worst case, and 84 % in the best case. It reduced the page load time by 25 % in the worst case, and 233 % in the best case, when compared to the baseline.

The remainder of this paper is organized as follows. Section II shows the background. Section III describes the proposed architecture. Section IV discusses the evaluation results. Related work is discussed in Section V. Finally, Section VI draws the conclusions and presents the future work.

II. BACKGROUND

This section discusses some concepts related to our proposal. It presents how to measure user satisfaction, the SDN paradigm, the reinforcement learning method used in our work, called MAB, and explains how the regret is calculated.

A. Ethanol

Ethanol is a SDN platform that allows the direct programming of network services by providing high level calls that control the underlying infrastructure of IEEE 802.11. It allows us to meet requirements like flexibility in the management and configuration, adaptability and independence of the supplier in wireless networks [16]. *Ethanol* defines a southbound interface that controls IEEE 802.11 APs as well as wireless stations that fully implement the IEEE 802.11/2016 standard (former 802.11k, 802.11m, and 802.11v amendments). *Ethanol* is available under *General Public License* (GPL) version 2. The source code can be found at https://github.com/h3dema/ethanol_hostapd for the AP, and https://github.com/h3dema/ethanol_controller for the controller.

Ethanol employs extensions of IEEE 802.11 to obtain station performance data, as well as to execute monitoring commands, so that the SDN controller obtains information about the observed link quality for each device. The use of *Ethanol* allows the controller to centrally request network usage metrics to the APs it manages, as well as to the client nodes. In this way a control algorithm can, using these metrics, adapt network parameters (such as the channel frequency, the transmission power, among others) to improve the perceived performance by the wireless network users.

B. Estimating Web QoE in Wi-Fi

QoE approaches were introduced to measure the pleasure or the discomfort perceived by the user in the use of a service, including system, human, and context factors. Obtaining the QoE is complex, since the individual experience is subjective in nature, being subject to the expectations and perceptions of the user. It is also difficult to quantify and measure, since it is impossible to force the user to make an evaluation of a service, and furthermore this evaluation is objective. Research has been conducted at the academy on the measurement of QoE for web browsing. Estimators are applied directly to browsing [6, 17] or file transfers [18].

Hora et al. [6] build a predictor that can be used by an ISPs in order to relate WLAN metrics to QoE of a user browsing the web. For that reason, they use two WLAN metrics available in APs that can coarsely correlate to the MOS: average PHY rate and the percentage of medium busy time. The first one represents the link quality, and the second, the medium availability. The MOS value is generated using a support vector regressor.

Hora et al.[6] instrument a commodity AP, and passively monitor WLAN metrics. They tried to infer the relationship between WLAN metrics and QoE, through controlled web browsing experiments, in a WLAN testbed. Their results were validated using data collected in an office environment, with data collected every 30 minutes, from 6AM to 11PM, for two weeks. To account for page complexity, Hora et al. ranks sites into three categories - *light*, *average*, and *heavy*. Their predictors (one *Support Vector Regression* (SVR) for each type of site) adhere to the MOS provided by a panel of users up to 93% in their validation set. They also made measurements on APs deployed in 4,880 residential customers of a large Asian-Pacific ISP, reporting over 23,000 devices to a backend server, collecting a total of 180 million samples. Because of its thorough evaluation, we decided to use their predictor as a feedback to our control system.

C. Multi-armed Bandit – MAB

Multi-armed bandit, introduced in the seminal paper of Lai and Robbins [19], is a special class of the more general paradigm of *Reinforcement Learning* (RL) for sequential optimization problems [20]. This class of learning algorithms allows an agent to interact with an unknown environment over a series of steps, observing the current state of the environment, taking actions, and receiving as feedback a scalar reward (in our case, the MOS). MAB assumes that the feedback is limited, thus learning uses a trial and error strategy [20, 21]. Also, due to the limited feedback, MAB presents a trade-off between exploration and exploitation.

Historically, the name "bandit" comes from a gambler who plays a slot machine in a casino. The casino does not want to lose money, hence the analogy of the machine being a bandit who gets the gambler's money. The problem considers that the player can bet on K machines, so she[1], with each move (time step), pulls the slot machine's lever (arm). Each time an arm is pulled, a random reward, regardless of any previous reward, is returned. The K arms (actions) are considered independent of each other. The player's objective is to accumulate the maximum reward possible in the long run.

Formally, the MAB problem is described as: Let K be the number of arms the player can pull. At each time step t, also called stage, the player (agent) pulls an arm, returning a scalar reward associated with this arm. Let the state of arm i at step t be $s_t^{(i)}$, then, if the agent selects arm $j = m(t)$ at time t, the states are updated as follows:

$$s_{t+1}^{(i)} = \begin{cases} s_t^{(i)} & i \neq j \\ T_i(s_t^{(i)}, w) & i = j \end{cases} \tag{1}$$

where $T_i(s_t^{(i)}, w)$ is a function that describes the (possibly stochastic) transition probability of the *i-th* arm, and accepts the state of the *i-th* arm, and a random disturbance w. Rewards are taken from an unknown distribution. We can say that the reward received by the player at time t is a function of the current state and a random element: $r_i(s_t^{(i)}, \omega)$. The process repeats over a $\top$ steps horizon. The player's goal is to maximize the sum of the (discounted) rewards. Further, in MAB the objective of the agent is to maximize the cumulative discounted reward.

A policy is a decision rule for selecting arms as a function of the state of the arms. Gittins [22] showed that there exists an optimal index policy. Thus, there is a function that maps the state of each arm to a scalar (real number) called the "index", such that the optimal policy is to choose the arm with the highest index at any given time. This index value

[1] In this paper, the pronoun *she* is used for an unknown referent to balance out the perceived sexism of the generic *he*.

reflects the expected reward estimate for each arm, as well as the estimate's confidence, thus, when using this index, the algorithm exploits the arm with the highest estimated reward, but also explores arms whose confidence in the estimate is low, since these arms can offer better rewards in the long run.

Regret

A common performance measure for bandit algorithms is the total expected regret. Regret measures the absolute difference between the sum of the rewards obtained by the strategy adopted by the agent, that is, of the rewards obtained by the sequence of arms actually selected, and the optimal strategy. In this way, it measures how much the agent lost by not choosing the best arm every time. It is defined for any fixed round of $\top$ steps as the difference between the reward that would be obtained when using an optimal policy π^*, and the sum of the estimated average reward values $\hat{r}_t^j$ actually obtained in each step by selecting arm j in step t:

$$R_T = V^{\pi^*} - \sum_{t=1}^{T} \hat{r}_t^{(j)} = r^* T - \sum_{j \in [1,K]}^{T} \hat{r}^{(j)} \mathbf{E}[N_j] \quad (2)$$

where $\mathbf{E}[N_j]$ is the expected number of times arm j will be selected. Therefore regret measures how much reward, on average, the system is losing by not achieving the maximum reward.

III. System Architecture

We propose a closed-loop control system for wireless networks, using reinforcement learning and SDN to improve Web application QoE. For that end, the devised control loop changes either the transmission power or the wireless channel being used by the APs.

Although simple, choosing the best channel and the highest transmission power does not optimize the QoE for the following reasons. One of the options to improve the performance of web applications that may seem simple is to optimize the flow of the stations. This approach seems to work at first glance, yet it is simplistic, and does not solve the problem from the user's point of view, because there is a limit amount of system throughput, that is, it limited by wireless transmission capacity using the IEEE 802.11 protocol, and the users' needs are regulated by their perception of the rendering time for each type of site. Even so if all users were wishing to access the same type of site, their perception is influenced by the page load time, which varies for example with the wireless connection bit rate, noise in the link between the AP and the station, or other factor. In this way, maximizing throughput can affect from what is perceived by each user, because one station can interfere with another, specially in a multi-agent environment.

Figure 1 shows the proposed architecture, where the Web QoE control loop sits on top of the *Application Programming Interface* (API) provided by *Ethanol* [16], thus running as a network management service. We will discuss the algorithm used in the next section. It is worth noting that the control loop does not require changes to the user's applications or devices. The control loop depends on the classification of the site type of the web browsing flow, since there are three predictors, one for each type of site. Such a classifier uses flow data obtained during the OpenFlow "Packet In" event to read the parameters needed to classify the type of site being accessed.

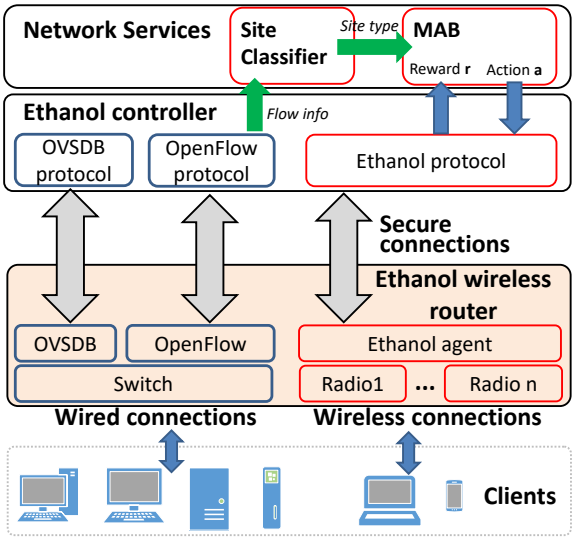

Figure 1: Reinforcement Learning control loop on top of *Ethanol*

We rely on a data plane composed by APs managed by *Ethanol*. Our learning algorithm can use any IEEE 802.11 "actuators" that *Ethanol* exports, such as the channel frequency, the transmission power, MAC protocol parameters (RTS/CTM usage, DTIM, etc.), QoS parameters, among others.

MOS: The regressor provided by Hora et al. [6] returns the predicted MOS.Since the MOS values are discrete, and the values read from the APs are continuous, the MOS used in the experiments is calculated as a linear regression of neighbor points of the read value as show in Figure 2. The predictor is used as a black box by the model, thus if a better or more suitable predictor appears, it can be used instead.

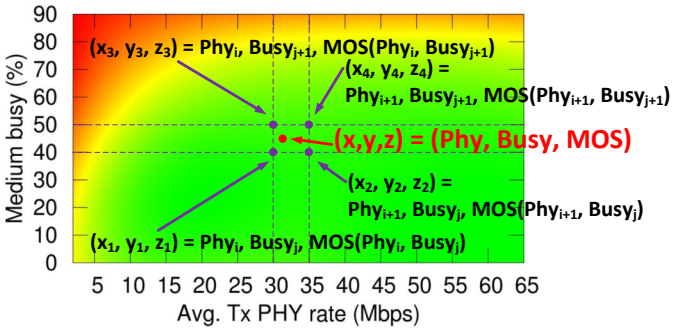

Figure 2: How MOS is calculated in our evaluation

Reward: The reward measures how much the new system configuration increases or decreases the QoE. For a time t, a wireless station j perceives a MOS equal to $MOS_{j,t}$, calculated using the predictor. The reward used in line 12

of Algorithm 1 is the average MOS of all the stations $\left(r_t = \frac{1}{m}\sum_{j=1}^{m} MOS_{j,t}\right)$, where m is the number of the stations. Thus our control loop seeks to maximize the average MOS. Another option, which we intend to evaluate in future work, is the use of Jain's definition of fairness [23] to guarantee a more balanced MOS, i.e., penalize situations in which the results for each client differ a lot, therefore making the overall result more homogeneous.

Actions in Multi-armed Bandit: The proposed control loops actuate on the system by changing the wireless channel, and the transmission power. In MAB, the actions are mapped to the arms, so we have an arm for each combination of (AP, channel number, transmission power). The transmission power is discretized into 1 dBm increments. Thus the total number of arms K, using our devices, is n AP $\times$ 11 channels $\times$ 15 power levels, i.e. it scales linearly on the number of controlled access points. 5 GHz band was not used in the paper, however the model supports any channel or frequency available.

Algorithm: Our proposal uses the ϵ-greedy approach, which is widely used because it is very simple. There are many approaches to MAB, but our proposal uses a class of MAB algorithms called *Upper Confidence Bound* (UCB), proposed by Auer et al. [24], in particular the algorithm uses UCB1 [24]. UCB is an elegant implementation of the idea of optimism in the face of uncertainty, as proposed by Lai and Robbins [19].

In Algorithm 1, the pseudo-code selects a deterministic policy at each step t. In line 3, the estimated average value $\hat{r}^{(j)}$ of each arm is initialized to one, the lowest MOS possible. Another option would be to start with the maximum MOS value, but that implies in forcing the algorithm to explore all arms in the first steps. In a transfer learning experiment, this value can be initialized using previous knowledge, i.e. values learned in previous runs. We intend to evaluate this in future work. $n_t^{(j)}$ is the number of times the arm *j* is pulled, and in the initialization, it is set to one for all arms. In time step t, the algorithm selects the arm j that maximizes the index $I^{(j)}$. The environment returns a scalar reward associated with the unknown reward distribution of the selected arm j.

In UCB1, after an arm j is selected and the reward r_t is received, the UCB1 indexes, the average reward and the number of selections of arm j are updated using the equations in Lines 11, 12, and 13. $I_t^{(i)}$, $n_t^{(i)}$ and $\hat{r}_i(t)$ retain the previous values, for $i \neq j$. $\hat{r}^{(j)}$ is the average reward obtained from arm j, $n^{(j)}$ is the number of times arm j has been pulled so far, and t is the overall number of plays done so far.

As the horizon is not known, this algorithm uses the "doubling trick" described in Section 2.3 of [25]. The idea is to partition time into periods of exponentially increasing lengths. When the period T ends, $n^{(j)}$ is reset and then is started again in the next period, and the length of T is doubled.

Algorithm 1 UCB algorithm

1: **function** UCB(K) ▷ K: number of arms
2: ▷ $n^{(i)}$: number of times the arm i was pulled
3: Initialize $I_t^{(j)} = \hat{r}_t^{(j)} = n_t^{(j)} = 1, \forall j \in [1, K], t = 0$
4: **while** $True$ **do**
5: $T \leftarrow 1$
6: **for** $t \in [1, T]$ **do**:
7: ▷ Select the arm j that maximize the index
8: $j = \max_j I_{t-1}^{(j)}$
9: Take action j ▷ Play the arm j
10: Observe arm j's $MOS_{j,t}$ e calculate r_t
11: update $n_t^{(j)} = n_{t-1}^{(j)} + 1$
12: update $\hat{r}_t^{(j)} = \frac{n_{t-1}^{(j)} \times \hat{r}_{t-1}^{(j)} + r_t}{n_t^{(j)}}$
13: update $I_t^{(j)} = \hat{r}_t^{(j)} + \sqrt{\frac{2\log(t)}{n_t^{(j)}}}$
14: $T \leftarrow 2 \times T$
15: $n^{(j)} \leftarrow 1, \forall j \in [1, K]$

IV. Control Loop Evaluation

This section presents three experiments used to test the control loop using four metrics: convergence time, page load time, average MOS, and regret.

A. Scenarios

Our experiments aim to evaluate how the control loop would work under the operating conditions found in production wireless networks. The hypothesis that we evaluate is whether the devised control loop improves the user's QoE. This is tested under three different scenarios. The first one is when the control loop only controls the user's AP, like in a residence, and in this way it has to adapt to the interference generated by the neighboring networks. Next, we evaluate the performance when several neighboring intelligent networks, with independent controllers (such as in a commercial building with several companies), coexist in the environment. And finally, we evaluate the performance of several neighboring intelligent networks being controlled by a single (logical) controller. This is similar to a large company or campus.

1) Stand-alone (SA) experiment: This experiment uses one controller, which manages one AP and two clients. The control is centralized, so this is a RL scenario with a single agent. In this way, the algorithm in the controller is executed on a state space that contains both stations, and therefore it seeks a global reward that maximizes the MOS of both stations. All feedback is obtained by reading information from the environment.

2) Multi-agent (MA) experiment: This scenario simulates a place with multiple APs, such as a shopping center, a condominium, etc., and each AP is managed by a different administrator who does not exchange information with other administrators. We will show that it is still possible to obtain benefits from the use of our control loop for the stations connected to the stations of the various networks. This is a classical case of multi-agent RL where the agents do not communicate, but have to cope with the cross-interference.

This experiment uses two independent controllers. Each controller manages one AP. The APs are near each other (less than one meter apart), so are the stations. The APs are started on the same channel and with the same power. This

experiment highlights the effects of adding intelligence to the system, which can reduce the cross interference using learning algorithms.

3) Centrally-controlled Multi-Agent (CA) experiment: This scenario simulates a place with multiple APs managed by a single administrator, such as in a company, a campus, etc. We will show that the control loop generates benefit for stations connected to the wireless network, even when there are neighboring APs, since the controller makes coordinated decisions to improve the global reward.

In this experiment, a single controller manages two APs. This experiment is similar to the *Stand-alone* (SA) experiment because the controllers can only obtain information by reading the environment, but now the controller coordinates two devices and the controller has to handle the cross interference that is generated by the station transmission to their respective APs.

B. Methodology

The scenarios contain APs that run a modified *HostAPd*, which runs the *Ethanol* agent. The AP runs NAT so its clients can access the Internet. The wireless stations are Linux computers equipped with a wireless card compatible with IEEE 802.11b/g/n. No software is needed on the stations, relying only on IEEE 802.11 standard support. As the AP may change the channel several times, we use stations that recognize *Channel Switch Announcement message* (CWAP) messages, since CWAP is used by an AP in a *Basic Service Set* (BSS) to advertise the new channel, before changing channels. This way the station recognizes in advance the channel change, and can accelerate the migration to the new channel, reducing the disconnection time.

The traffic is not simulated, i.e, the stations download live pages over the Internet. Each client runs an command line web browser that continuously downloads the default web page from the three sites that listed in the Hora et al.'s paper. The browser requests a new default page when the current request is completed. The page load time considers the download of the whole page, i.e, the html file, scripts, RSS, and images.

The controllers are connected to a gigabit Ethernet network, just like the stations and the APs, as shown in Figure 3. The controllers and the stations use the Ethernet as an "experiment control plane", and as such no application traffic is sent over the wired interface. The APs use the Ethernet to receive control data from the controller but also to forward the station traffic (download) to the Internet. The devices are aligned, and the distances among them are shown in Figure 3.

Combination of sites: Because Hora et al. rank sites into three categories (*light* – Google, *average* – Facebook, and *heavy* – Amazon) and since we have two clients, we run the experiment for the six possible traffic combinations. For each combination, the experiment is repeated 30 times with a timeout of 30 minutes.

Online classification of the flow by site type: In order for our proposal to work in an online environment, it must be able to classify the web traffic passing through the AP in one of the three site types defined in Hora et al., so the correct MOS regressor is selected. For the experiments, as the stations accessed the three sites defined by the authors, it was enough to use the OpenFlow "*Packet In*" event to identify the destination IP, as shown in Figure 1, and, with this information, inform the learning algorithm the site type using a simple mapping table. We are working in a semi-supervised learning classifier that employs only data accessible from the "PacketIn" events to do online classification of the sites.

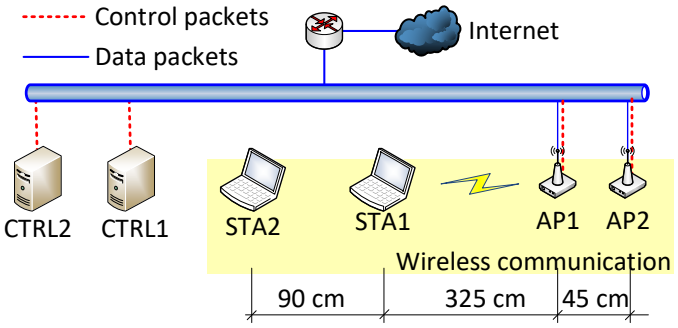

Figure 3: Network layout of the experimen

Devices description: The controllers are dual core Intel(R) i5 PCs with 16GB of RAM. The stations are PCs with dual core Pentium(tm) 2.4GHz CPU, 2GB of RAM and a RT3062 802.11n wireless card. Because *Ethanol* runs on Linux, our APs are ASUS notebooks with Intel(R) i7 CPU @ 1.80GHz, 8GB of RAM and Atheros AR9485 adapters. All computers run Ubuntu Linux version 14.04 LTS.

Baseline: As a baseline to compare the results obtained using the controller, for each experiment configuration, we execute 20 runs without the controller actuation for 30 minutes. Executions are also made for the six combinations of site types. APs are started on a randomly selected channel, and with maximum transmission power. The experiments are performed in an environment with more than 60 other APs.

C. Results

This section presents the results for the three experiments used to test the control loop. We used four metrics: convergence time, page load time, MOS, and regret (recall section II-C for the definition of regret).

1) Convergence Time: Table I shows the number of iterations of the algorithm so that both wireless stations reach the maximum MOS value, i.e. both reach $MOS = 5$. We show in the table rows the results for the site type combinations, followed by the average number of iterations with the 95% confidence interval and the median. Each iteration occurs in approximately one second, thus the number of iterations also represents an approximation of the time spent to reach the maximum on both stations. There is one sampling per iteration, then there are 1,800 sampling per run We did not present in Table I the convergence time for the baseline, because the baseline does not employ a control scheme (max transmit power and fixed channel), so there is no convergence time. When a light page is requested, the convergence time tends to be longer than the other cases, especially in the *Multi-agent*

Table I: Comparing the convergence time

Client 1	Client 2	SA Experiment		MA Experiment		CA Experiment	
		Number of Iterations	Median	Number of Iterations	Median	Number of Iterations	Median
light	light	32.8 ± 56.46	**2.0**	1356.4 ± 233.31	1515.0	383.5 ± 168.94	223.0
light	average	166.6 ± 126.84	**2.0**	1309.7 ± 243.56	1255.0	60.9 ± 37.33	2.0
light	heavy	195.7 ± 177.47	**2.0**	1123.9 ± 213.27	1205.0	226.2 ± 57.65	193.0
average	average	6.9 ± 8.08	**1.0**	860.2 ± 171.32	803.5	8.9 ± 6.38	3.0
average	heavy	243.5 ± 279.73	**3.0**	804.4 ± 150.03	840.5	211.6 ± 42.50	202.0
heavy	heavy	100.7 ± 44.15	**69.0**	635.5 ± 108.77	679.5	199.7 ± 37.65	184.0

Table II: Comparing the regret with the baselines

Client 1	Client 2	SA experiment		MA experiment		CA experiment	
		MAB	Baseline	MAB	Baseline	MAB	Baseline
light	light	**0.743** ± 0.004	1.611 ± 0.005	**0.675** ± 0.006	1.579 ± 0.007	**0.869** ± 0.005	1.579 ± 0.007
light	average	**0.712** ± 0.003	1.904 ± 0.006	**0.610** ± 0.006	1.843 ± 0.009	**0.788** ± 0.004	1.843 ± 0.009
light	heavy	**0.440** ± 0.003	2.706 ± 0.011	**1.034** ± 0.008	3.019 ± 0.006	**1.055** ± 0.006	3.019 ± 0.006
average	average	**1.013** ± 0.005	2.386 ± 0.009	**0.771** ± 0.010	2.477 ± 0.008	**0.969** ± 0.006	2.477 ± 0.008
average	heavy	**0.637** ± 0.005	3.108 ± 0.013	**1.388** ± 0.012	3.583 ± 0.006	**1.260** ± 0.007	3.583 ± 0.006
heavy	heavy	**1.411** ± 0.014	3.164 ± 0.014	**1.770** ± 0.013	3.780 ± 0.003	**1.737** ± 0.008	3.780 ± 0.003

(MA) experiment, where the two controllers do not exchange information with each other. In this case, the competition between the two devices generates cross-interference, which makes the algorithm work harder to find a suitable configuration. Note that in all experiments, the learning algorithm starts without prior knowledge, so any available action is equivalent. A simple heuristic may never converge, since the medium quality may change in any moment. RL, on the other hand, balances exploration (puts the system into a good response) and exploitation (looks for better values), and that is very hard to translate to a simple heuristic.

The confidence intervals for the MA experiment were also higher, in general, indicating a more pronounced variability in the convergence time. This is because the controllers are not coordinated, thus they do not exchange information with each other, getting feedback only through the readings of the environment. The placement of our equipment accentuated the interference between the devices, since the AP are located side-by-side, as well as the two stations. So in the MA experiment, the selection of an action, seeking to improve the QoE, can worsen the quality of the other station. We intend, in future work, to allow controllers to exchange minimum information (for example a local QoE indicator), then the experiment can be modeled as a cooperative multi-agent environment, as each controller can add to its local equation the reward variation, accounting for the effect on the other agents (controllers).

2) Regret: Our agent's performance can be compared with an optimal strategy that consistently reproduces the arm (selects the action) that is best in the first n steps, for any horizon of n steps, as shown in Equation 2. Although we do not know the optimal strategy, we know that the maximum reward is five, so we can compare the strategies using MAB, and the baseline, assuming that there is an optimal strategy that always gets the maximum reward. Table II shows the mean regret with a 95% confidence interval for the three experiments, showing the results using the control loop and the baseline. This result indicates that the control loop improves the QoE perceived by the user by at least 45 % in the worst case, and by 84 % in the best case. In all cases, the regret using MAB is lower than the baseline, and, in many cases, its confidence interval is tighter.

3) MOS distribution: Figures 4, 5, and 6 show the cumulative probability distribution for the predicted average MOS during the execution of the experiments, considering the site type combinations for each of the three test environments described in the Section IV-A. The MOS shown in this section is obtained using the predictor developed by Hora et al., using measurements that are performed in the APs [6]. The curves show in the X-axis the value of the MOS, which varies between 1 and 5. The Y-axis shows the cumulative probability. This way if one traces a line parallel to the X axis at a given Y value, she can identify that the process with the best MOS is the one that intersects this line to the rightmost position. Due to the available space, we did not show all the combinations, but only the worst and the best curves. Each graph contains the average MOS achieved by the stations accessing the web sites, i.e., each point in the curve represents the value resulting from each station's obtained MOS value, at each iteration, divided by the total number of stations. The continuous curve is the baseline, while the other curve (dotted) is the result with control loop using MAB. We can observe that in most of the graphs, the control loop curve presents better results than the baseline, i.e., the control loop's probability of obtaining better results, and thus increase user satisfaction, is greater than the baseline. In the *light-light* curves, the curves are closer. This is because the *light* site demands few network resources, so the baseline can already achieve good results. Note, however, that the regret for these cases is roughly the double for the baseline than for the control loop. However, in cases where one of the sites is a *heavy* site, the performance of the control loop is significantly better, and in the case of the SA experiment, the values are up to six times better than the baseline.

4) Page Load Time: The page load time consists of the time that takes to download the whole page, i.e., the html file, all script files, RSS files, and images files, and is measured at the stations for each download. Table III shows the average page load time, in seconds, of both stations, without using

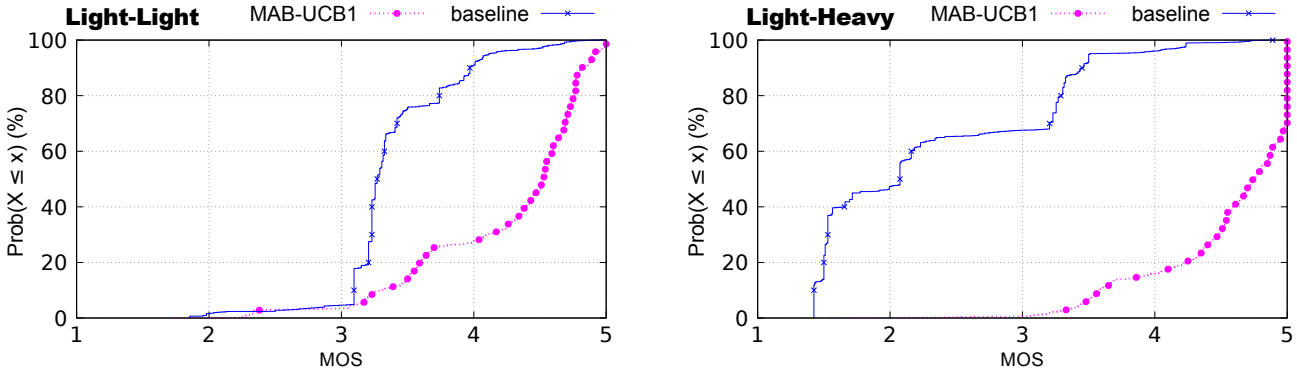

Figure 4: SA experiment - Cumulative distribution of MOS using MAB

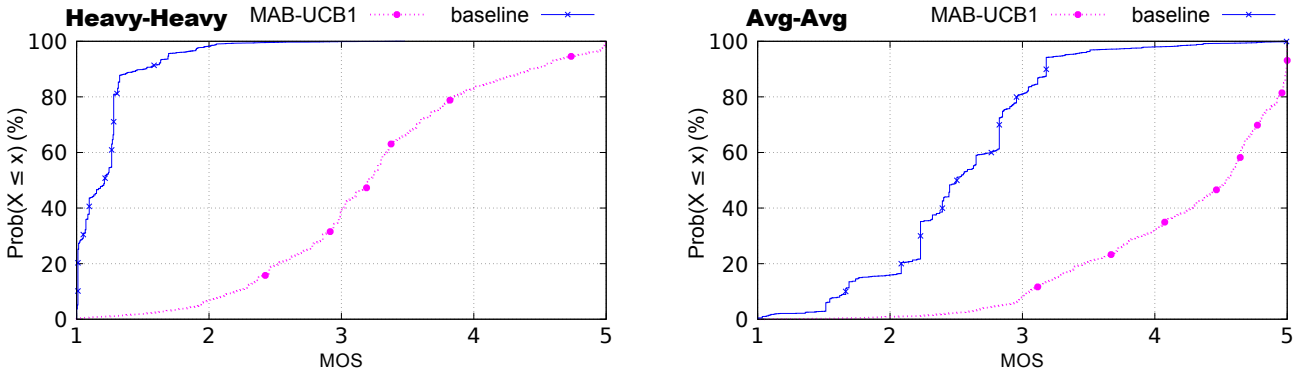

Figure 5: MA experiment - Cumulative distribution of MOS using MAB

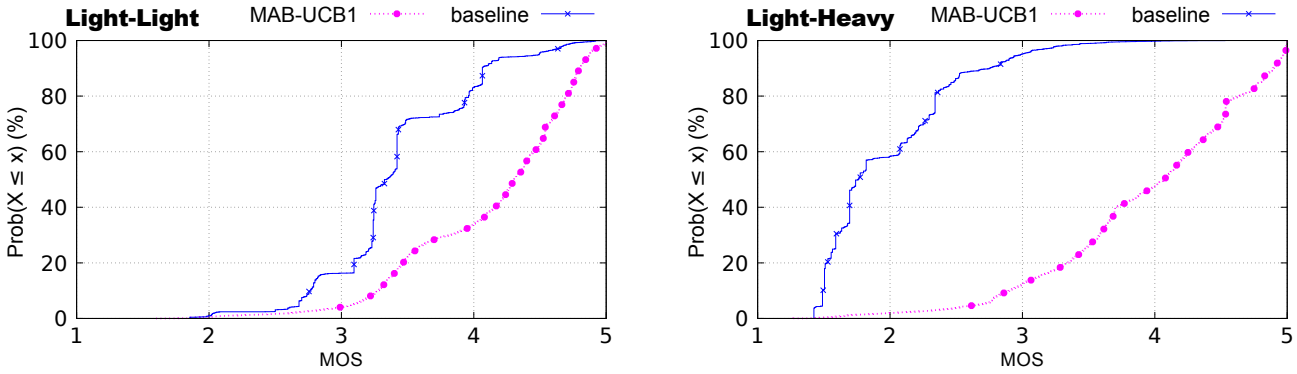

Figure 6: CA experiment - Cumulative distribution of MOS using MAB

the control algorithm and when using it. Each row represents one of the six site type combinations, and the columns are grouped by the experiment. The gain, i.e. how much better is the performance with the controller that without it, is also shown. All values are the average of the experiment, and with a confidence interval of 95%. The perceived variation at each request is because the accessed web servers return a slightly different set of images and scripts to each request, e.g. different publicity for each request, and due to small variations in the test environment.

The use of learning algorithms results in shorter page loading times. For example, in the *light–average* case for the SA experiment, the improvement is mainly achieved by reducing the page load time of the *average* type page. However, we did not observe a pattern among the experiments. That is, no combination has consistently shown an improvement pattern over another combination.

V. Related Work

Wireless channel conditions can be modeled using a finite-state Markov model [26], thus RL approaches can be applied to many wireless problems. Some previous research used RL for routing, channel and power control in ad-hoc and wireless sensor networks, like in [27, 28]. These were successful approaches, but they are not user-centric approaches, only optimizing *Quality of Service* (QoS) parameters.

Table III: Comparing page load time (in seconds)

Client 1	Client 2	SA Experiment			MA Experiment			CA Experiment		
		Without Controller (s)	With Controller (s)	Gain (%)	Without Controller (s)	With Controller (s)	Gain (%)	Without Controller (s)	With Controller (s)	Gain (%)
light	light	5.8 ± 0.6	2.8 ± 0.1	**108 %**	5.7 ± 0.0	3.3 ± 0.9	73 %	5.7 ± 0.0	4.5 ± 0.7	**25 %**
light	average	13.0 ± 0.1	3.8 ± 0.4	**240 %**	13.6 ± 0.2	9.6 ± 1.6	41 %	13.6 ± 0.2	6.7 ± 5.4	**101 %**
light	heavy	9.6 ± 0.1	4.4 ± 0.4	**118 %**	9.7 ± 0.1	5.4 ± 0.2	79 %	9.7 ± 0.1	4.0 ± 7.0	**140 %**
average	average	20.6 ± 0.2	7.2 ± 0.5	**186 %**	20.9 ± 0.1	9.0 ± 2.6	**131 %**	20.9 ± 0.1	18.3 ± 4.8	14 %
average	heavy	16.5 ± 0.1	8.5 ± 0.8	94 %	17.1 ± 0.1	8.5 ± 5.3	**101 %**	17.1 ± 0.1	8.7 ± 5.5	95 %
heavy	heavy	12.6 ± 0.2	7.3 ± 2.2	73 %	13.1 ± 0.1	8.0 ± 5.0	63 %	13.1 ± 0.1	7.2 ± 10.4	81 %

Recently, the research focuses in satisfying the users' need (an QoE approach). Baraković and Skorin-Kapov [3] surveyed the state-of-the-art of QoE Management, focusing on wireless networks and addressing three management aspects: (a) modeling, (b) monitoring and measurement, and (c) adaptation and optimization. They explored the use of MOS in cellular networks, but did not provide a wireless network QoE metric. The task of obtaining a user satisfaction assessment during the network operation is complicated, expensive, and not always well accepted by the user, thus control solutions usually employ estimates.

Some methods estimate QoE with metrics obtained at stations. For example, Habachi et al. [29] created a MOS-aware TCP congestion control using an online learning algorithm that outperforms other congestion control schemes in terms of QoE. They used packet loss rate and jitter, and their proposal selects the TCP congestion window size. Aguiar et al. [30] studied many QoE metrics in video transmissions, but they did not propose a control mechanism. Shaikh et al. [17] used linear, logarithmic, exponential and power regression to correlate page download time, throughput perceived by the user, and packet loss to QoE, but they did not apply the metric to control the network. A QoE/QoS correlation is provided by Kim et al. [31] for IPTV QoE evaluation, using bandwidth, burst level, delay and jitter. However, they evaluated this correlation only in cabled broadband connections, which are much more stable than a wireless link, and also did not use it to improve the QoE. Ghahfarokhi and Movahhedinia [8] and Wu et al. [9] proposed QoE-aware handover approaches, but, in both proposals, control runs at the mobile user station.

Other approaches consider metrics measured at the source of the data stream, such as on the video server in Gadaleta et al. [12]. They proposed D-DASH, which improves the QoE using Deep Q-Learning, but it only controls the video server, not the network as in our approach. Zhang et al. [32] propose a cache management scheme for HTTP servers using adaptive bit rate streaming in wireless networks to maximize the users' QoE. Their work uses a logarithmic function of the playback rate to infer the QoE metric, which was validated by 22 users, but they did not perform active learning or use an SDN approach.

The network nodes can be used as decision points. For example, in Yin et al. [14], the wireless medium access is decided locally by each node in the network, using a learning algorithm. However, without global view, it is harder to optimize the whole network. He et al. [18] used a metric to decide if data must be cached or dropped at each node of the network, and Chenji et al. [33] used QoE to allocate bandwidth in wireless networks, but the decision was made by solving linear optimization problems.

Other work uses SDN to improve the network performance, e.g., Amani et al. [34] proposed an offloading mechanism for 5G networks using SDN and wireless networks, using QoS metrics to support the decision. They used a network model to tackle the problem, and did not use a learning approach.

VI. Conclusion and Future Work

Automatic control mechanisms, which use the user satisfaction as feedback, allow network administrators to improve satisfaction while maintaining market competitiveness. Thus one of the main challenges today is to develop intelligent and resilient methods that can ensure the provision of good quality network services and meet the users' QoE expectations. Therefore, a control loop using learning algorithms should increase the adaptability of network services, while it maintains or increases the perceived QoE.

This paper proposes a closed control loop for WLAN using SDN, RL, and QoE, to improve the user's satisfaction. A prototype that optimizes web QoE was developed to test the proposal. Results show that the learning algorithm can reduce the average regret by at least 45% in the worst case, and 84% in the best case. The MOS can be improved by at least 4% in the worst case, and 167% in the best case, thus benefiting the user. The page load time was also reduced by 25 % in the worst case, and 233 % in the best case, compared to the baseline.

As future work, we will expand the number of devices in the experiments to better evaluate the scalability of our solution. Learning from previous experience, and generalization should also be tested, thus the algorithm starts using values learned in previous runs. Finally, to better cope with the user experience, we should consider the time to render the visible page to predict QoE, and for that we need to refine Hora et al.'s methodology.

ACKNOWLEDGMENT

The authors would like to acknowledge CAPES, CNPq and FAPEMIG, funding agencies from the Brazilian federal and state government, for financing this research.

References

[1] S. Biswas, J. Bicket, E. Wong, R. Musaloiu-e, A. Bhartia, and D. Aguayo, "Large-scale measurements of Wireless

Network behavior," in *ACM SIGCOMM Computer Communication Review*, vol. 45. ACM, 2015, pp. 153–165.
[2] L. DiCioccio, R. Teixeira, and C. Rosenberg, "Characterizing home networks with HOMENET profiler," *UPMC Sorbonne Universits, Tech. Rep. CP-PRL-2011-09-0001*, 2011.
[3] S. Baraković and L. Skorin-Kapov, "Survey and challenges of QoE management issues in wireless networks," *Journal of Computer Networks and Communications*, vol. 2013, 2013.
[4] N. Marchetti, N. R. Prasad, J. Johansson, and T. Cai, "Self-organizing networks: State-of-the-art, challenges and perspectives," in *2010 8th International Conference on Communications*. IEEE, 2010, pp. 503–508.
[5] R. Barco, P. Lazaro, and P. Munoz, "A unified framework for self-healing in wireless networks," *IEEE Communications Magazine*, vol. 50, no. 12, pp. 134–142, December 2012.
[6] D. N. d. Hora, R. Teixeira, K. van Doorselaer, and K. van Oost, "Predicting the Effect of Home Wi-Fi Quality on Web QoE," in *Proceedings of the 2016 Workshop on QoE-based Analysis and Management of Data Communication Networks*, ser. Internet-QoE '16. New York, NY, USA: ACM, 2016, pp. 13–18.
[7] H. A. Tran, A. Mellouk, S. Hoceini, and B. Augustin, "Global state-dependent QoE based routing," in *2012 IEEE International Conference on Communications*. IEEE, 2012, pp. 131–135.
[8] B. S. Ghahfarokhi and N. Movahhedinia, "A personalized QoE-aware handover decision based on distributed reinforcement learning," *Wireless networks*, vol. 19, no. 8, pp. 1807–1828, 2013.
[9] Y. Wu, F. Hu, S. Kumar, Y. Zhu, A. Talari, N. Rahnavard, and J. D. Matyjas, "A learning-based QoE-driven spectrum handoff scheme for multimedia transmissions over cognitive radio networks," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 11, pp. 2134–2148, 2014.
[10] N. Coutinho, R. Matos, C. Marques, A. Reis, S. Sargento, J. Chakareski, and A. Kassler, "Dynamic Dual-reinforcement-learning Routing Strategies for Quality of Experience-aware Wireless Mesh Networking," *Comput. Netw.*, vol. 88, no. C, pp. 269–285, Sep. 2015. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2015.06.016
[11] M. Jaber, M. A. Imran, R. Tafazolli, and A. Tukmanov, "A multiple attribute user-centric backhaul provisioning scheme using distributed SON," in *2016 IEEE Global Communications Conference*. IEEE, 2016, pp. 1–6.
[12] M. Gadaleta, F. Chiariotti, M. Rossi, and A. Zanella, "D-DASH: A Deep Q-Learning Framework for DASH Video Streaming," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 703–718, 2017.
[13] N. Aharony, T. Zehavi, and Y. Engel, "Learning wireless network association control with gaussian process temporal difference methods," in *Proceedings of OPNETWORK*, 2005.
[14] J. Yin, Y. Mao, S. Leng, X. Wang, and H. Fu, "Qoe provisioning by random access in next-generation wireless networks," in *2015 IEEE Global Communications Conference*. IEEE, 2015, pp. 1–7.
[15] R. Matos, N. Coutinho, C. Marques, S. Sargento, J. Chakareski, and A. Kassler, "Quality of experience-based routing in multi-service Wireless Mesh Networks," in *2012 IEEE International Conference on Communications*. IEEE, 2012, pp. 7060–7065.
[16] H. Moura, G. V. C. Bessa, M. A. M. Vieira, and D. F. Macedo, "Ethanol: Software Defined Networking for 802.11 Wireless Networks," *IFIP/IEEE International Symposium on Integrated Network Management*, 2015.
[17] J. Shaikh, M. Fiedler, and D. Collange, "Quality of experience from user and network perspectives," *Annals of Telecommunications*, vol. 65, no. 1-2, pp. 47–57, 2010.
[18] X. He, K. Wang, H. Huang, T. Miyazaki, Y. Wang, and S. Guo, "Green resource allocation based on deep reinforcement learning in content-centric IoT," *IEEE Transactions on Emerging Topics in Computing*, 2018.
[19] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Advances in applied mathematics*, vol. 6, no. 1, pp. 4–22, 1985.
[20] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1.
[21] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.
[22] J. Gittins, K. Glazebrook, and R. Weber, *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011.
[23] R. Jain, D.-M. Chiu, and W. R. Hawe, *A quantitative measure of fairness and discrimination for resource allocation in shared computer system*. Eastern Research Laboratory, Digital Equipment Corporation Hudson, MA, 1984, vol. 38.
[24] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the Multiarmed Bandit Problem," *Machine learning*, vol. 47, no. 2-3, pp. 235–256, 2002.
[25] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. Cambridge University Press, 2006.
[26] Q. Zhang and S. A. Kassam, "Finite-state markov model for rayleigh fading channels," *IEEE Transactions on communications*, vol. 47, no. 11, pp. 1688–1692, 1999.
[27] A. Forster, "Machine Learning Techniques applied to Wireless Ad-Hoc Networks: Guide and Survey," in *3rd International Conference on Intelligent Sensors, Sensor Networks and Information, 2007.* IEEE, 2007, pp. 365–370.
[28] R. V. Kulkarni, A. Forster, and G. K. Venayagamoorthy, "Computational intelligence in Wireless Sensor Networks: A Survey," *IEEE communications surveys & tutorials*, vol. 13, no. 1, pp. 68–96, 2011.
[29] O. Habachi, Y. Hu, M. Van der Schaar, Y. Hayel, and F. Wu, "MOS-based congestion control for conversational services in Wireless Environments," *IEEE Journal*

on Selected Areas in Communications, vol. 30, no. 7, pp. 1225–1236, 2012.

[30] E. S. Aguiar, B. A. Pinheiro, J. F. S. Figueirêdo, E. Cerqueira, A. J. G. Abelém, and R. L. Gomes, "Trends and Challenges for Quality of Service and Quality of Experience for Wireless Mesh Networks," in *Wireless Mesh Networks*, N. Funabiki, Ed. Intech, January 2011, ch. 6, pp. 127–148.

[31] H. J. Kim, S. G. Choi, H. S. Kim, and S. G. Choi, "A study on a QoS/QoE correlation model for QoE evaluation on IPTV service," in *2010 The 12th International Conference on Advanced Communication Technology*, vol. 2. IEEE, 2010, pp. 1377–1382.

[32] W. Zhang, Y. Wen, Z. Chen, and A. Khisti, "QoE-driven cache management for HTTP adaptive bit rate streaming over wireless networks," *IEEE Transactions on Multimedia*, vol. 15, no. 6, pp. 1431–1445, 2013.

[33] H. Chenji, Z. J. Haas, and P. Xue, "Low complexity QoE-aware bandwidth allocation for wireless content delivery," in *2015 IEEE Military Communications Conference*, Oct 2015, pp. 419–425.

[34] M. Amani, T. Mahmoodi, M. Tatipamula, and H. Aghvami, "SDN-based Data Offloading for SDN-based Data Offloading for 5G Mobile Networks," *ZTE Communications*, vol. 12, p. 34, 2014.