Automatic MAC Protocol Selection In Wireless Networks Based on Reinforcement Learning

André Gomes, Daniel F. Macedo and Luiz F. M. Vieira[1]

ARTICLE INFO

Keywords:
MAC sub-layer, MAC protocol selection, switching MAC protocols, reinforcement learning

ABSTRACT

Existing MAC protocols do not address the dynamism and complexity of the environment and applications of today's wireless networks. The running applications and the environment change all the time, and as a consequence the requirements of the wireless transmissions change. For example, in one moment a client is transmitting video, requiring high throughput; next it will control a robotic arm, requiring bounded delays. In this example, a contention-based MAC protocol would cope with flexible traffic demands, however it does not meet the delay constraints. Reservation based protocols, meanwhile, provide performance guarantees, but at a higher overhead. Hence, wireless networks require adaptive techniques that change how the network reacts over time. To that end, we propose SOMAC (Self-Organizing MAC), a system that uses reinforcement learning techniques to switch the MAC protocol in structured wireless networks according to the ongoing network demand. The novelty of SOMAC lies in its use of reinforcement learning, which solves the following shortcomings in the literature: *i*) the lack of models that cope with changes in its environment or lack of representative data during training; *ii*) the capacity to self-optimise based on a number of metrics. To showcase its genericity, we evaluated the model using two different optimisation metrics (throughput and delay) on a testbed.

Results indicate that our solution performs similar to an oracle choosing the most suitable MAC protocol from the list of implemented protocols up to 90% of the time. Further, SOMAC outperforms the state of the art by up to 20% in terms of protocol selection.

1. Introduction

Wireless networks are dynamic systems that are sensitive to problems such as interference, hidden nodes and link quality fluctuations. The propagation medium is highly variable in time, frequency and space [1]. These problems become even worse in networks operating on unlicensed frequencies because of the high competition for the spectrum. Furthermore, there is a diversity of applications running on wireless networks, each with different requirements. For example, real-time voice applications rely on lower latency connections and do not require high throughput. On the other hand, file transfer applications do not require low latency, but rely on high throughput [2, 3].

Despite the dynamism and complexity of wireless networks, MAC protocols usually focus on specific network scenarios. Let's consider two popular classes of MAC protocols as an example. CSMA-based protocols are appropriate when there are both low medium competition and low interference [4], while TDMA-based protocols have better performance in scenarios of higher medium usage since they are less sensitive to interference [4]. Medium competition, interference as well as other link characteristics usually vary both over time and in space. As a consequence, neither CSMA-based nor TDMA-based protocols are the most suitable choice at all times. The same argument holds true for other MAC protocols in the literature and their specific application scenarios. Examples such as using directional antennas but being depreciated on omnidirectional networks, or coping with lossy links with an added overhead that penalises reliable links, among others [5, 6, 7, 8, 9].

An alternative to address the dynamism of wireless networks is an automatic and adaptive switch of the MAC protocol according to the network conditions. This solution is feasible nowadays with Software Defined Radios (SDR) [10, 11]. According to the *Wireless Innovation Forum*, at least 90% of the commodity network devices already use the SDR concept to some extent [12]. Further, there are commercial solutions with a full software implementation of complex communication standards (e.g. the SRS 4G eNodeB) [13], together with efforts to enhance SDR performance on low cost hardware [14], suggesting the feasibility of our solution in real-world applications.

In this context, we have previously developed FS-MAC, a flexible platform for changing the active MAC protocol in the network [15]. Nevertheless, the FS-MAC platform has a limited MAC protocol selection algorithm. It uses fuzzy logic, which relies on the expertise of an external agent in order to fine tune the inference engine. Like most expert systems in the literature, FS-MAC requires changes when adding new protocols. Further, since the model is static (i.e. it does not learn over time), there is no guarantee that it will work when new services or networks are encountered.

The literature of adaptive wireless protocols, to the best of our knowledge, lacks extensible MAC adaptation algorithms that improve their operation over time. This is essential in wireless networks, since the medium conditions may change in ways that may not have been imagined during the design of the algorithm. In this context, our work goes a step

ORCID(s): 0000-0003-3612-6847 (André Gomes); 0000-0001-6668-4175 (Daniel F. Macedo); 0000-0002-9050-3001 (Luiz F. M. Vieira)

[1]The authors are with the Computer Science Department, Universidade Federal de Minas Gerais, Brazil. (e-mails: {andre.gomes, damacedo, lfvieira}@dcc.ufmg.br)

further from FS-MAC, proposing a decision algorithm that copes with unknown medium dynamics as well as unspecified MAC protocols. SOMAC is a Self-Organizing MAC sublayer that switches MAC protocols in structured networks (those with a fixed infrastructure, such as gateways or access points) based on Reinforcement Learning (RL) techniques. SOMAC adapts to network variations, resulting in a more effective selection engine. Further, its decision model is extensible, accepting new MAC protocols without changes to the RL engine. As a consequence, there is no need for trained specialists to deploy or train SOMAC. Finally, the model optimizes any performance metric chosen by the network administrator, unlike related works later described in this paper.

The main contributions of this study are: (*i*) the development of a RL mechanism that selects in real-time the best MAC protocol according to the network conditions; (*ii*) the evaluation and testing of the proposal in real wireless networks using SDR, and (*iii*) the release of the project as open source code[1]. Results indicate that SOMAC performs optimally up to 90% of the time. Further, it outperforms its competitors by up to 20% while selecting the best MAC protocol.

The rest of this paper is structured as follows. Section 3 lists the most related works from the literature. Section 4 briefly presents an overview of RL and the Q-Learning algorithm. Our proposal is described in Section 5 while Section 6 discusses the experimental results. Finally, Section 7 concludes the main remarks of our work.

2. Basic Concepts

2.1. Change MAC protocol or change the MAC parameters?

Changing from one MAC protocol to another is a harsh decision. However, each MAC protocol has certain features that are inherent to its construction and which cannot be changed easily. We will present our argument by means of an example.

We evaluate SOMAC changing from a TDMA-based protocol to a CSMA-based protocol. However, one may argue that the same objective could be achieved using a hybrid protocol, such as IEEE 802.11 [16]. In this protocol, the airtime can be divided into contention-free periods using PCF and contention-based access periods using DCF. Hence, the adaptation algorithm would adjust the proportion of time in PCF and DCF in order to achieve the expected goals.

Unfortunately, this approach is independent of only one of the infinite choices for medium access control, which is the choice of the contention method. What if the designer wants to change another parameter of the protocol, and that parameter is fixed in the protocol? IEEE 802.11 does not support FEC, so the designer should switch to another protocol. Likewise, a new protocol would be needed if the radio should switch from half duplex to full duplex operation.

[1] https://github.com/avgsg/gr-somac/

2.2. Optimality of MAC Protocols

As mentioned in [17], the main characteristics of TDMA and CSMA come from how they time their transmissions. TDMA is a timer-based system, in which nodes transmit at certain pre-defined times. Meanwhile, CSMA is an event-triggered system, where nodes attempt to transmit once they have new data. Because of that characteristic, CSMA is more suited for elastic traffic, while TDMA provides more effective transmissions for sensitive applications.

CSMA is shown to perform well, having a small access time to the medium as well as a high throughput when a small percentage of the stations attempt to access the medium [18]. However, when more stations attempt to access the same time, the stochastic nature of the collision avoidance algorithm generates undesired characteristics. First, the maximum delay to access the medium is unbounded. Second, some of the stations may be subjected to blocking, as their transmission is deferred by others. This behavior is inadequate for networks where periodic transmissions are required, as in vehicular networks [19]. Although those drawbacks can be reduced with admission control, reservation and prioritization schemes, most situations requiring reliability and predictability tend to employ TDMA.

TDMA is a good fit for time sensitive networks as well as flows with constant traffic demands. This is due to its high predictability and orderly operation. Further, TDMA copes well with dense networks, since the scheduled access reduces the amount of collisions when compared to CSMA. On the other hand, TDMA suffers from synchronization problems. All stations must follow the same transmission schedule, so TDMA requires a certain overhead to keep the clocks in tune. Furthermore, the duration of the transmission slots depend on this synchronization. Longer slots require less overhead, however they delay the start of a packet transmission [17].

3. Related work

The literature presents a number of works investigating adaptive MAC protocols. The focus of this review is on the adaptation techniques, and not on the MAC protocols themselves. Overall, those techniques are applicable to any MAC protocol with very few adaptations. Because of that, the surveyed works are not limited to a single scenario (e.g. sensor networks, vehicular networks), or to a single class of protocols (e.g. WLANs, WBANs). This section groups the relevant works into three categories, based on how they approach network adaptation.

3.1. Hybrid MAC protocols

Hybrid MAC protocols lump characteristics of two or more protocols together. The goal is to exploit the best of each MAC protocol. However, a side effect of this approach is also lumping together some undesirable characteristics, resulting in new limitations.

CTh-MAC is proposed by [20] and divides the network into different subsets according to the distance from the sink

node. A time slot is assigned to each subset in a TDMA fashion, while nodes within the same subset compete for medium access in a CSMA fashion. This approach is highly dependent on the network topology. As a consequence, subsets must be recomputed periodically in order to deal with mobility.

Both [21] and [22] have hybrid operational modes that mix characteristics of CSMA and TDMA. The medium access is ruled by time slots in a TDMA fashion, while nodes compete for each time slot similarly to CSMA. Although nodes are free to compete for any time slot, there are priority schemes during high demanding periods. In Z-MAC [21], the owner of the time slot always has the highest priority for transmitting data. LA-MAC [22], however, handles priority by changing the contention window size of competing nodes. High priority nodes are given smaller contention windows than low priority ones. The high initialization cost is a common disadvantage of both solutions.

3.2. Adaptive MAC protocols

Adaptive MAC protocols adjust parameters as a response to network changes. They may also refer to generic MAC protocols capable of adapting themselves to policies of a specific wireless network. However, they also present some limitations as we describe below.

The work of [23] consists of an adaptive CSMA protocol. The solution uses Markov chains controlled by network traffic patterns to set the contention window size of CSMA. The smaller the contention window, the more aggressive the node is to access the medium. As a disadvantage, this approach is still prone to classic CSMA problems such as hidden and exposed nodes.

In [24], the authors propose DLMA (Deep-reinforcement Learning Multiple Access). DLMA employs RL to learn how to use idle time slots while coexisting with slot-based protocols such as TDMA and slotted ALOHA. A drawback of this approach is that DLMA passively disputes medium access, resulting in poor performance while operating in high demanding networks. Indeed, the proposed evaluation lacks such scenarios, as idle time slots are guaranteed while DLMA coexists with TDMA for example.

3.3. Switching MAC protocols

This category differs from the previous two as it does not modify MAC protocols per se. Indeed, both hybrid and adaptive MAC protocols could be used, switching from one to another according to the network state.

TAISC is an independent hardware framework for MAC protocol development [25]. The framework allows MAC protocol upgrades after deployment as well as the implementation of multiple MAC protocols, which can be switched in response to network changes or applications needs. Although the authors present a mechanism capable of switching MAC protocols, the selection engine (i.e. how switching is triggered) is not exploited in [25].

HybridMAC is proposed by [26] and exploits characteristics of both CSMA and TDMA protocols. The solution switches between CSMA and TDMA modes and is focused on WSN (Wireless Sensor Networks). TDMA is the initial protocol. After receiving a RREQ (route request) message, a node switches to CSMA mode. Nodes along the route keep operating according to CSMA while others switch back to TDMA mode. The downside of this approach is that the selection engine relies explicitly on WSN control messages, making it difficult to generalize the solution to other kinds of networks.

AMAC switches between CSMA and TDMA [27]. The authors propose two different selection algorithms. The first relies on throughput degradation, where the MAC protocol is switched every time the performance drops more than 20%. This approach is naive when the throughput degrades for both MAC protocols. For example, an external source of interference affects the overall network performance of both CSMA and TDMA. In this case, switching the MAC protocol does not necessarily improve performance. The second selection algorithm predicts future network patterns. Specifically, AMAC predicts the average size of network packets. If the upcoming network packets have either medium or large size, AMAC selects TDMA; otherwise, CSMA is selected. This approach is also naive because network performance is not strictly bounded by the size of network packets. The evaluation results are drawn from real wireless networks using SDR.

In [28], the authors propose mathematical models to establish which protocol to use. There are two MAC protocols available, DCF and D-TDMA. A mathematical model is created for each one, and the so-called switching points are then derived. This approach lacks scalability for adding new protocols, since a mathematical model must be created for every new MAC protocol, requiring technical expertise from the network administrator. Furthermore, [28] accounts for a few characteristics of the propagation medium, considering, for example, an error-free wireless channel.

We previously proposed FS-MAC, a flexible MAC platform for wireless networks [15]. The MAC protocols available in FS-MAC are CSMA and TDMA. The main disadvantage of FS-MAC is the selection algorithm using fuzzy logic. Similarly to [28], FS-MAC does not scale well, since it requires technical expertise to derive new fuzzy rules when adding MAC protocols. Additionally, FS-MAC does not cope with link variations over time. In such environments, the network response to MAC protocols changes over time, and the fuzzy rules are likely to become obsolete, requiring new analysis and updates from the network administrator.

SMAC is a MAC protocol switching mechanism based on machine learning (ML) [29]. There are two MAC protocols available, CSMA and TDMA, which are selected by a Sequential Minimal Optimization (SMO) classifier. SMAC deploys supervised ML and, therefore, requires a labeled training set. This set was derived from network simulations, in which the authors compared the performance of each protocol under the same network circumstances. This labeling process is hardly feasible in real-world wireless networks, since it is at mercy of external agents that limit the repeata-

	[27]	[28]	[15]	[29]	SOMAC
Experimental eval.	✓	-	✓	-	✓
No. protocols	2	2	2	2	2
Prot. extensibility	✓	-	-	-	✓
ML based	-	-	-	✓	✓
Non-stationary env.	✓	-	-	-	✓

Table 1
Comparison between SOMAC and other related papers

bility of the experiments. In addition, SMAC is limited to the knowledge acquired from the training set, which is an issue because of the non-stationary nature of wireless networks.

3.4. Comparison with the state of the art

Table 1 summarizes the main characteristics of each related work as well as SOMAC. Neither TAISC nor Hybrid-MAC is listed there, as the former does not have a selection engine and the latter exclusively focuses on WSN. Only AMAC [27], FS-MAC [15] and SOMAC are experimentally evaluated on real hardware. Although all of them are tested against two MAC protocols, only [27] and SOMAC easily scale to a larger number of protocols, not requiring third-party technical expertise. SMAC [29] and SOMAC employ ML based selection algorithms, however the former does not cope with the dynamicity of wireless networks (the "non-stationary environment" row on the table).

The novelty of SOMAC with regards to the state of the art lies in the use of reinforcement learning, which copes with a significant flaw in traditional supervised learning methods: the difficulty of ensuring that most of the possible scenarios are representing in the training set. This is specially important in wireless communications, where the amount of different combinations is almost endless (e.g. distance from sender to transmitter, versions of hardware and software, interference patterns, etc), so it is very hard to build extensive training sets. Since in reinforcement learning the algorithm is still learning during the deployment, every new situation that the system encounters will improve the model.

Second, the algorithm optimises the MAC protocol selection using any numeric performance variable. In this work we optimise the choice of MAC protocol using the throughput, and then show the flexibility of our solution by optimising the MAC protocol based on frame delay.

4. Overview of Reinforcement Learning

Reinforcement Learning (or simply RL) is designed to learn through interactions in a trial-and-error fashion [30]. The learning process is guided by rewards, as illustrated in Fig. 1. An agent periodically takes actions (a_t) over an unknown environment. Based on that action, the agent is led to a state s_{t+1}, which might be either different or equal to the previous one, s_t. In addition, the agent receives a reward r_{t+1} that dictates the quality of the chosen action a_t. The goal is to take actions that maximize the reward over time.

An RL process is formally defined as a Markov Decision Process (MDP), where $S = s_1, s_2, ..., s_n$ is a finite set

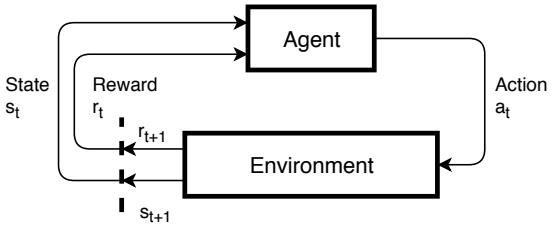


Figure 1: RL overview from [30]

of states, $A = a_1, a_2, ..., a_m$ is a finite set of actions, and $R_t(s_t, a_t, s_{t+1})$ is the expected reward function from state s_t to state s_{t+1} through action a_t [30]. The mapping between states and actions is called policy (π). The function $\pi(a|s)$ represents the probability of taking action a given a state s [30]. An optimal policy π^* leads the RL agent to achieve maximum cumulative reward through a set of optimal actions.

Additionally, RL methods are mostly divided into two types. The first one is called model-based and deploys models of the environment for planning [30]. Such methods usually infer the solution of a decision-making problem from models before actually executing the decision (e.g. Dyna-Q [31]). The second method is called model-free, and primarily relies on learning rather than planning [30]. In this case, there is no model to infer the reward to an action. Therefore, the reward is only known after executing a certain action (e.g. Q-Learning [32], Sarsa [30]).

4.1. Q-Learning

Q-Learning is a model-free RL algorithm [32]. The algorithm employs Eq. 1, where the function $Q(s_t, a_t)$ represents the expected discounted reward of state s_t after taking action a_t [33]. Parameters α and γ are the learning rate and the discount factor, respectively, and determine to what extent the algorithm learns from a new experience. More specifically, the discount factor dictates how relevant is the maximum expected Q-value from state s_{t+1} ($\max_a Q(s_{t+1}, a)$), and r_{t+1} is the reward received after taking action a_t from state s_t.

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a)] \quad (1)$$

The convergence of Q-Learning requires that all state-action pairs continue to be updated [30]. This is even more important in non-stationary environments, where the environment's response changes over time. As a consequence,

the Q-function must be updated on a regular basis, otherwise it may lead the agent to an undesirable policy. On the other hand, continuous testing of new state-action pairs may also lead to bad actions when the Q-function is still up to date. The dilemma between trusting the already known Q-function and testing new state-action pairs (to ensure convergence of Q-Learning) is called *exploration and exploitation dilemma*. There are several different approaches to tackle it, and the three most popular ones are summarized below.

ϵ-greedy: In ϵ-greedy, the agent randomly selects an action with probability ϵ, where $0 \leq \epsilon \leq 1$ [34]. Otherwise, it follows a greedy policy based on Q-values (see Eq. 2). Larger values of ϵ lead to more exploration.

$$a_t \leftarrow \begin{cases} \underset{a}{arg\ rand}\{Q(s_t, a)\} & \text{with prob. } \epsilon \\ \underset{a}{arg\ max}\{Q(s_t, a)\} & \text{otherwise} \end{cases} \tag{2}$$

Upper-Confidence-Bound (UCB): This approach ponders the uncertainty of less visited states [34]. Although the agent selects actions greedily, a bonus factor is added to each state-action pair alongside the Q-function as per Eq. 3. Whilst the function choice may vary, an example is as follows [30]: $c\sqrt{ln(t)/N_t(s,a)}$, where c is a hyparameter constant and $N_t(s, a)$ is a counter of how many times action a has being taken from state s at time t. The more visited the state is, the smaller is its bonus. Over time, the bonus factor may influence the greedy selection and lead to the exploration of the least visited states.

$$a_t \leftarrow \underset{a}{arg\ max}\{Q(s_t, a) + bonus(s_t, a)\} \tag{3}$$

Softmax: Another approach is the selection of actions according to action probabilities. Softmax relates the Q-values to a Boltzmann distribution according to the policy described in Eq. 4 [34]. In other words, the action a_j has a probability of $\pi(a_j|s_t)$ of being chosen, given the current state s_t. The parameter T is called temperature. If $T \to 0$, the agent does not explore at all while $T \to \infty$ leads to intensive exploration (i.e. random actions).

$$\pi(a|s_t) \leftarrow \frac{e^{Q(s_t,a)/T}}{\sum_{i=1}^{m} e^{Q(s_t,a_i)/T}} \tag{4}$$

The Q-Learning algorithm is described in algorithm 1. The hyperparameters are α, γ and the parameters of the chosen exploration strategy (e.g. ϵ if Q-policy is ϵ-greedy). An action is selected based on the Q-policy. The chosen action leads to a state s_{t+1} and a reward r_{t+1}, which are used for updating the Q-function. If necessary, the algorithm also updates the parameters of the respective exploration strategy.

Algorithm 1 Q-Learning algorithm

Input: hyperparameters

Initialize $Q(s, a) = 0\ \forall s \in S, a \in A(s)$
loop
  Choose a_t from s_t using Q-policy
  Take action a_t, go to state s_{t+1}, compute reward r_{t+1}
  $Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t)+\alpha[r_{t+1}+\gamma \underset{a}{max} Q(s_{t+1}, a)]$
  if Q-policy is UCB **then**
    Update $bonus(s, a)\ \forall s \in S, a \in A(s)$
  else if Q-policy is softmax **then**
    $\pi(a|s_t) \leftarrow \frac{e^{Q(s_t,a)/T}}{\sum_{i=1}^{m} e^{Q(s_t,a_i)/T}}\ \forall a \in A(s_t)$
  end if
  $s_t \leftarrow s_{t+1}$
end loop

5. Changing the MAC layer dynamically using reinforcement learning

SOMAC (Self-Organizing MAC) is a MAC sublayer that automatically selects MAC protocols according to changes on the network's characteristics. SOMAC optimizes a certain performance metric and switches MAC protocols when such a decision will improve the network performance. Any performance metric can be chosen as a target, such as delay, throughput, packet drop rate, or even a combination of those, as long as the wireless nodes provide a numeric feedback to the SOMAC decision engine. Further, since SOMAC employs RL, it should adapt to changes in the network, such as new applications, node churn or variations on link quality.

The architecture of SOMAC is divided into two parts. First, the network architecture. Second, the software modules that form SOMAC. We will describe in more details the MAC selection algorithm based on RL, since it is the main contribution of this work.

5.1. Network architecture

The network architecture is based on a master-slave hierarchy. An example is illustrated in Fig. 2. The master node dictates which MAC protocol must be used on the network by periodically broadcasting control messages to other nodes as illustrated by Fig. 2, where a symbolic MAC envelope is sent by the master to other nodes. The granularity of those messages is 5 seconds and was empirically established. There is only one master node per network.

At the moment, we consider that SOMAC will be employed on networks using a fixed gateway to the Internet (e.g. WiFi networks in infrastructure mode). In such situations, the gateway is the logical choice for the master node. Due to the existence of a fixed master node, SOMAC should not be employed as is ad hoc networks. In such situations, the network should self-organize into clusters, and elect a cluster-head to act as the master for a period of time.

Other nodes are called slaves. They operate according to the MAC protocol informed by the master node, and also send control messages on a regular basis. Those messages report the performance metrics to the master. Again, the granularity was empirically established to 30 seconds, based on previous work from the group [15]. This value is long enough to suppress sudden network fluctuations and small enough for effective network monitoring.

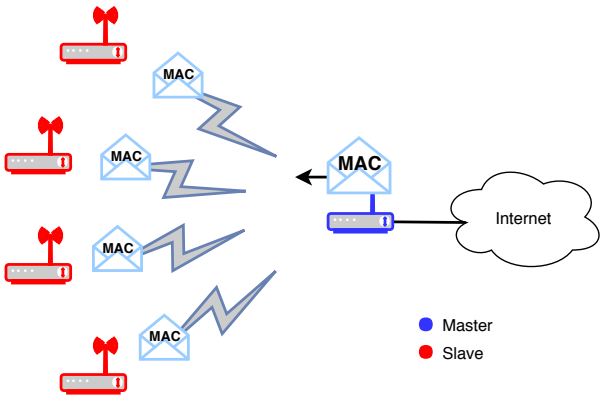


Figure 2: Example of network architecture

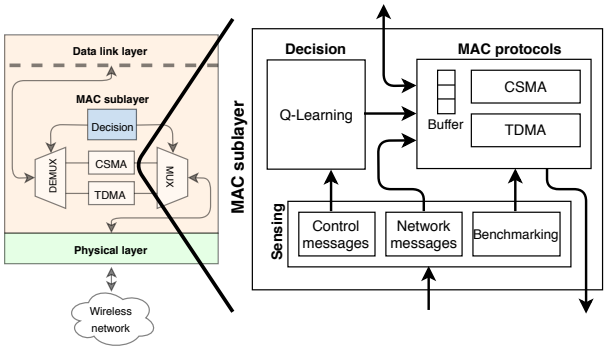


Figure 3: Software architecture

Lest messages from the master node are lost, some of the nodes may operate under different MAC protocols. To reduce the number of nodes misaligned with the master's decision over time, SOMAC periodically broadcasts which protocol is in use. This approach is inspired by our previous work [15], where we showcase a network overhead of only 2%.

Network performance metrics are aggregated by the master node at time intervals of 60 seconds. This is the same time interval of the MAC protocol selection. The available aggregation operations are: *sum*, *minimum*, *maximum* and *average*. Any network performance metrics (or a combination of those) can be used for protocol selection, and different aggregations give flexibility to the selection engine.

5.2. Software architecture

The software design of SOMAC is based on FS-MAC [15]. Fig. 3 illustrates the software architecture. Except for the "Decision" module, which is present only on the master, master and slave have the same components. There are three main modules, which are briefly described below.

5.2.1. Decision Module

It runs on the master, and controls the MAC protocol used by the MAC sublayer. We use Q-Learning for this task, as described in further details in a later section. The decision is based on the input of the network metrics collected by the sensing module. After deciding, the master node informs the slaves of its decision. The decision module of the slaves controls the running MAC protocol according to the decision of the master.

5.2.2. MAC protocols

Two MAC protocols are currently available in SOMAC: CSMA/CA and TDMA. The running protocol connects the upper part of the Data Link layer to the Physical layer, and vice-versa. This module also houses a protocol independent packet buffer. This buffer contains frames from upper and lower layers. As a consequence, no frames are lost if the MAC protocol is switched. Both master and slave have those buffers, since the two types of nodes may generate application messages.

5.2.3. Sensing Module

It classifies and redirects network frames. Control messages are forwarded to the decision module, while data go to the running MAC protocol. In addition, the sensing module assesses the performance metrics of the network. The measurement is periodically sent to the master, to be used in the decision process.

SOMAC is implemented over SDRs using GNU Radio[2]. The decision module is implemented in Python, while both MAC protocols and sensing modules are implemented in C++ for better performance. The upper layers are from the Linux operating system of host computers and are connected to GNU Radio via TUN/TAP interfaces [35]. The Physical layer is also implemented in software, and is provided by [36], being IEEE 802.11a/g/p compatible.

5.3. MAC selection algorithm

The MAC protocol selection algorithm answers the following question:

> *Given the current MAC protocol P_x and a performance metric m, should we change to another MAC protocol P_y to optimize m, or keep using P_x?*

We employ Q-Learning for this task. The native form of Q-Learning is used in SOMAC, since it copes with non-stationary environments and is model-free [30]. The state space consists of the available MAC protocols. The actions are either *keep* or *switch to* another MAC protocol. The general definition is formally given by $S = P_1, P_2, ..., P_n$ and $A(s) = a_{s1}, a_{s2}, ..., a_{sn}$, where P_n stands for the n^{th} available MAC protocol and a_{si} is the action of either switching to protocol P_i given a state s or keeping the current MAC protocol if s is already P_i. Fig. 4 illustrates this MDP. In this Figure, the states represent each available protocol. Arrows indicate the possible actions. We do not represent the reward in the figure, however for each state transition there will be a reward, given by the function $R(s', a')$, where s' is the current state and a' is the selected action.

[2] A software development toolkit for signal processing and SDR programmability, https://www.gnuradio.org/

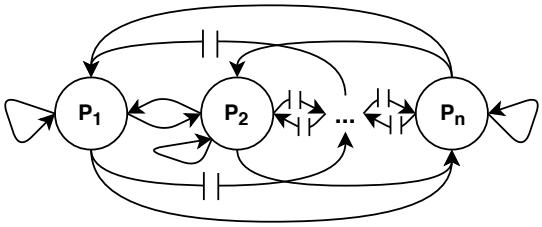


Figure 4: Generic representation of SOMAC's Markov Decision Process

The proposed model is independent of any specifics of the protocol, and because of that the selection algorithm works with any MAC protocol. Further, Section 5.5 will discuss how this model could configure multiple parameters at the same time.

State and action size. Based on the description above, if we have n protocols, then there will be n states in the model. Meanwhile, for each state we will have n possible actions (change to another protocol, or keep the current protocol). As a consequence, since this is a directional graph, we have n^2 actions in total in the state-action graph.

The reward function is described by Eq. 5. The function $g_t(m)$, restricted to $[-1, 1]$, stands for the percentage gain and refers to what extent a network performance metric m has changed. For example, if $m = latency$ and the latency drops from 100 ms to 80 ms from $t-1$ to t, $g_t(m)$ accounts for a gain of 20%, resulting in $g_t(m) = 0.2$. Conversely, $g_t(m) = -0.2$ if latency increases by 20%. Negative rewards are used as punishment, expressing that a bad action has been taken. A percentage gain scale gives flexibility for choosing any performance metric, independent of unit and magnitude. A scaling constant ξ is used to balance exploration and exploitation, so the reward is large enough for fast exploration but not so much to the point of excessive exploitation.

$$R_t = \begin{cases} 0, & \text{if } g_t(m) > 0 \textbf{ and } \text{no protocol switch} \\ g_t(m) \times \xi, & \text{otherwise} \end{cases} \tag{5}$$

Note that the reward function may use any scalar value to define which protocol to use. Hence, the model could use even more complex metrics such as a metric related to the quality of experience of the user. In order to optimise for a number of metrics, one could create a new metric which is the weighted sum of different performance parameters. For example, the protocols could optimise at the same time for throughput and latency using a metric $m = \delta\lambda + (1-\delta)D$, where λ is the normalized throughput, D is the normalized delay, and δ is a prioritisation constant.

5.4. Choice of Machine Learning Method

There are two main reasons for choosing Q-Learning instead of other ML techniques. First, wireless networks are non-stationary environments, whose performance relies on user behaviours (e.g. running applications) and variations on the wireless link (e.g. weather conditions and physical signal obstacles). As a consequence, the wireless links vary their characteristics in time, space and frequency [1]. In this context, supervised ML techniques are unfeasible because they rely on the acquired knowledge from a previously obtained training set. Although there are several updating mechanisms for adapting ML models over time, they usually focus on massive data streams and are prone to problems such as stability-plasticity and catastrophic forgetting [37, 38].

The second reason relates to the complexity of building accurate models for wireless networks. As previously stated, the propagation medium is highly variable, making it difficult to model. Additionally, intrinsic characteristics of wireless devices such as antenna gain, signal sensitivity and effective radiated power also affect the network performance. Even the MAC protocol itself impacts network performance and must be considered by the model [28], resulting in many parameters to be accounted while building a wireless network model. Conversely, [30] highlights that models have to be reasonably accurate to be useful, making it difficult to build a functional model for wireless networks. A model-free technique such as Q-Learning is thereby more appropriate for our context.

An alternative is to deploy variations of Q-Learning such as DQN (Deep Q-Network), where Deep Neural Networks derive the Q-function [39]. However, this approach is still prone to non-stationary environments and, consequently, is challenging to maintain the Deep Neural Network up to date.

Q-Learning, on the other hand, does not only deal with non-stationary environments but is also a model-free technique [30]. Also, our simple modelling approach copes with the results of [40, 41], which indicate that simple heuristics tend to outperform more advanced algorithms in real-world scenarios.

5.5. Controlling Multiple Parameters

The proposed solution could optimize multiple MAC parameters at the same time. For example, instead of changing the MAC entirely, it could configure certain parameters of the MAC. Further, it would be possible to control parameters from lower layers, such as the physical layer. This section will briefly comment on how to extend the model to cope with both options.

As shown in Fig. 4, the model employs one state for each possible configuration of the wireless nodes. The current configuration is the current state, and the model decides when to switch to another state in the Markov Decision Process, hence changing the network configuration (in our case the MAC protocol).

This model is easily extensible to cope with multiple parameters at the same time. Let's assume that the model should control n system parameters, from the choice of the MAC protocol to things such as the transmission power, the choice of the wireless channel or even internal parameters of a MAC protocol. For each parameter $p_i \in P$, we assume a finite number of configurations equal to $|p_i|$. Each possible configuration will be given by a tuple $(p_1, P_2, ..., p_{n-1}, p_n)$, thus the model would have $\prod_{i=1}^{n} |p_i|$ possible states. In each

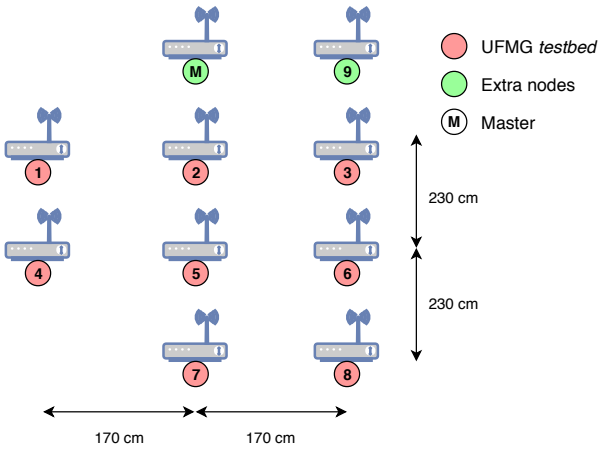


Figure 5: The layout used for experimental evaluation.

transition from one state to the other, the difference in elements in the tuple would indicate the required configuration change. In order to make the problem simpler, the state transitions should be allowed only when a single element of the tuple changes among states. Note that no changes are needed to the reward function.

6. Evaluation

We experimentally evaluated SOMAC in wireless networks using SDR. This section presents both the deployed methodology and the results of such evaluations.

6.1. Experimental Setup

Our results are obtained in an SDR testbed. We used the UFMG testbed from the FUTEBOL project [42], which is an open infrastructure, availeble to any researcher remotely using Fed4Fire. The testbed provides homogeneous hardware, that is, all nodes have the same configuration. Overall, the testbed consists of 8 USRP B200 and B210 SDRs, each connected to an Intel Core i7-6700T computer with 16GB of RAM memory running Ubuntu 16.04 LTS [43]. In addition, we also used two extra wireless nodes available at the research lab WINET[3]. As a consequence, the USRPs as well as the compute devices have an heterogeneous configuration. Those nodes consist of Intel Core i5-3317U computers with 8GB of RAM memory running Ubuntu 16.04 LTS and are connected to software radios USRP B210. Fig. 5 presents the resulting layout. Note that this layout is a grid due to limitations in the testbed. This is a public testbed, so the nodes should not be moved in order to avoid impacts on experiments from other users. This layout also has a characteristic that there are no hidden terminals, as every node is able to communicate with each other.

Regarding the setup of the PHY and MAC layers, the following configuration has been employed. The wireless physical layer works in an 802.11a/g/p fashion and is the same implementation from the work of [36]. We employ a fixed modulation, based on OFDM, so there is no rate adaptation algorithm. Further, channel bonding is not implemented, and the transmission power is fixed. All the experiments are performed using the same wireless channel (13, 2.472GHz), in order to have similar interference conditions for each protocol. It is worth noting that the testbed is not isolated from external interference, so the experiments suffer from uncontrolled interference from nearby Wi-Fi networks and other types of signals operating in the unlicensed spectrum.

In the MAC layer, we have kept the same implementations of TDMA and CSMA of [15]. The CSMA protocol is similar to IEEE 802.11, leaving out medium reservation (RTS/CTS messages) and PCF. All the other characteristics, such as the exponential back-off and the carrier sensing methodology is the same as in IEEE 802.11. For TDMA, we have adopted a superframe logic. The superframe starts with a allocation period, in which stations have pre-defined times to indicate whether they would like to transmit in that superframe. Next, the communication period starts, with the transmissions ordered by the requests in the allocation phase. The size of each transmission slot is fixed, so the superframe will be shorter when less stations are willing to transmit.

The application traffic is constant bit rate. We employed ping, sending 64-byte packets at a rate of 100 packets per second per station. Since our implementation creates a TUN/TAP network adapter, in future work we plan to evaluate the performance of SOMAC using more realistic traffic (e.g. video transmissions).

6.2. Methodology

Each experiment corresponds to a different configuration of transmitting nodes over time. For simplicity, we assume nodes communicating only with the master node, which is equivalent to infrastructured networks where nodes communicate exclusively to an access point (AP). Further, we repeat each experiment 10 times for every MAC protocol (i.e. CSMA/CA, TDMA and SOMAC), and present the results with a confidence interval of 90%, unless stated otherwise.

SOMAC is also evaluated against two related works, [15] and [27]. We do not evaluate [28] because of the mathematical expressions for establishing the switching points, exclusively related to MAC protocols that are not available in SOMAC. Moreover, [28] does not account for characteristics of the propagation medium, making it unlikely to work in real-world wireless networks. Neither is [29] evaluated against SOMAC. In this case, obtaining the training set is hardly feasible and too costly in real wireless networks because it requires systematic repetitions of network configurations for labeling data. On the other hand, [15] and [27] consider practical aspects of real-world wireless networks, making their implementation and evaluation in SDR feasible.

The initial MAC protocol is randomly selected for each experiment while operating with SOMAC or related works. The goal is not to benefit one protocol or another in specific network configurations, biasing our results. Furthermore, we initialize the Q-values in SOMAC to zero. The initial Q-

[3]The research lab that owns the UFMG testbed, http://www.winet.dcc.ufmg.br

values may be used to express previous knowledge about the system. However, solutions that focus on initial values tend to perform poorly in non-stationary environments as stated by [30]. In our approach, it is also hard to establish which protocol performs better without initializing the network in first place. Therefore, we opt for a zero initialization while evaluating our proposal.

The evaluation is based on three criteria. The first one is called *optimality*, and represents the percentage of time that SOMAC operates using the best MAC protocol. We compute this metric by looking up the log files of our experiments. Alg. 2 describes this computation, where line 3 determines the optimal protocol (i.e., either *max*, e.g., throughput, or *min*, e.g., latency, average performance) at time t, and line 4 checks if SOMAC performs in accordance to that protocol. In some cases, both CSMA and TDMA perform similarly. So, we inspect if SOMAC's performance overlaps with the confidence interval of either protocols (if available; otherwise, compare absolute values), accounting periods when all protocols perform likewise. Hence, *optimality* is said 90%, for example, if the performance of SOMAC is in accordance with the performance of the best protocol 90% of the time.

Algorithm 2 Optimality

Input: SOMAC, CSMA, TDMA

1: $count \leftarrow 0$
2: **for** t in T **do**
3: OPT $\leftarrow arg\ opt\{avg(\text{CSMA}_t), avg(\text{TDMA}_t)\}$
4: **if** $\text{SOMAC}_t \in$ OPT **then**
5: $count \leftarrow count + 1$
6: **end if**
7: **end for**
8: $optimality \leftarrow \frac{count}{T}$

Output: $optimality$

The second criterion is the *network performance*, and relates to the network performance metric chosen for the reward (e.g. average throughput or average delay as in Section 6.4). The last criterion is called *regret* and is given by Eq. 6. In RL, regret corresponds to the difference between following the optimal policy π^* and following a policy π [44]. In our context, the optimal policy π^* corresponds to the set of actions that leads to using the best MAC protocol over time. This is obtained by comparing both CSMA/CA and TDMA for each time step in our evaluations. CSMA/CA is said the best MAC protocol at the time step t if it has a better performance than TDMA at that moment. On the other hand, a policy π corresponds to the set of actions selected by SOMAC.

$$\eta = \frac{1}{T}\sum_{t}^{T} r_t(s, a|\pi^*) - r_t(s, a|\pi) \tag{6}$$

6.3. Results – Parameter Tuning

We tune the hyperparameters of Q-Learning by looking up the log files of our preliminary results. The value of ξ is empirically set to 5 (see Eq. 5). Fig. 6 presents the normalized average regret (η (%)) of SOMAC when using ϵ-greedy. Values of α and γ correspond, respectively, to the learning rate and the discount rate of Eq. 1. The ϵ-greedy solution performs best for small values of ϵ, and the best value was obtained when $\epsilon = 0.1$. Besides, the smallest percentual value of η is when $\alpha = 0.8$ and $\gamma = 0.7$, in Fig. 6a.

Fig. 7 shows the average regret when SOMAC employs softmax. The best performance is obtained when $T = 0.5$. The best values for α and γ are, respectively, 0.7 and 0.8. These values do not necessarily provide the smallest regret, however they are surrounded by the smallest values of η. We chose such values because this is a region of stability in the hyperparameters, meaning that the performance of Q-Learning should not be affected significantly for different test scenarios.

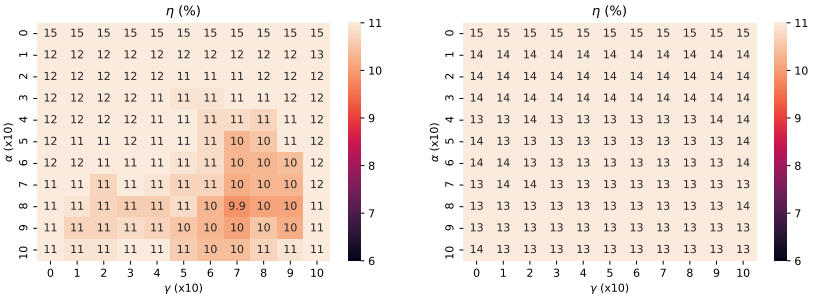
The results for UCB are depicted in Fig. 8, where the bonus function is given by $c\sqrt{ln(t)/N_t(s,a)}$ [30]. For UCB, the selected values were $\alpha = 0.9$, $\gamma = 0.5$, and $c = 2$. Similar to softmax, the choice of parameters was lead to the stability of the results when using values of such magnitudes.

Additionally, Fig. 10 displays the average regret of each exploration strategy when employing the most effective values of their respective hyperparameters. Softmax presents the smallest average regret, being slightly better than UCB, while ϵ-greedy is the worst performer. This result corroborates [34], where the authors point a better overall performance of softmax over ϵ-greedy and UCB. Further, ϵ-greedy uniformly selects actions while exploring, which may lead to a sequence of bad actions, and UCB requires periodic restarts for balancing exploration and exploitation (i.e., an extra hyperparameter, restart frequency) [45]. Softmax, on the other hand, accounts for the potential of each action while exploring and does not require periodic restarts. Therefore, we evaluate only softmax in the following sections, using its tuned hyperparameters (i.e., $\alpha = 0.7$, $\gamma = 0.8$, $T = 0.5$, and $\xi = 5$).

6.4. Results – Pre-defined topologies

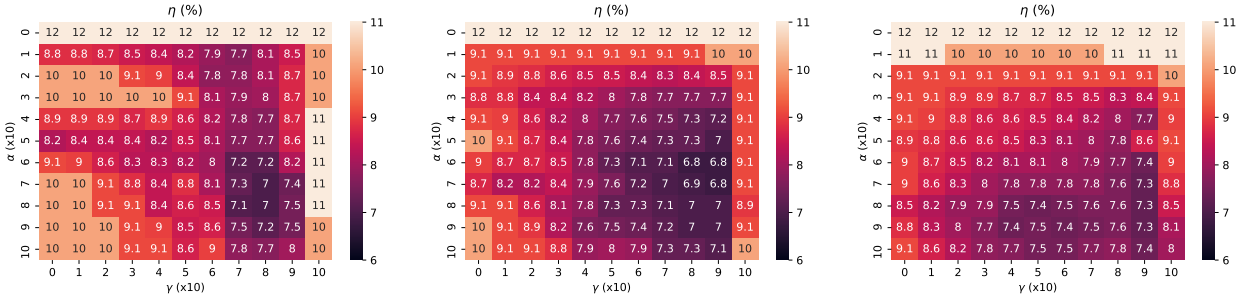
We start by evaluating SOMAC in pre-defined network configurations, comparing its performance to pure-TDMA and pure-CSMA/CA. The goal is to identify whether SOMAC is able to operate according to the best MAC protocol. Those network configurations are represented by Figures 9a-b. The vertical lines indicate when a certain node in the testbed is generating packets to be transmitted. The Y axis indicates the identifier of the node in the testbed (the numbers are the same as Fig. 5). Further, the X axis indicates the time, in minutes. The first topology represents a scenario in which a single protocol should be the best one during the entire experiment. Scenario b was carefully built so there are two moments when there should be a change of protocols (around t=30s and t=60s). Finally, in the third scenario nodes transmit at random.

For the network configurations of Fig. 9a-b, we chose the average throughput as the performance metric of the reward function (i.e., g_t of Eq. 5). Those network configurations are manually made in order to represent scenarios varying from stability to dynamicity.



a. $\epsilon = 0.1$ b. $\epsilon = 0.5$

Figure 6: ϵ-greedy – heatmap for η when varying α, γ, and ϵ



a. $T = 0.1$ b. $T = 0.5$ c. $T = 1$

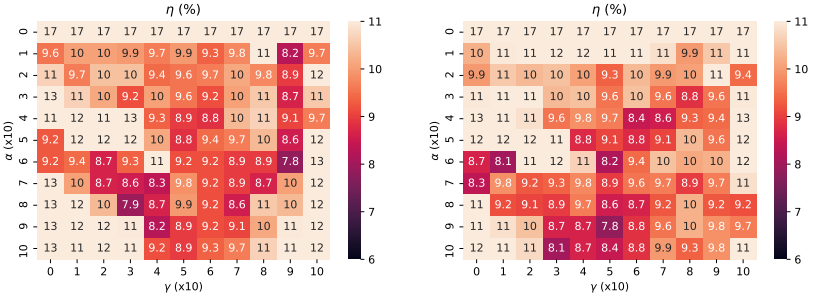
Figure 7: Softmax – heatmap for η when varying α, γ, and T

In Fig. 9a, the protocol CSMA/CA is the best MAC protocol all the time, which aims to evaluate the stability of our solutions when the network does not change over time (or changes are slow). Fig. 11 illustrates the overall result. Initially, SOMAC underperforms CSMA/CA but catches up after minute 7, reaching 90% of *optimality* in this case. The poor performance during the first minutes is explained by the random choice of the initial MAC protocol at each repetition.

Fig. 9b, on the other hand, illustrates a network configuration with two switching points (SP), representing a more dynamic network where conditions change over time. The switching points occur at minutes 20 and 60, approximately, and the resulting performance is shown in Fig. 12. In this case, SOMAC reaches 80% of optimality in total.

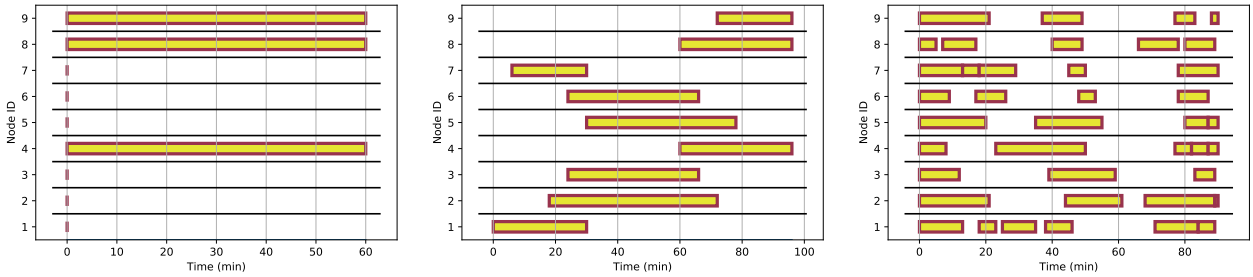
Those results are summarized in Fig. 13, where we plot the overall throughput and regret of network configurations shown in Fig. 9a-b. As we see, SOMAC outperforms both CSMA/CA and TDMA in terms of throughput and regret.

As mentioned previously, the decision engine may employ any performance metric as target metric to be optimised. In order to show this capability, Fig. 14 displays the results for latency optimisation. Latency optimisation is relevant for applications such as computation offloading [46], in which the computation can be offloaded from the end-device to the cloud in order to improve the response time for the user. It is important to highlight the log-scale of the vertical axis in Fig. 14. Both CSMA/CA and TDMA perform similarly between minutes 0 and 35, explaining the initial instability of SOMAC. From minute 35 on, SOMAC tends to follow TDMA, being nearly stable after minute 45. In addition, SO-



a. $c = 1$ b. $c = 2$

Figure 8: UCB – heatmap for η when varying α, γ, and c



a. Pre-defined topology 1, static b. Pre-defined topology 2, two SPs c. Random topology

Figure 9: Timelines of the evaluated network configurations

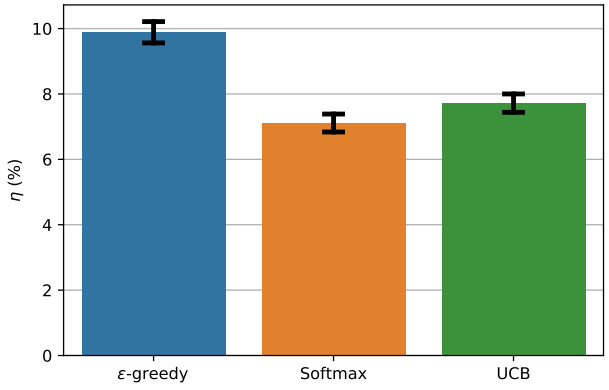


Figure 10: Overall performance of each exploration strategy – best adjustment of hyperparameters

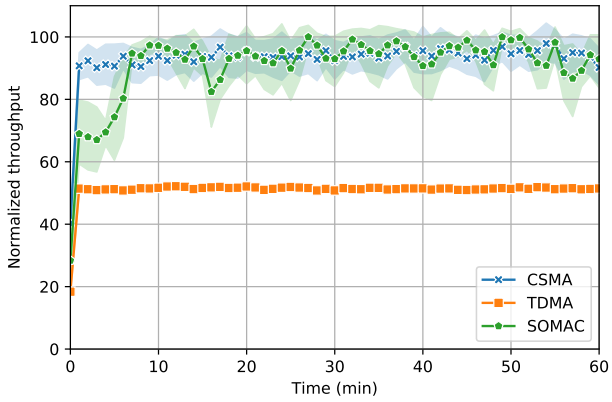


Figure 11: Static network configuration

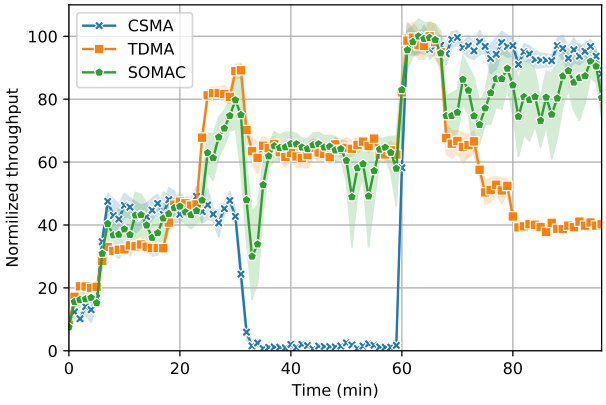


Figure 12: Network configuration with 2 switching points

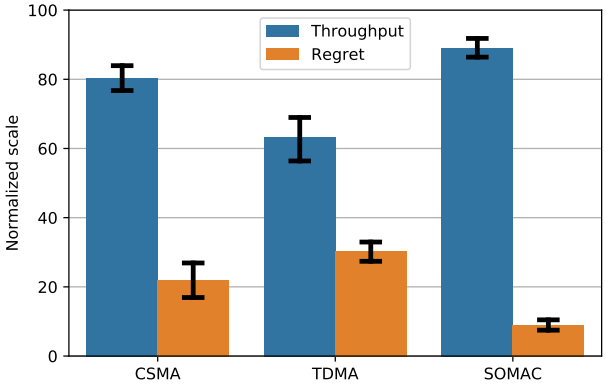


Figure 13: Overall network performance and regret

MAC has *optimality* of 84% in this case.

6.5. Results – Random topology

This scenario evaluates SOMAC under random generated network configuration, which is illustrated in Fig. 9c. The result is displayed in Fig. 15, where CSMA/CA tends to be the best MAC protocol (red arrows), except between minutes 50 and 60 (purple arrow). Notice that SOMAC performs mostly according to the best MAC protocol, being analogous to CSMA/CA except for that period, when it tends to follow TDMA.

Although the gains may seem visually modest, it is important to recall the normalized scale in Fig. 15. When performing distinctly, the difference between CSMA/CA and TDMA is up to 20% and so is, therefore, SOMAC's gains.

Only for a few moments SOMAC underperforms the best MAC protocol. More specifically, our solution lags behind others just after minute 50 and just before minute 70. There are two possible reasons for that. First, the decisions take time, so the network condition may change in-between, postponing the protocol change a few actions ahead. Second, earlier decisions may bias the algorithm to keep the protocol for longer by severely punishing the agent while changing the protocol previously. In that case, the agent is skeptical about trying another change until its Q-values are updated.

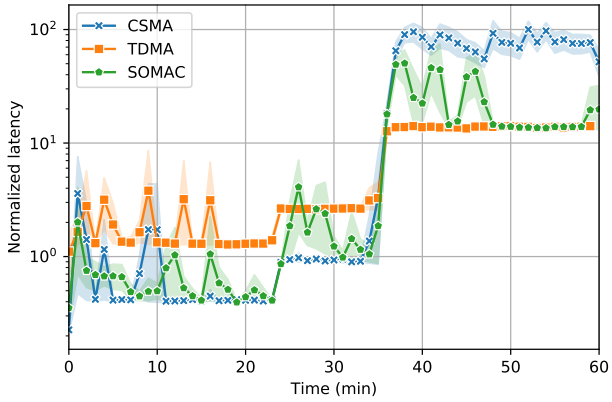


Figure 14: Latency as the performance metric

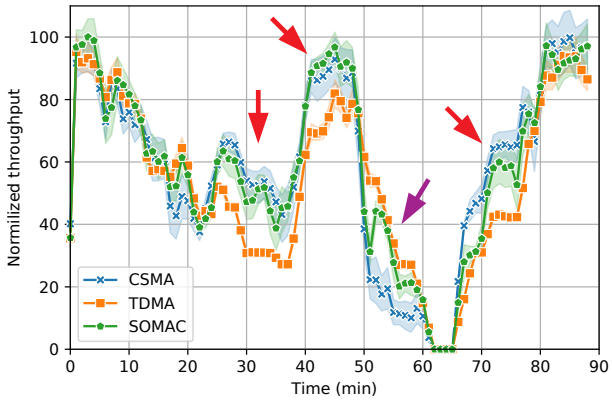


Figure 15: Random network configuration

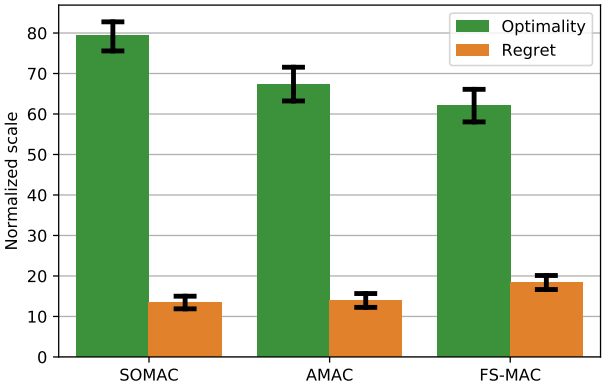


Figure 16: SOMAC vs. the State of the art in random configurations

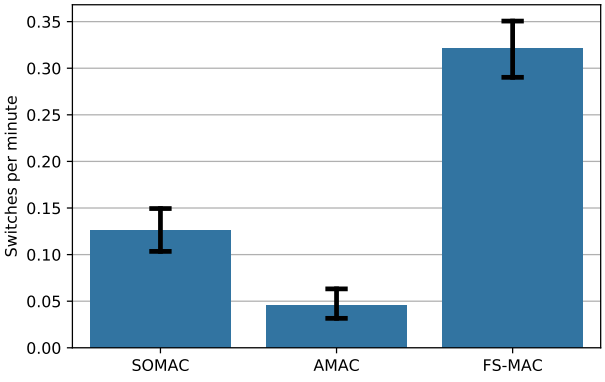


Figure 17: Switching rate in random configurations

Despite that, SOMAC still reaches 84% of optimality, performing according to the best protocol most of the time.

6.6. Results – Comparison with the state of the art

We have previously evaluated SOMAC against different network configurations, from static to dynamic cases, with an alternative performance metric, and even in a random case. As we have seen, SOMAC tends to perform according to the best MAC protocol over time. Now, we evaluate its performance against the state of the art.

As pre-defined topologies might benefit one algorithm over others, we evaluate SOMAC against its competitors in random network configurations. For that, we ran SOMAC, FS-MAC and AMAC on 10 different network configurations of 30 minutes each. The network configurations are randomly generated by selecting random transmission and idle periods for each node until the end of the experiment's time span. Fig. 16 presents the overall results. SOMAC has average *optimality* close to 80%, similar to our previous results. On the other hand, FS-MAC and AMAC have *optimality* levels lower than 70%. Indeed, FS-MAC is slightly above 60%, only 10% better than a random walk (i.e. purely random decisions). SOMAC has also the lowest regret, though it is similar to AMAC. In this case, the short time span of the experiments may explain the small difference, as neither CSMA/CA nor TDMA outperform each other for a long time.

Additionally, Fig. 17 plots the switching rate of each solution. FS-MAC has the highest switching rate, which suggests instability of the selection engine and justifies the lowest performance in Fig. 16. The lower performance is likely related to the overhead of switching protocols. SOMAC has higher switching rate than AMAC, which may also explain the similar regret between SOMAC and AMAC in Fig. 16. In this context, our solution is more overhead-prone than AMAC, even though performing better.

7. Conclusion

This paper presented an automatic MAC protocol selection sublayer based on RL called SOMAC. Our proposal employs Q-Learning to cope with non-stationary environments such as the wireless links in infrastructured networks. We implemented SOMAC using SDR and evaluated the solution in real-world wireless networks. The results indicate that SOMAC not only selects the best MAC protocol over time, with optimality greater than 80%, but also outperforms both pure-CSMA/CA and pure-TDMA protocols, overall. Further, SOMAC copes with different metrics being used as a reward, as exemplified in the evaluation. Additionally, SOMAC outperforms other dynamic protocol schemes from 10-

20% in terms of the optimality of protocol choices.

As future work, we intend to investigate the influence of multiple SOMAC masters within the same region. Each RL agent may introduce interference to each other, affecting the overall performance of the networks. Further, we plan to investigate the initialisation cost of the transition values in the RL model. With that in mind, we will investigate how to perform knowledge transfer from one deployment to another, so that the initialisation takes less time. We also intend to explore the use of Deep RL algorithms, searching for solutions to cope with the non-stationary nature of wireless networks. Finally, we also intend to improve evaluate our contribution under more realistic scenarios, for example considering bursty traffic as well as node mobility.

Acknowledgments

The authors would like to thank the following Brazilian agencies for their financial support: CAPES, CNPq, and FAPEMIG. We also thank the FUTEBOL project (European Project H2020 grant no. 688941) for partly funding this work.

References

[1] P. Ferrand, M. Amara, M. Guillaud, S. Valentin, Trends and Challenges in Wireless Channel Modeling for an Evolving Radio Access, arXiv:1606.02143 [cs, math] URL http://arxiv.org/abs/1606.02143.

[2] X. Networks, Latency & Jitter in networking performance evaluation, Tech. Rep., Xena Networks, URL https://www.xenanetworks.com/wp-content/uploads/xenadocuments/Latency-Jitter-WP.pdf, 2004.

[3] A. Technologies, Testing the performance of applications over wide area networks, Tech. Rep., Apposite Technologies, Inc, URL https://www.eiseverywhere.com/file_uploads/57cae25666ab5953f57569a2c6d0f4cb_Linktropy-WhitePaper.pdf, 2008.

[4] B. A. Sharp, E. A. Grindrod, D. A. Camm, Hybrid TDMA/CSMA protocol for self managing packet radio networks, in: Fourth IEEE International Conference on Universal Personal Communications, 929–933, 1995.

[5] R. R. Choudhury, X. Yang, R. Ramanathan, N. H. Vaidya, On designing MAC protocols for wireless networks using directional antennas, IEEE Transactions on Mobile Computing 5 (5) (2006) 477–491, ISSN 1536-1233.

[6] H. Menouar, F. Filali, M. Lenardi, A survey and qualitative analysis of mac protocols for vehicular ad hoc networks, IEEE Wireless Communications 13 (5) (2006) 30–35, ISSN 1536-1284.

[7] L. Shengbin, Z. Xiaoliang, A Survey on MAC Layer in Ad Hoc Wireless Networks, in: International Conference on Applied Informatics and Communication (ICAIC), ISBN 978-3-642-23220-6, 691–699, 2011.

[8] S. A. Gopalan, J.-T. Park, Energy-efficient MAC protocols for wireless body area networks: Survey, in: International Congress on Ultra Modern Telecommunications and Control Systems, ISSN 2157-0221, 739–744, 2010.

[9] I. Demirkol, C. Ersoy, F. Alagoz, MAC protocols for wireless sensor networks: a survey, IEEE Communications Magazine 44 (4) (2006) 115–121, ISSN 0163-6804.

[10] M. Dillinger, K. Madani, N. Alonistioti, Software defined radio: Architectures, systems and functions, John Wiley & Sons, 2005.

[11] D. F. Macedo, D. Guedes, L. F. M. Vieira, M. A. M. Vieira, M. Nogueira, Programmable Networks—From Software-Defined Radio to Software-Defined Networking, IEEE Communications Surveys Tutorials 17 (2) (2015) 1102–1125, ISSN 1553-877X.

[12] W. I. Forum, Software Defined Radio - Rate of Adoption, http://www.wirelessinnovation.org/sdr_rate_of_adoption, 2011.

[13] Sofware Radio Systems, https://www.softwareradiosystems.com/, Accessed: 2019-07-17, 2019.

[14] A. Puschmann, P. Di Francesco, M. A. Kalil, L. A. DaSilva, A. Mitschele-Thiel, Enhancing the Performance of Random Access MAC Protocols for Low-cost SDRs, in: Proceedings of the 8th ACM International Workshop on Wireless Network Testbeds, WiNTECH '13, ACM, New York, NY, USA, ISBN 978-1-4503-2364-2, 9–16, URL http://doi.acm.org/10.1145/2505469.2505481, 2013.

[15] J. R. S. Cordeiro, E. Lanza, L. F. M. V. D. F. Macedo, FS-MAC: A Flexible MAC Platform for Wireless Networks, in: IEEE Wireless Communications and Networking Conference, 2018.

[16] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Standard 802.11, 1999.

[17] Q. Wang, K. Jaffres-Runser, Y. Xu, J.-L. Scharbarg, Z. An, C. Fraboul, TDMA versus CSMA/CA for wireless multi-hop communications: a comparison for soft real-time networking, in: 2016 IEEE World Conference on Factory Communication Systems (WFCS), IEEE, 1–4, 2016.

[18] R. Nelson, L. Kleinrock, The spatial capacity of a slotted ALOHA multihop packet radio network with capture, IEEE Transactions on Communications 32 (6) (1984) 684–694.

[19] K. Bilstrup, E. Uhlemann, E. G. Ström, U. Bilstrup, On the Ability of the 802.11P MAC Method and STDMA to Support Real-time Vehicle-to-vehicle Communication, EURASIP J. Wirel. Commun. Netw. 2009 (2009) 5:1–5:13, ISSN 1687-1472, URL http://dx.doi.org/10.1155/2009/902414.

[20] X. Yang, L. Wang, J. Su, Y. Gong, Hybrid MAC Protocol Design for Mobile Wireless Sensors Networks, IEEE Sensors Letters 2 (2) (2018) 1–4, ISSN 2475-1472, URL https://ieeexplore.ieee.org/document/8340783/.

[21] I. Rhee, A. Warrier, M. Aia, J. Min, M. L. Sichitiu, Z-MAC: A Hybrid MAC for Wireless Sensor Networks, IEEE/ACM Transactions on Networking 16 (3) (2008) 511–524, ISSN 1063-6692.

[22] W. Hu, H. Yousefi'zadeh, X. Li, Load Adaptive MAC: A Hybrid MAC Protocol for MIMO SDR MANETs, IEEE Transactions on Wireless Communications 10 (11) (2011) 3924–3933, ISSN 1536-1276, URL http://ieeexplore.ieee.org/document/6025329/.

[23] Libin Jiang, J. Walrand, A Distributed CSMA Algorithm for Throughput and Utility Maximization in Wireless Networks, IEEE/ACM Transactions on Networking 18 (3) (2010) 960–972, ISSN 1063-6692, 1558-2566, URL http://ieeexplore.ieee.org/document/5340575/.

[24] Y. Yu, T. Wang, S. C. Liew, Deep-Reinforcement Learning Multiple Access for Heterogeneous Wireless Networks, in: 2018 IEEE International Conference on Communications (ICC), IEEE, 1–7, URL http://arxiv.org/abs/1712.00162, arXiv: 1712.00162, 2017.

[25] B. Jooris, J. Bauwens, P. Ruckebusch, P. De Valck, C. Van Praet, I. Moerman, E. De Poorter, TAISC: A cross-platform MAC protocol compiler and execution engine, Computer Networks 107 (2016) 315–326, ISSN 13891286, URL https://linkinghub.elsevier.com/retrieve/pii/S1389128616300974.

[26] T. Hsieh, K. Lin, P. Wang, A hybrid MAC protocol for wireless sensor networks, in: 2015 IEEE 12th International Conference on Networking, Sensing and Control, 93–98, 2015.

[27] K. C. Huang, X. Jing, D. Raychaudhuri, MAC Protocol Adaptation in Cognitive Radio Networks: An Experimental Study, in: International Conference on Computer Communications and Networks, ISSN 1095-2055, 1–6, 2009.

[28] Q. Ye, W. Zhuang, L. Li, P. Vigneron, Traffic-Load-Adaptive Medium Access Control for Fully Connected Mobile Ad Hoc Networks, IEEE Transactions on Vehicular Technology 65 (11) (2016) 9358–9371, ISSN 0018-9545, 1939-9359.

[29] M. Qiao, H. Zhao, S. Wang, J. Wei, MAC protocol selection based on machine learning in cognitive radio networks, in: International Symposium on Wireless Personal Multimedia Communications (WPMC), 453–458, 2016.

[30] R. S. Sutton, A. G. Barto, Reinforcement learning: an introduction, Adaptive computation and machine learning, MIT Press, ISBN 978-0-262-19398-6, 1998.

[31] R. S. Sutton, Dyna, an Integrated Architecture for Learning, Planning, and Reacting, SIGART Bull. 2 (4) (1991) 160–163, ISSN 0163-5719.

[32] C. J. C. H. Watkins, P. Dayan, Q-learning, Machine Learning 8 (3) (1992) 279–292, ISSN 1573-0565.

[33] A. Marinescu, I. Dusparic, S. Clarke, Prediction-Based Multi-Agent Reinforcement Learning in Inherently Non-Stationary Environments, ACM Transactions on Autonomous and Adaptive Systems 12 (2) (2017) 1–23, ISSN 15564665.

[34] A. D. Tijsma, M. M. Drugan, M. A. Wiering, Comparing exploration strategies for Q-learning in random stochastic mazes, in: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), ISBN 978-1-5090-4240-1, 1–8, 2016.

[35] M. Krasnyansky, Universal TUN/TAP device driver, https://www.kernel.org/doc/Documentation/networking/tuntap.txt, 2000.

[36] B. Bloessl, M. Segata, C. Sommer, F. Dressler, An IEEE 802.11a/g/p OFDM receiver for GNU radio, in: Proceedings of the second workshop on Software radio implementation forum - SRIF '13, ISBN 978-1-4503-2181-5, 9, 2013.

[37] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, MOA: Massive Online Analysis, Journal of Machine Learning Research (2010) 4.

[38] A. Gepperth, B. Hammer, Incremental learning algorithms and applications, in: European Symposium on Artificial Neural Networks (ESANN), 2016.

[39] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533, ISSN 0028-0836, 1476-4687.

[40] V. Kuleshov, D. Precup, Algorithms for multi-armed bandit problems, Tech. Rep., CoRR, URL http://arxiv.org/abs/1402.6028, arXiv: 1402.6028, 2014.

[41] J. Vermorel, M. Mohri, Multi-armed Bandit Algorithms and Empirical Evaluation, in: D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, J. Gama, R. Camacho, P. B. Brazdil, A. M. Jorge, L. Torgo (Eds.), Machine Learning: ECML 2005, vol. 3720, Springer Berlin Heidelberg, Berlin, Heidelberg, ISBN 978-3-540-29243-2 978-3-540-31692-3, 437–448, URL http://link.springer.com/10.1007/11564096_42, 2005.

[42] FUTEBOL, Testbeds, http://www.ict-futebol.org.br/index.php/infrastructure/testbeds/, 2018.

[43] FUTEBOL UFMG, WELCOME TO THE FUTEBOL UFMG FACILITY, http://futebol.dcc.ufmg.br, 2018.

[44] A. Garivier, E. Moulines, On Upper-Confidence Bound Policies for Switching Bandit Problems, in: J. Kivinen, C. Szepesvári, E. Ukkonen, T. Zeugmann (Eds.), Algorithmic Learning Theory, ISBN 978-3-642-24412-4, 174–188, 2011.

[45] N. Cesa-Bianchi, G. Lugosi, Prediction, learning, and games, Cambridge University Press, ISBN 978-0-511-19178-7 978-0-511-54692-1 978-0-511-18995-1 978-0-511-19059-9 978-0-511-19091-9 978-0-511-19131-2 978-0-521-84108-5, 2006.

[46] J. L. D. Neto, S. y. Yu, D. F. Macedo, J. M. S. Nogueira, R. Langar, S. Secci, ULOOF: a User Level Online Offloading Framework for Mobile Edge Computing, IEEE Transactions on Mobile Computing (2018) 1ISSN 1536-1233.