A neural-network-based MF-TDMA MAC scheduler for collaborative wireless networks

Conference Paper · April 2018

DOI: 10.1109/WCNC.2018.8377044

4 authors:

Ruben Mennes
University of Antwerp
19 PUBLICATIONS **138** CITATIONS
SEE PROFILE

Miguel Camelo
University of Antwerp
42 PUBLICATIONS **221** CITATIONS
SEE PROFILE

Maxim Claeys
Ghent University
35 PUBLICATIONS **880** CITATIONS
SEE PROFILE

Steven Latré
University of Antwerp
236 PUBLICATIONS **3,000** CITATIONS
SEE PROFILE

A Neural-Network-based MF-TDMA MAC Scheduler for Collaborative Wireless Networks

Ruben Mennes, Miguel Camelo, Maxim Claeys and Steven Latré
IDLab, Department of Mathematics and Computer Science, University of Antwerp - imec
Email: {ruben.mennes, miguel.camelo, maxim.claeys, steven.latre}@uantwerpen.be

***Abstract*—In the unlicensed spectrum, many wireless technologies (e.g Wi-Fi, Bluetooth) use the same spectrum for wireless transmission. This often results in cross-technology interference effects, which are hard to address. Without new methods to manage this shared spectrum, wireless communication is increasingly challenging as too many nodes attempt at accessing the same spectrum. Collaboration between different wireless networks that use the same spectrum will be required to handle this massive amount of devices. In this paper, we present two algorithms based on Neural Networks (NNs) to demonstrate that a function approximation can accurately predict free slots in a Multiple Frequencies Time Division Multiple Access (MF-TDMA) network. By observing the spectrum, we are able to do online learning and let the corresponding NN predict the behavior of the spectrum a second in advance using our approach. We are able to reduce the number of collisions by half if the nodes from other networks are sending data following a Poisson distribution. When the nodes of the other network follow a more periodic traffic pattern, a collision reduction of factor 15 could be achieved.**

I. Introduction

Nowadays, the radio spectrum is managed by governmental agencies and divided into frequency bands of fixed sizes. Although most of these are reserved for license holders, there are several unlicensed bands, the so-called industrial, scientific and medical (ISM) radio bands, that can be accessed and used by anyone for free. These bands allow the implementation and deployment of several wireless network technologies such as Wi-Fi (on 2.4 GHz and 5GHz), Bluetooth (on 2.4 GHz), and ZigBee (on 915 MHz and 2.4GHz). As some of these technologies are used daily and can be found in almost every mobile device, also the number of users is increasing exponentially [1]. These two factors, multiple technologies sharing the spectrum and the massive amount of users, are leading to a saturation of the spectrum and impact the network performance and the user's Quality of Experience. An example of a multi-technology setup is illustrated in Figure 1.

One of the major problems with the network technologies in the ISM radio bands is that they try to optimize their own network performance without considering sharing access of the spectrum with other technologies. As a result, radio interference between networks leads to suboptimal optimization results.

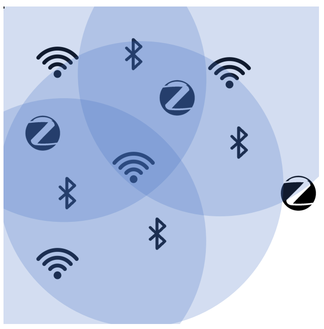

Fig. 1. Coverage of a multi-technology setup with Wi-Fi, Bluetooth and ZigBee Nodes. All sharing the same (unlicensed) spectrum

In order to improve network performance and to allow sharing access to the medium with a low probability of packet loss, several wireless Medium Access Control (MAC) protocols have been proposed. One of the features that these MAC protocols implement is a listen-before-talk strategy, such as the RTS/CTS (Request To Send/Clear To Send) flow control mechanism. This method tries to avoid collisions by creating a coordinated access to the medium using control signals between nodes. Other MAC protocols use a Time Division Multiple Access (TDMA) schedule that allows defining specific time slots to each user to transmit and receive without interference. In recent years, these protocols have been extended and combined with the simultaneous use of multiple frequencies (e.g. Multiple Frequencies Time Division Multiple Access (MF-TDMA)) [2]. However, these approaches alone are not enough to solve the problem of interference between multiple technologies since they are only used among nodes using the same technology. In latest years, MF-TDMA communication is quite well-investigated, both in more traditional wireless networks [3], as in new sensor wireless network technologies such as 6TiSCH [4] but to the best of our knowledge, there is no MF-TDMA algorithm that is able to learn cross-technology behavior to avoid cross-technology interference.

In this paper, we present two algorithms to create a schedule for accessing the medium in MF-TDMA-like networks. These algorithms avoid interference by learning the correct frequency and time slot to receive packets without global information. Both algorithms apply the same Neural Network (NN) to learn the behavior of surrounding networks. While the first algorithm only uses local information from both the environment and their previous schedules, the second algorithm also uses a positive feedback loop to learn an even better schedule. Using the NN, a probability matrix is constructed. These probability matrices are used in a centralized scheduling algorithm to create a MF-TDMA schedule, but it should also be possible to use these probabilities to augment existing (centralized or even distributed) algorithms to improve their efficiency.

The remainder of this paper is structured as follows: related work is discussed in Section II. The formal problem statement is presented in Section III. Both algorithms are described in Section IV. In Section V the results of the performance evaluations are presented and analyzed. Finally, conclusions and future work are described in Section VI.

II. Related work

A. TDMA Scheduling

Scheduling in Wireless Sensor Networks (WSNs) is a well-studied topic in literature. Depending on the network management and control, the scheduling algorithms can be classified into two main groups: centralized and decentralized. In the centralized environment, the network nodes send all their knowledge about topology, routing and application-specific parameters to a unique node. This node runs the scheduling algorithm and broadcasts its results back to all nodes [5]. Typically a few slots in the schedule are reserved for sending all information to the root of the network and a schedule back to the nodes. It is clear that these schedulers can create a collision-free schedule, for the network only, based on the common knowledge. On the other hand, it is also clear that these approaches require a significant amount of control messages to setup a schedule. These centralized algorithms work well in static networks where the central node can be easily reached by the other nodes.

Secondly, we have the decentralized algorithms. The first proposal for a decentralized Time Synchronized Channel Hopping (TiSCH) scheduler was made by Tinka et al. [6]. The work of Tinka et al. was never broadly implemented but was the inspiration for several other decentralized algorithms. Some schedulers use a Resource Reservation Protocol (RSVP)-like approach, where a chain of messages pass from source to destination while on each node a slot is allocated [7]. This approach is very efficient to allocate flows in a more or less static network. From the moment flows need to change the route, the reservation needs to start over. Other approaches only exchange messages with their current neighbors to allocate cells in the current schedule. The DeBras algorithm [8] performs better with changes in routes, because the allocation happens only with the neighboring nodes where first slots are proposed at the transmitter. The receiver needs to confirm part of the slots before the slots will be allocated in the actual schedule. The advantage of the protocol is that it can easily use the On-the-Fly bandwidth reservation algorithm [9]. This algorithm can be used to calculate the number of cells you need to reserve in the schedule to sleep as much as possible and is very dynamic.

B. Machine Learning for wireless network management

Machine Learning (ML) is not new in wireless telecommunication systems. There is some literature that already included ML for solving problems on different layers of the wireless stack [10, 11].

Reinforcement Learning (RL) is one of the ML techniques widely used in this context. The idea behind RL is that an agent learns a certain model based on the experience it achieves by acting in the environment. The advantage of this system is that it can learn during the lifetime of the algorithm or node. In order to apply RL in a decentralized environment, e.g. a wireless network, a framework such as Multi-Agent Reinforcement Learning (MARL) [10] or game theory [12] needs to be applied. In this context, every network node is a learning agent and learns how to solve a specific problem, either by itself or by collaborating with others. Other techniques like Swarm intelligence, where agents act as an ant colony or birds exchanging side by side information, can be used for solving routing problems [10].

In addition to the application of ML to solve problems at the network layer, there are several works at the MAC layer as well. Chu et al. [13] used Q-Learning to improve Slotted ALOHA where they achieve twice the maximum throughput of Slotted ALOHA while being more energy efficient. For each node, they artificially create a frame that contains N slots. Each slot in the frame represents a score in the Q-Table. Based on the Q-Learning algorithm, these values can change if there was a collision or a successful communication. This approach maintains the very low complexity of ALOHA, combined with a quite easy Q-Learning approach. This makes the ALOHA-QIR very light-weighted, and ideal for sensor nodes. If the amount of nodes and data grows, this approach will not scale anymore, this is because the approach can use only a single channel. In the approach of this paper, more channels can be used and this means more bandwidth is available to be used.

III. Formal problem statement

We assume a network with N MF-TDMA nodes, of which one acts as the root node. All the nodes can communicate to each other on C channels by a single-hop communication. These channels are also the frequencies that are used in the MF-TDMA schedule. Each channel is divided into time slots, and S time slots form a frame. In order to execute a correct scheduling, all the nodes need to have the same notion of time. We assume that all nodes implement an algorithm to synchronize their clocks [14]. This means that all the nodes will execute the same frame f for all frames F. In each slot, all nodes are synchronized at the same time (f, s), frame, slot number, and node n execute the action $A^{f,n}_{s,c}$ on channel c. This action is an action from the set $A = \{\text{Idle}, \text{TX}, \text{RX}\}$, where:

$$TX^{f,n}_{s,c} = \begin{cases} 1 & \text{if } A^{f,n}_{s,c} = \text{TX} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$RX^{f,n}_{s,c} = \begin{cases} 1 & \text{if } A^{f,n}_{s,c} = \text{RX} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

In this model, we include M external network nodes or bots. These bots access the spectrum in an independent way, there is no communication between them and the nodes in the MF-TDMA network. $TXB^{f,m}_{s,c} = 1$ if and only if bot m uses channel c at time (f, s) to transmit.

In this paper, we focus on a centralized scheduler based on the work of Palattella et al. [15]. Therefore, we assume that before every frame all N nodes exchange information about the next frame with the root. The next frame starts once all the nodes received the new schedule.

Communication is successful if and only if there is only one node (or bot) transmitting data (i.e., uses the TX action) at each time in each channel, and at least there is one network node, the destination node, ready to receive data at the same slot and channel. The destination node always listens to the correct channel at the correct time since a correct scheduling ensures it. Follows from this, we could simplify the assumption. Communication is successful if and only if there is only one node (or bot) that is sending at the same time in the same channel. We use $\gamma^f_{s,c}$ to determine if there was successful communication at time (f, s) on channel c.

$$\gamma^f_{s,c} = \begin{cases} 1 & \text{if } \sum_n TX^{f,n}_{s,c} = 1 \wedge \sum_m TXB^{f,m}_{s,c} = 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

In the same way, it is possible to define when a collision occurs in a particular time slot. Let $\omega^f_{s,c} = 1$ if and only if there was a collision on time (f, s).

$$\omega^f_{s,c} = \begin{cases} 0 & \text{if } \sum_n TX^{f,n}_{s,c} + \sum_m TXB^{f,n}_{s,c} \leq 1 \\ 1 & \text{otherwise} \end{cases} \tag{4}$$

Since the number of collisions among network nodes (internal nodes and external ones as bots) and the number of assigned slots to RX/TX per nodes impact the throughput of the network, we use this metric to define the objective function to optimize. In that case, we maximize the throughput Γ^f for each frame f. Where Γ^f is the number of successful packets in frame f.

$$\Gamma^f = \sum_{s\in[0,S)} \sum_{c\in[0,C)} \gamma^f_{s,c} \tag{5}$$

In order to increase the throughput Γ^f we can either (i) increase the amount of TX/RX cells, (ii) generate more data, or (iii) decrease the number of collisions. Because data generation is fixed and tries to ensure a fair use of the spectrum , we focus on increasing throughput by reducing the collision among nodes. This means that the only way of increasing the throughput is to decrease the number of collisions. So actually we want to minimize Ω^f for each frame f.

$$\Omega^f = \sum_{s\in[0,S)} \sum_{c\in[0,C)} \omega^f_{s,c} \tag{6}$$

We assume that every node n can measure energy on all the C channels at every time slot (f, s). Note that this assumption is valid since several radio hardware technologies support simultaneous energy measurement on multiple channels. For every slot-channel pair (s, c) of every frame f, we could make an observation $o^{f,n}_{s,c}$. A time slot is used if the amount of energy on channel c at time (f, s) is bigger than a given threshold θ. For every frame f we could generate an observation matrix $O^{f,n}$ for every node n, where $O^{f,n}_{s,c} = 1$ if and only if $o^{f,n}_{s,c} > \theta$.

IV. Centralized AI scheduler

In this section, we address the algorithmic details of both algorithms. The algorithms learn, for each node, which slots will be free in the next frame and therefore no other nodes in the network will use them. Note that it also means that the nodes in the network learn and predict the behavior of the bots. A state matrix $X^{f,n} \in \mathbb{R}^{S\times C}$ is created for each node n in each frame f. These matrices contain the state of the network in frame f. With the use of a NN, we predict the probabilities that a slot-channel-pair is used. We use the symbol $\tilde{X}^{f,n} \in \mathbb{R}^{S\times C}$ as a representation of a predicted state.

At the end of each frame f, every node n predicts its probability matrix $\tilde{X}^{f+1,n}$ and forwards this information to the root node together with the traffic information. The root node creates a schedule when it receives all the probability matrices of all nodes in the network. The centralized scheduler first selects the slots with a higher probability of being free and allocates these slots first for receiving. In other words, the nodes actually indicate when they are able to receive packets. This is also called receiver-initiated communication [16]. The root broadcasts the full schedule A^{f+1} before the frame starts.

$$f(X^{f-t+1,n}, X^{f-t+2,n} \cdots, X^{f-1,n}, X^{f,n}) = \tilde{X}^{f+1,n} \tag{7}$$

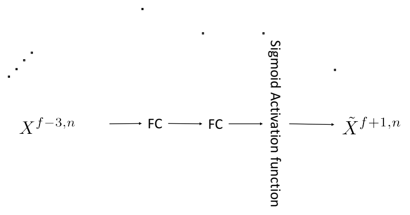

Fig. 2. Neural network used in each node n to predict the next state.

Our algorithms try to predict if a slot will be free based on some history, this leads us to a function approximation. As shown in Equation 7, a function approximation can be based on t observations to predict to next frame. Function approximation with NN is also a well-studied area [17].

As shown in Figure 2, the NN takes the four last frame states as input. The NN we use takes the four last frames state as input. There are two fully connected hidden layers. The first fully connected hidden layer takes $S \times C \times t$ input and output neurons, while the second fully connected layer takes again $S \times C \times t$ input neurons, but only $S \times C$ output neurons followed by a sigmoid activation function. The output is the $S \times C$ probability matrix $\tilde{X}_{f+1,n}$. Note that we choose to use a small NN to decrease the execution time just for a proof of concept.

The training of the network follows an online approach. This means that we store a sliding window of the last states. If there are enough samples in the sample pool we train the neural network (on the fly). Note that these samples can be also used to train the NN before it starts. To simulate this scenario we delayed the startup of the nodes. During the startup period, only the bots are active, while the MF-TDMA nodes are all idle. Note that we used an Adam Optimizer with a learning rate that is equal to 10^{-3}, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$.

A. Without ACK sharing

$$X^{f,n}_{s,c} = \begin{cases} 0 & \text{if } O^{f,s}_{s,c} = 1 \wedge \forall i \in [0, N) : A^{f,i}_{s,c} \neq \text{TX} \\ 1 & \text{if } O^{f,s}_{s,c} = 0 \\ 0.5 & \text{otherwise} \end{cases} \tag{8}$$

The first algorithm only uses broadcast information. This means that every node n can only use the observation frames $O^{f,n}$ and the schedule information. We assign the state information as noted in Equation 8. The algorithm assigns 0 to the slot-channel-pair if it measures energy at the slot-channel and no other node is sending on that channel-slot. Note that no other node will use that slot because we assume that the root broadcasts the complete schedule to all the nodes. Otherwise, if the node did not measure any energy on the channel, the algorithm assigns 1 to the slot-channel state. In the last case, the node measured some energy on a specific slot-channel-pair that is used in the scheduled. The algorithm assigns 0.5 to that state. This because it could be the TX-node or it could be a bot. The node cannot distinguish between both situations since it was not in RX-mode, and therefore can't determine if the node received the packet of the neighbor node, or if there was a collision, or if it received another signal.

B. With ACK sharing

$$X^{f,n}_{s,c} = \begin{cases} 0 & \text{if } O^{f,s}_{s,c} = 1 \wedge \forall i \in [0, N) : \beta^{f,i}_{s,c} = 0 \\ 1 & \text{if } O^{f,s}_{s,c} = 0 \vee \\ & \left[\text{if } O^{f,s}_{s,c} = 1 \wedge \exists i \in [0, N) : \beta^{f,i}_{s,c} = 1\right] \end{cases} \tag{9}$$

The second algorithm extends the first one by adding Acknowledgement (ACK) information from the receiver side. Every node broadcast the ACK information to all other nodes in the network. Let us define $\beta^{f,n}_{s,c}$ to be 1 if node n received an ACK on the packet send on slot s and channel c of frame f. The ACK broadcasts information is sent at the end of each frame. This enables us to make the state a bit more precise, as defined in Equation 9. With this information, it is possible to avoid the 0.5 value case in our states. In the case that the node measured no energy, we assign 1 to that observation state, just like in subsection IV-A. If the node discovered energy on a specific slot-channel-pair there are two cases: (i) when another node received an ACK, the algorithm assigns 1 to the state. Here we assume that the detected energy was from the TX-node. (ii) No other node broadcast an ACK for that slot-channel-pair. The algorithm assigns 0 to the state because it is clear that there was a collision on the channel.

V. Performance evaluation

A. Evaluated use case

We simulated a wireless setup to verify the proposed algorithms. There are five nodes in the MF-TDMA network. They are located at random positions in a room of 100 $\times$ 100 meter and all the nodes can communicate with each other. The number of available channels to transmit $C = 2$ and each frame contains $S = 101$ slots. All nodes generate traffic following a Poisson process with $E[W] = 5$. This means that on average each node has new data available every five slots. We assume that the amount of packets that need to be sent is known at the beginning of each frame. Algorithms as On-the-Fly introduced by Palattella et al. [9] can be used to calculate the number of cells needed. The destination of the packets is equally distributed, so every node has, on average, the same amount of packets for each other node in the same frame.

Additionally, the simulated setup contains three bots. These bots are located on the diagonal of our 100x100 meter room and generate traffic. Based on the traffic pattern of the bots, both use cases have two different scenarios. In the first scenario, the bots generate traffic with a fixed interval (in our case every six slots). In a second scenario, traffic is also generated following a Poisson process with $E[W] = 6$. For both scenarios, each bot b has also a probability matrix that is used to determine which channel is used in the next slot, based on the current channel. The probability matrix is built in such a way that there is always at least a probability of 60% to follow the same pattern.

B. Experimental setup

To verify our approach, we implemented both algorithms in a 6TiSCH discrete event simulator written in Python by Palattella et al. [9]. We implemented the topology defined in subsection V-A and extended the simulator by adding the possibility to have 2 kinds of nodes to include nodes that can behave as bots and, in general, behave as nodes from different networks. Because the simulator is a discrete event simulator, the bots and nodes both have the same notion of time but this will not influence the proof of concept for this paper. To speed up the training process, the n nodes share the same NN in the simulator. This reduces execution time and reduces the memory usage. It should be clear that if we use pre-trained NNs, this will boil down to the same. We also simulate pre-trained learning by delaying the first packets of the nodes with a variable δ frames.

As a base reference, we executed a scenario where the root did not use the probability matrices $\tilde{X}^{f,n}$. This reference could be any centralized scheduling algorithm because our scheduler will not schedule any collision between the nodes.

C. Results description

To have a notion of collisions, we have defined the collision ratio metric as the number of collisions in the node network divided by the amount of total sent packets in the node network. Figure 3 and Figure 4 show the performance of the first scenario. In this scenario, the bots are sending with a default interval of six slots with a probability of 0.7 or 1. Every simulation was executed three times and the results were averaged. Because the nodes send packets following a Poisson distribution, the resulting collision ratios were smoothed by using a Savitzky-Golay smoothing filter of order three with a window size of 21 frames. Figure 3 shows that the algorithm learns the behavior of the bots after a few frames, the predictions are accurate enough to avoid 83% of the collisions. In Figure 4, the average collision ratio in the last 600 frames of the simulation is presented. Our algorithm reduces the number of collisions up to 15 times, in comparison with the reference, allowing to increase the overall throughput of the network. Note that when we use the second algorithm, which includes the ACK information into the model, the algorithm learns a near-optimal schedule with no need of a sending delay in the nodes to pre-train the network. In the case where the bots send every six frames, the average amount of collisions without learning at all is around the 16.17%. While, by using one of the algorithms described before using the NN, the number of collisions dropped under 0.07% or less. For the scenario where the bots only send in 70 % of the cases, the reference has on average collisions in 11.63% of the cases. In the last case, our algorithm reduces the number of collisions to 1.42% or less.

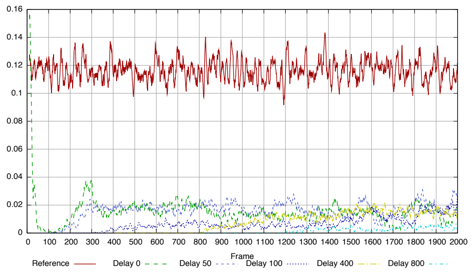

Fig. 3. Collision Ratio in the scenario with regular bots sending with ratio of 0.7 without ACK.

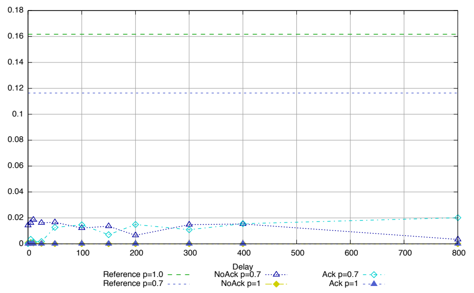

Fig. 4. Average collision ratio for every delay interval in window frame [1400, 2000] for the scenario without Poisson bots.

In the second scenario, we changed the behavior of the bots and they send data following a Poisson distribution. Due to the small size of the NN used in the algorithm and the short training/learning period, the prediction of the bot's behavior is not accurate anymore and the performance drops. Figure 5 shows the average amount of collisions between frame 1400 and 2000 for the different algorithms and different sending delays for 100 simulations each. It is possible to see that a pre-training phase, or sending delay, is necessary to have advantages of the algorithms. The performance of the algorithm is worse if it did not use a training phase. On the other hand, if we include the training phase the algorithm can decrease the number of collisions from 11% to 5% for the first algorithm, and for the second algorithm, it can decrease the amount of collisions even to 2.5%. In the scenario where the bots send with a probability of 100%, the algorithm can even reach collision ratio lower than 4.6% and 1.9% on each scenario.

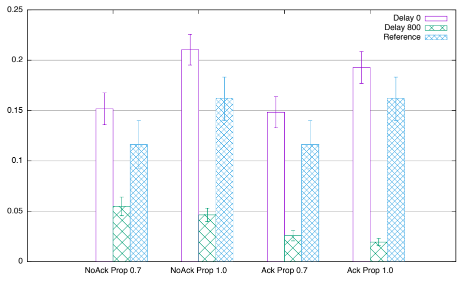

Fig. 5. Average collision ratio and standard deviation different algorithms for scenario with Poisson bots.

VI. Conclusion and future work

This paper presents two algorithms that can be used to improve the RX/TX scheduling in a MF-TDMA like network by avoiding traffic from other nodes. Both algorithms use a NN to learn the behavior of the nodes and trying to avoid the transmission pattern they are using. By simulating different scenarios in a discrete event simulator, we created a proof of concept for the algorithms, using a NN to predict the probabilities that a slot in the next frame is used, i.e. it is not free to TX/RX for the node, by the surrounding networks. The simulation results showed that using a pre-trained NN, the collisions with other networks are reduced by half. But if the pattern is even more predictable, it becomes even easier to predict the free slots and to avoid up to 15 times more slots than a non-collaborative scheduler. The algorithms use a low dimensional NN that is built for function approximation. to predict the probabilities of using a slot in the next frames based on previous observations.

Acknowledgment

This work has been funded by Research Foundation - Flanders (FWO) and the Flemish Government.

References

[1] I. Cisco Systems, "Cisco visual networking index: Global mobile data traffic forecast update, 20162021 (white paper," pp. 1–35, 2017.

[2] C. Cormio and K. R. Chowdhury, "A survey on mac protocols for cognitive radio networks," *Ad Hoc Networks*, vol. 7, no. 7, pp. 1315–1329, 2009.

[3] A. Raniwala and T.-c. Chiueh, "Architecture and algorithms for an ieee 802.11-based multi-channel wireless mesh network," in *INFOCOM 2005*, vol. 3. IEEE, 2005, pp. 2223–2234.

[4] D. Dujovne, T. Watteyne, X. Vilajosana, and P. Thubert, "6tisch: deterministic ip-enabled industrial internet (of things)," *IEEE Communications Magazine*, vol. 52, no. 12, pp. 36–41, 2014.

[5] Á. A. Farías and D. Dujovne, "A queue-based scheduling algorithm for pce-enabled industrial internet of things networks," in *Embedded Systems (CASE), 2015 Sixth Argentine Conference on*. IEEE, 2015, pp. 31–36.

[6] A. Tinka, T. Watteyne, K. Pister, and A. M. Bayen, "A decentralized scheduling algorithm for time synchronized channel hopping," in *ADHOCNETS*. Springer, 2010, pp. 201–216.

[7] A. Morell, X. Vilajosana, J. L. Vicario, and T. Watteyne, "Label switching over ieee802. 15.4 e networks," *Transactions on Emerging Telecommunications Technologies*, vol. 24, no. 5, pp. 458–475, 2013.

[8] E. Municio and S. Latré, "Decentralized broadcast-based scheduling for dense multi-hop TSCH networks," *Proceedings of the Workshop on Mobility in the Evolving Internet Architecture - MobiArch '16*, pp. 19–24.

[9] M. R. Palattella, T. Watteyne, Q. Wang, K. Muraoka, N. Accettura, D. Dujovne, L. A. Grieco, and T. Engel, "On-the-fly bandwidth reservation for 6tisch wireless industrial networks," *IEEE Sensors Journal*, vol. 16, no. 2, pp. 550–560, 2016.

[10] A. Forster, "Machine learning techniques applied to wireless ad-hoc networks: Guide and survey," in *Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007. 3rd International Conference on*. IEEE, 2007, pp. 365–370.

[11] T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.

[12] Z. Han, *Game theory in wireless and communication networks: theory, models, and applications*. Cambridge University Press, 2012.

[13] Y. Chu, P. D. Mitchell, and D. Grace, "ALOHA and Q-Learning based medium access control for Wireless Sensor Networks," *2012 International Symposium on Wireless Communication Systems (ISWCS)*, pp. 511–515.

[14] F. Sivrikaya and B. Yener, "Time synchronization in sensor networks: a survey," *IEEE network*, vol. 18, no. 4, pp. 45–50, 2004.

[15] M. R. Palattella, N. Accettura, M. Dohler, L. A. Grieco, and G. Boggia, "Traffic aware scheduling algorithm for reliable low-power multi-hop ieee 802.15. 4e networks," in *Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on*. IEEE, 2012, pp. 327–332.

[16] J. Garcia-Luna-Aceves and A. Tzamaloukas, "Receiver-initiated collision avoidance in wireless networks," *Wireless Networks*, vol. 8, no. 2/3, pp. 249–263, 2002.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press.