Agent-Based Modeling for Distributed Decision Support in an IoT Network

M. Majid Butt, *Senior Member, IEEE*, Indrakshi Dey, *Member, IEEE*, Merim Dzaferagic, Maria Murphy, Nicholas Kaminski, and Nicola Marchetti, *Senior Member, IEEE*

***Abstract*—An increasing number of emerging applications, e.g., Internet of Things (IoT), vehicular communications, augmented reality, and the growing complexity due to the interoperability requirements of these systems, lead to the need to change the tools used for the modeling and analysis of those networks. Agent-based modeling (ABM) as a bottom-up modeling approach considers a network of autonomous agents interacting with each other, and therefore represents an ideal framework to comprehend the interactions of heterogeneous nodes in a complex environment. Here, we investigate the suitability of ABM to model the communication aspects of a road traffic management system as an example of an IoT network. We model, analyze, and compare various medium access control (MAC) layer protocols for two different scenarios, namely uncoordinated and coordinated. Besides, we model the scheduling mechanisms for the coordinated scenario as a high-level MAC protocol by using three different approaches: 1) centralized decision maker (DM); 2) DESYNC; and 3) decentralized learning MAC (L-MAC). The results clearly show the importance of coordination between multiple DMs in order to improve the information reporting error and spectrum utilization of the system.**

***Index Terms*—Agent-based modeling (ABM), complex communications systems (CCSs), Internet of Things (IoT).**

I. INTRODUCTION

IN THE context of this article, a complex system is defined as any system featuring a large number of interacting components (agents, processes, etc.) whose aggregate activity is nonlinear (not derivable from the summations of the activity of individual components) and typically exhibits hierarchical self-organization under selective pressures [1]. Considering this definition of complex systems, the next generation of communication networks [e.g., Internet of Things (IoT), cellular networks, and vehicular networks] can be regarded as a complex system due to a growing number of technologies and connected devices. Complexity in decision making (scheduling and routing) for a large IoT system requires new modeling and decision-making tools and methodologies, which motivates its study by means of complex systems science (CCS) [2], [3]. The tools used to model and analyze these networks must evolve in order to optimally utilize the available resources (e.g., spectrum and processing power) at affordable complexity.

Agent-based modeling (ABM) is a bottom-up approach of modeling that considers a network of autonomous agents. Each agent has its own set of attributes and behaviors. These behaviors describe how the agents interact with other agents and their environment. If needed, the agents can exhibit learning capabilities that allow them to adapt to changes in the system, altering the internal attributes and the behaviors toward other agents. Therefore, ABM is suitable to model complex systems [4]–[6] that would require large computational complexity to be modeled otherwise.

ABM has previously been used to model a wide range of applications in sectors, such as ecology, biology, telecommunications, and traffic management. Some examples include: [7] where ABM is used to model intracellular chemical interactions and [8] to analyze the parking behaviors in a city.

Recently, ABM has been used to solve various complex problems in telecommunication networks. Savaglio *et al.* [9] provides a survey of recent works on the use of ABM for modeling IoT systems. Laghari and Niazi [10] showed how a cognitive agent-based computing modeling approach, such as ABM, can be used to model complex problems in the domain of IoT. Following an approach similar to [10], this article examines the use of ABM to model an IoT network that requires distributed decision making. The IoT network in question is a road traffic management system that adjusts the timing of traffic lights based on the number of vehicles waiting at an intersection. Our focus is on the modeling of the communication aspects of the system and in particular, we want to analyze the impact of the medium access control (MAC) protocol on the application itself, i.e., the timing of traffic lights.

Many applications of ABM in the telecommunications industry have focused on economic and social aspects such as consumer behavior. Twomey and Cadman [11] modeled customer behavior in a telecommunication network. Also, in [12], ABM is used to analyze the wireless cellular service market. There have been quite a few applications of ABM that model the network itself. Tonmukayakul and Weiss [13] described how ABM can be applied to model spectrum sharing techniques in future 5G networks. The authors model a system that considers economical, technical, and regulatory considerations

Manuscript received April 9, 2019; revised June 30, 2019, August 21, 2019, and February 7, 2020; accepted February 13, 2020. Date of publication February 27, 2020; date of current version August 12, 2020. This work was supported in part by the Science Foundation Ireland and in part by the European Regional Development Fund under Grant 13/RC/2077. *(Corresponding author: Merim Dzaferagic.)*

M. Majid Butt was with Trinity College Dublin, Dublin 2, Ireland. He is now with Standardization Research, Nokia Bell Labs, 92160 Nozay, France (e-mail: majid.butt@nokia-bell-labs.com).

Indrakshi Dey, Merim Dzaferagic, Maria Murphy, Nicholas Kaminski, and Nicola Marchetti are with the CONNECT Center for Future Networks, Trinity College Dublin, Dublin 2, Ireland (e-mail: deyi@tcd.ie; dzaferam@tcd.ie; nicola.marchetti@tcd.ie).

Digital Object Identifier 10.1109/JIOT.2020.2976802

TABLE I
COMPARISON OF ABM AND MATHEMATICAL MODELING APPROACHES

Property	Mathematical Model	ABM
Response to system state change	Difficult to capture	Convenient to capture
Implementation complexity for large systems	Moderate	High
Modeling of heterogeneous agents	Not possible	Possible
Response to randomness	Difficult to model	Relatively easy
Access to intermediate results	Difficult to access	Easily available

when leasing spectrum. The agents are able to remember what spectrum sharing conditions were beneficial for them previously and learn/adapt based on their previous choices. Horváth *et al.* [14] analyzed the spectrum trading mechanism by modeling the heterogeneous nodes as agents in an ABM framework. The motivation for ABM came from the emergence of structures, patterns, and unexpected properties. ABM allowed them to model and understand market models with dynamics that are beyond the scope of familiar analytical formulations such as differential equations.

Other applications of ABM to communication networks are presented in [15]–[17]. Niazi and Hussain [15] analyzed the effectiveness of ABM to model self-organization in peer-to-peer and *ad hoc* networks. They also outlined the limitations of using current modeling and simulation software. Their work shows that tools, such as OMNeT++ and Opnet, and specialized tools such as the Tiny OS simulator is limited as they tend to focus solely on computer networks. Interactions with humans and mobility cannot be modeled with enough flexibility. Network parameters can be easily modified but other conditions are difficult to be considered. The authors highlight the flexibility of an ABM approach, showing how easily the system can be updated and allow for powerful result abstraction. Liu *et al.* [16] analyzed a decentralized spectrum resource access model as a complex system, modeling the decentralized decision making and cooperation of distributed agents in a way that allows them to partially observe the state of the system, meaning that each agent has only the information about its own local environment. Niazi and Hussain [17] introduced an ABM framework to formally define all necessary elements to model and simulate a wireless sensor network (WSN). As a proof of concept, they demonstrated the application of the framework to a model of self-organized flocking of animals monitored by a random deployment of proximity sensors. Hernández *et al.* [18] provided an ABM simulation framework applied to IoT networks and show results on the suitability of their proposed model to evolving IoT networks. These studies provide further motivation to our investigation on the use of ABM for the dynamic scenarios where ABM is more suitable as compared to other optimization methods. A very recent work in [19] summarizes when continuous-learning-based approaches can outperform model-based approaches in network optimization.

Many studies have been carried out to try to optimize traffic flow (specifically in urban areas) using WSN. For example, in [20]–[22], sensor networks for monitoring traffic are proposed. Ma *et al.* [23], Hager *et al.* [24], and Lansdowne [25] used ABM for traffic optimization and simulation. Ma *et al.* [23] described an ABM solution to generate personalized real-time data to present route information to travelers. Hager *et al.* [24] modeled the effect of an increasing population on traffic congestion. In [25], a detailed traffic simulator using NetLogo was designed. It analyzes the effect on traffic congestion when various different lanes of traffic are introduced. Jang *et al.* [26] proposed a deep learning and ABM-based solution for traffic light control.

Based on our discussion in this section on the application use cases where ABM is an effective approach, we summarize the pros and cons of the ABM and mathematical model-based approach in Table I. An ABM approach is very effective when the system is highly dynamic and different entities of the system interact with each other. ABM allows the system to respond to any change in environmental conditions and take action, which is not possible with mathematical models where only steady-state response can be captured. Similarly, ABM facilitates modeling of agents with different properties, a feature difficult to model with mathematical modeling. ABM allows access and study of intermediate results of the system, where mathematical modeling focuses on steady-state results. Due to the large system state space, ABM is suitable for systems with a moderate number of agents and limited interactions while mathematical modeling is suitable for the large system with steady-state result focus.

Though the work presented in the above papers and others such as [27], provide motivation to this article, their focus lies in the functionality and optimization of the traffic light systems and not on the modeling and analysis of the communication aspects of the related sensor network.

Due to the suitability of the ABM approach to model large systems composed of autonomous decision-making entities, we believe that ABM is a perfect match to model and analyze the problem addressed in this article, i.e., road traffic management. IoT systems in the future will comprise thousands of devices and decision making will be increasingly more complex. Traditional centralized approaches have little potential to succeed in such large systems, while completely independent decision making will make the system severely suboptimal. ABM allows us to model individual agents and the effect that those agents have on their local environment, and as a result, we can observe the cumulative/system-level behavior that results from the agents interacting with each other and with the environment. The beauty of this approach is that by modeling the interactions and their effects locally, we actually model a complex decision-making system that is decentralized in nature. It may not be optimal as compared to centralized decision-making entities, but it provides a low complexity decision-making framework.

Building on the preliminary work in [28], we have proposed a comprehensive analysis of an ABM approach to model a distributed IoT system. As an example of the IoT system, we apply ABM on the traffic intersection system; but the results can be generalized to other systems with appropriate definitions of agents and their interactions. We summarize the main contributions of this article as follows.

1) We aim to demonstrate a distributed modeling methodology that can be applied to study various MAC protocols in IoT systems. We believe that our framework will help further studies on protocol and routing algorithms for wireless networks with the large number of nodes. We demonstrate the use of ABM to model the communication aspects of IoT networks with suitable definitions of agents [sensors and decision makers (DMs)] and their interactions.
2) Furthermore, we investigate the impact of different MAC protocols on the information reporting error gathered by a sensor network. We first model a single intersection case for developing intuition and then extend the work to more complex multi-intersection case where the impact of system changes in one intersection can be observed over other intersections.
3) Using the ABM approach, we evaluate the spectrum utilization of different MAC protocols in a multilayered network configuration, comprising of a smaller scale (intra-intersection) interaction between nodes using TDMA, slotted Aloha or CSMA/CA protocols, and a larger scale (inter-intersection) interaction in the system using DESYNC and learning MAC (L-MAC) protocols. We quantify the results using numerical evaluation in Section IV. It is worth noting that the goal of the evaluation is not the performance comparison of MAC protocols from an optimization point of view, but to demonstrate a use case for analysis of such problems in communication systems where distributed decision making needs to be performed in a system with large system state.

The remainder of this article is organized as follows. Section II presents the description of the model and outlines the algorithm used for modeling the ABM system. Section III introduces the MAC protocols that are implemented in the *Mesa* framework. In Section IV, we present the methodology used for the analysis and discuss the results gathered from the simulations. In Section V, we elaborate on the main findings and discuss conclusions on the work.

II. Description of Model

A. Agent-Based Modeling Framework

We first define the fundamental terms used in ABM.

Definition 1 (Agent): An autonomous computational object with particular properties and capable of particular actions is called an agent.

Agents are completely autonomous entities in their decision making. As shown in Fig. 1, every agent has a set of attributes and methods that define how and with whom it can interact. This defines the topology of an ABM system. Not all the agents are connected with each other; instead, an agent is connected with a particular set of agents, called *neighbors*, who influence its localized decision making.

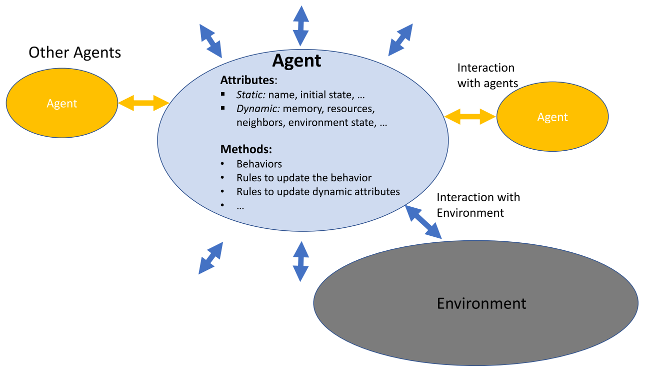

Other Agents
Agent
Agent
Attributes:
Static: name, initial state, …
Dynamic: memory, resources, neighbors, environment state, …
Methods:
Behaviors
Rules to update the behavior
Rules to update dynamic attributes
…
Interaction with agents
Agent
Interaction with Environment
Environment

Fig. 1. Each agent can interact with the environment and/or other agents in its neighborhood. Agents also have a set of static and dynamic attributes storing the properties of the agent and its knowledge about the surrounding agents and the environment. Agents can be heterogeneous having different attributes as illustrated in different colors. The simple rules that an agent is following are encoded in the methods.

Definition 2 (Environment): A set of entities that influence the behavior of an agent constitutes the environment for an agent.

Fig. 1 shows that agents interact not only with other agents but also with the environment.

To model a problem using ABM, we have to define the agents, the environment as well as associated methods and interactions in a way that reflects the original problem. The goal is to model the distributed optimization mechanisms that would converge reasonably to the optimized solution by modeling the interactions of a decentralized system such that the complexity remains manageable.

B. Road Traffic ABM Model

In order to explain the model (i.e., the agents and the environment of the ABM model), we start with the single intersection of roads. The visualization of this model is shown in Fig. 2. The environment of our model is represented by the road. The agents in our model are:

1) sensors;
2) traffic lights;
3) vehicles;
4) DM.

These are the main entities of the system that interact with each other. This single intersection model contains 20 sensors. The sensors are represented by the black dots surrounding the perimeter of the roads. There are four traffic lights. These can be seen as the red/green dots near the center of the image. The yellow squares in Fig. 2 symbolize vehicles traveling on the road.[1] The blue square in the upper right section of the image represents a central DM that will be responsible for managing the timing of the traffic lights in the model.

The interaction between the sensor agents and the environment (the road) is modeled by the collection of traffic measurements. Those measurements represent the number of vehicles approaching the intersection. Once a sensor observes

[1]It should be noted that the vehicles follow U.K. and Ireland driving conventions and travel on the left-hand side of the road.

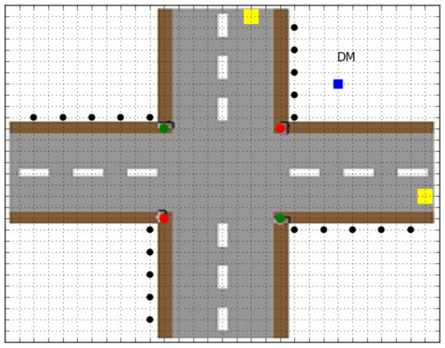


Fig. 2. Sensors, DM, traffic lights, and vehicles represent ABM agents. The sensors are represented by the black dots surrounding the perimeter of the road. The green and red dots are traffic lights. Vehicles are represented by yellow squares, and the DM is represented by a blue square in the upper right section.

Algorithm 1 Model

```
$T_c \leftarrow$ Current Tick number
$T_{max} \leftarrow$ Tick number limit
repeat
    Set tick_number to 0
    for each existing vehicle do
        if no car in front and green light then
            Move forward
    if random_number < probability of new vehicle then
        Create new vehicle
    while $T_c < T_{max}$ or Transmission not succ. do
        Sensor attempt to transmit
        Increment $T_c$
    if $T_c == T_{max}$ then
        DM make decision
```

a vehicle approaching the intersection, it tries to transmit this information to the DM.

The DM controls the timing of the four traffic lights with the aim of optimizing the waiting time of the vehicles traveling through the intersection. It is important to note that we do not focus on the optimization algorithms related to the vehicle traveling time, we rather focus on the analysis of the communication aspects of the sensor network that collects the information about the traffic. In Fig. 2, the traffic lights appear in red and green colors. The red color symbolizes that traffic should stop when it reaches the traffic light. The green color represents that the vehicles are free to move past the traffic light.

The model description is outlined with Algorithm 1. The tick number limit (T_{max}) is a user-defined parameter, that allows us to define the upper time limit for the sensor transmission attempts and at the same time the decision-making interval of the DM agent. One tick is equivalent to a transmission time slot on the MAC layer. Since we dedicate one tick to the movement and generation of the vehicles and one additional tick for the DM decision-making function, the number of ticks that is dedicated for the transmission of the measurements (T_{trans}) is calculated as

$$T_{\text{trans}} = T_{\text{max}} - 2. \quad (1)$$

As shown in Algorithm 1, each car that is currently on the grid will attempt to move one space forward in each simulation iteration (one iteration takes T_{max} ticks). For simplicity sake, the vehicles will always travel in a straight line. If there is already another vehicle in the space a certain car intends to move to or the space in front of that, it will not be allowed to move forward. This prevents vehicles from colliding with each other or traveling too close to each other. Vehicles are also prohibited from moving if they are close to a traffic light in their trajectory and the traffic light is red.

Each simulation iteration involves the creation of new vehicles on the grid. The number of vehicles added to the grid depends on the user-defined parameter (*probability of new vehicle*). Vehicles will only be initialized on the edges of the grid either traveling north, south, east, or west. The initial position of the newly generated vehicles is chosen randomly (i.e., uniformly sampled from a list of all available positions). If there is already a vehicle currently blocking the placement of the newly generated vehicle, the newly generated vehicle will be discarded. Once the vehicles are placed on the grid and the existing vehicles move according to the above-mentioned rules, the sensors collect the vehicle position information and attempt to convey those measurements to the DM. The sensors detect stopped vehicles by remembering the grid space where vehicles were detected in the previous cycle. If this grid space is still occupied by the same vehicle in the current cycle, which means that the vehicle has stopped moving. On the other hand, if the grid space is no longer occupied, the vehicle has moved on. We model the MAC layer protocols (i.e., TDMA, slotted Aloha, and CSMA/CA) for the communication between the sensors and the DMs, and in the case of multiple intersections, we also model the communication between the DMs (TDMA like scheduling). It is important to keep in mind that the implemented model is discrete (the time is divided into slots of equal duration). Each sensor can only transmit at the beginning of a slot. The chosen MAC protocol defines how to deal with potential collisions. A collision happens in case a sensor attempts to transmit in the same slot as one of its neighbors. A neighboring sensor is the one that is in the selected sensor's Moore neighborhood. The Moore neighborhood represents the eight grid spaces surrounding the selected sensor's grid space. Hence, a sensor transmission is affected by the transmission of sensors in all directions including diagonals.

In our analysis, we consider three MAC protocols for the communication between the sensor nodes and the DMs, i.e., slotted Aloha, TDMA, and CSMA/CA. Slotted Aloha deals with collisions by introducing back-off time, meaning that in case two neighboring sensors try to transmit at the same time, both transmissions will be unsuccessful and the sensors will choose a random back-off time to retry the transmission. The Aloha protocol also introduces a timeout time, which in case it is reached without a successful transmission implies that the packet should be discarded. As opposed to the Aloha protocol which does not involve any type of synchronization between the nodes and therefore, potentially leads to packet collisions,

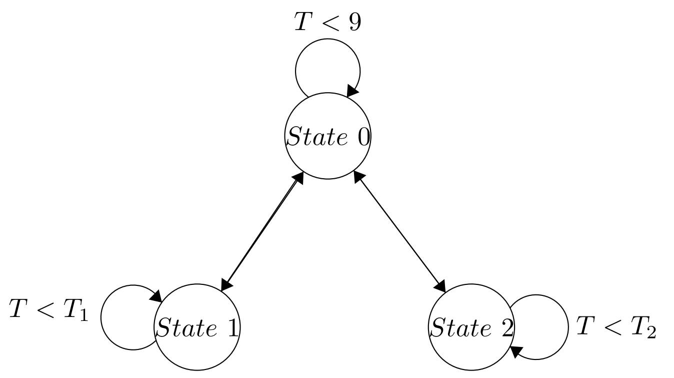


Fig. 3. Traffic light finite-state machine describes the transition of the traffic lights between the three predefined states.

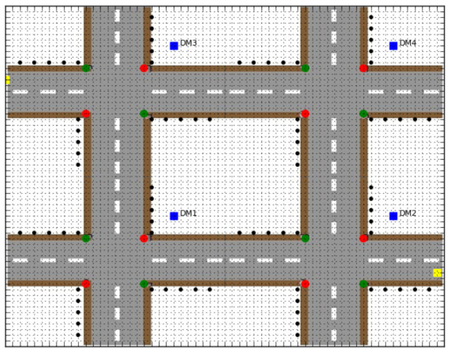


Fig. 4. Four intersections model, showing all the sensors (black), traffic lights (red and green), vehicles (yellow), and DMs (blue) that are part of our model.

the TDMA protocol is implemented by allowing the DM to assign each sensor a specific time slot for packet transmissions. The centralized coordination, results in a collision-free environment, if only one DM exists (i.e., the single intersection of roads scenario). In case multiple DMs are managing the communication of their sensors (i.e., Fig. 4), we have to introduce some type of coordination between the DMs to avoid potential packet collisions. The CSMA/CA protocol, like Aloha, is an opportunistic approach. If a sensor has information to send, it first checks if any of its neighbors is currently using the spectrum resource. If the resource is currently being used, a backoff time will be computed. If the resource is not being used, the packet will be transmitted collision free and the DM will send an acknowledgment packet back to the sensor node to confirm the reception of the packet. It should be noted that the model assumes that there are no hidden nodes. Therefore, all potential collisions will be successfully sensed before transmission.

As shown in Algorithm 1, the final time slot of each cycle is reserved for the DM entity to make a decision. The DM analyzes the information received from the sensors and based on that controls the timing of the traffic lights. The traffic lights follow a strict set of rules that can be summarized with the finite-state machine shown in Fig. 3. The traffic lights can be configured to be in one of three different states—State 0, State 1, and State 2. Fig. 2 shows the system in State 1, allowing cars traveling eastwards and westwards to pass. In State 0, all traffic lights are red, and hence no vehicles are allowed to pass through any traffic lights. State 2 allows only vehicles traveling northbound or southbound to pass. The DM determines how long the traffic lights stay in a certain state, i.e., the DM based on the collected sensor information calculates the values of T_1 and T_2 in Fig. 3. Fig. 3 also shows that between each transition of State 1 and State 2, a period of nine cycles in State 0 takes place. This period allows all traffic that has recently passed through the traffic lights to safely clear the intersection, preventing collisions with vehicles coming from other directions.

As previously mentioned, if we consider a more complex scenario (i.e., the four neighboring intersections scenario), we have to introduce coordination between the DMs. Again, our focus is not the coordination of the decision-making functionalities of the DMs in order to optimize the traffic flow. Therefore, the states of the traffic lights are completely independent from each other. We focus on the optimization of the communication aspects of this scenario, meaning that the DMs coordinate the transmission time slots for their sensors in order to minimize the number of collisions. As shown in Fig. 4, each intersection has 20 sensors, four traffic lights, and one DM. The vehicles can now be generated in more locations compared to the basic model (i.e., one intersection model). We also define a neighbor radius set that defines the distance between two sensors within which their transmissions could result in collisions.

III. Higher MAC Protocols

In this section, we discuss some MAC protocols to model interactions between agents and the environment. In order to coordinate the transmissions for neighboring intersections, we introduce a higher MAC layer that schedules DMs in a TDMA like manner. Each DM gets its own dedicated time slot for communication with its own sensors and such an arrangement models interagent interaction. We also introduce a protocol for communication of higher and lower MAC layers. One slot on the higher MAC layer is the equivalent of 20 slots on the lower MAC layer. The lower MAC layer protocols are described previously (i.e., slotted Aloha, TDMA, and CSMA/CA), whereas the higher MAC layer uses one of the following three approaches to coordinate the communication amongst multiple DMs: 1) centralized DM (CDM); 2) decentralized L-MAC; and 3) DESYNC.

A. Centralized Decision Maker

The approach that involves a CDM entity assumes that this centralized node has all the information needed to control all involved DMs. The centralized node needs to know how many time slots should be assigned to each DM and how to synchronize the activation of all DMs. As mentioned previously, our approach schedules 20 time slots, using a selected protocol—either TDMA, slotted Aloha or CSMA/CA, per DM in a round-robin fashion.

B. Decentralized L-MAC

This approach allows us to coordinate the transmission among multiple DMs by implementing a decentralized TDMA schedule by using the L-MAC protocol outlined in [29]. Each

DM defines a probability vector of length C where C is the available number of time slots in a round. Initially, each DM chooses a transmission slot with equal probability. Based on the success and failure rate of transmission for the chosen transmission slot, the probability vector for each DM gets updated to allow a more intelligent choice of slots in the next round. The result of this is a collision-free schedule, provided that the number of DMs is less than C.

C. DESYNC

The DESYNC algorithm is described in [30]. Each DM initializes a slot for the communication with its sensor nodes. Each DM also listens for messages that are transmitted by other DMs and stores the timestamps of transmissions that occurred before and after its own slot. This information is used by the DMs to adjust their own slot, by computing the midpoint between the previous and next slot. The method described in [30] is concerned with a continuous model. We had to adapt this in order to fit our discrete model. The midpoint (T_m) between the previous (T_p) and next (T_n) time slots is computed as

$$T_m = \left\lfloor \frac{T_p + T_n}{2} \right\rfloor \tag{2}$$

where T_p and T_n are the firing slots of the DMs immediately before and after the DM whose T_m is being calculated, respectively. Equation (2) was further adapted to deal with the periodic nature of our timestamps (i.e., time cycles). For example, let us assume that the round time is $T_r = 10$, $T_p = 8$, and $T_n = 2$. The midpoint slot (T_m) calculated for the next round should be equal to 10. It is to be noted here that T_r is the total round time for each cycle, i.e., the total number of time slots in each cycle. However, using (2), it results in $T_m = 5$. Therefore, if $T_p > T_n$, then the following equation should be used:

$$T_m = \left\lfloor \frac{T_p + T_n + T_r}{2} \right\rfloor. \tag{3}$$

This ensures that each DM will position itself in the midpoint slot between the DMs transmitting before and after it, resulting in a collision-free TDMA schedule.

IV. SIMULATION STUDY

The model was built using the ABM *Python* library *Mesa* [31]. *Mesa* is an open-source framework that is built with the functionality of popular ABM simulation software, such as NetLogo, Repast, and Mason. *Mesa's* data collector module allows us to easily collect data from the agents in the model at specified intervals. *Mesa* enables us to visualize the entire system at each simulation step, helping with the debugging and verification of the traffic lights finite-state machine.

In this section, we present the simulation results for both scenarios: 1) uncoordinated and 2) coordinated. The results are generated for a varying range of input parameters, such as the selection of MAC protocols, and the number of time slots available for the DMs on the higher MAC layer and neighbor radius. The results are evaluated using the information reporting error and the spectrum utilization as criteria.

The reporting error of the information received by the DMs is important to the overall functionality of the system. The actual number of vehicles waiting at a given moment at the traffic lights is denoted by N_W. The number of vehicles that has been registered by the sensor nodes is denoted by N_S, and the number of vehicles that has been reported to the DM is N_{DM}. Since we focus on the communication aspects of the system, we assume perfect sensing implying $N_S = N_W$. We define information reporting error A as the difference between the number of vehicles reported to the DM and the actual number of vehicles waiting at the traffic lights

$$A = N_{\mathrm{DM}} - N_S. \tag{4}$$

A reporting overestimation error ($A > 0$) means that the DM believes that there are more vehicles waiting than the actual figure. A reporting underestimation error ($A < 0$) means that the DM believes that there are fewer vehicles waiting than the actual ones. This data are collected once every cycle before the DM action step outlined in Algorithm 1. Since we assume perfect sensing ($N_S = N_W$), any discrepancies can be attributed to interference within the system, i.e., collisions of packets transmitted from neighboring nodes in the same tick.

The spectrum utilization is a metric that allows us to understand what proportion of the available information in the system is actually transferred to the DM in order to make a more informed decision about the traffic light states. For example, if there are five vehicles waiting, the total amount of information/packets that should be available at the DM is 5. If two sensors successfully utilize the spectrum, the utilization is 40%. Therefore, if the number of successfully transmitted packets in a cycle is denoted with N_{succ} and the actual number of vehicles waiting at the traffic lights is N_W, the spectrum utilization is calculated as

$$U = \frac{N_{\mathrm{succ}}}{N_W} \times 100. \tag{5}$$

A. Uncoordinated Scenario

The uncoordinated scenario assumes that the DMs are not aware of each other's scheduling decision, meaning that increasing the number of neighboring intersections will lead to an increase in the number of collisions, due to the lack of coordination between the neighboring DMs.

Fig. 5 shows the value of the information reporting error averaged over 10^4 simulation steps for the single intersection of roads scenario. The neighbor radius is set to 15, meaning that sensors that are within 15 hops away can potentially interfere with each other.

Fig. 5 highlights that the choice of protocol and level of traffic affect the reporting error of information received by the DM. Regardless of the traffic level, the model using the TDMA protocol makes transmissions with zero error, resulting in highly accurate results. In a single intersection model, each sensor is allocated its own time slot to send. Therefore, there are no collisions of packets in a slot, resulting in highly accurate data transmission to the DM. Thus, the DM is always aware of exactly how many vehicles are currently waiting.

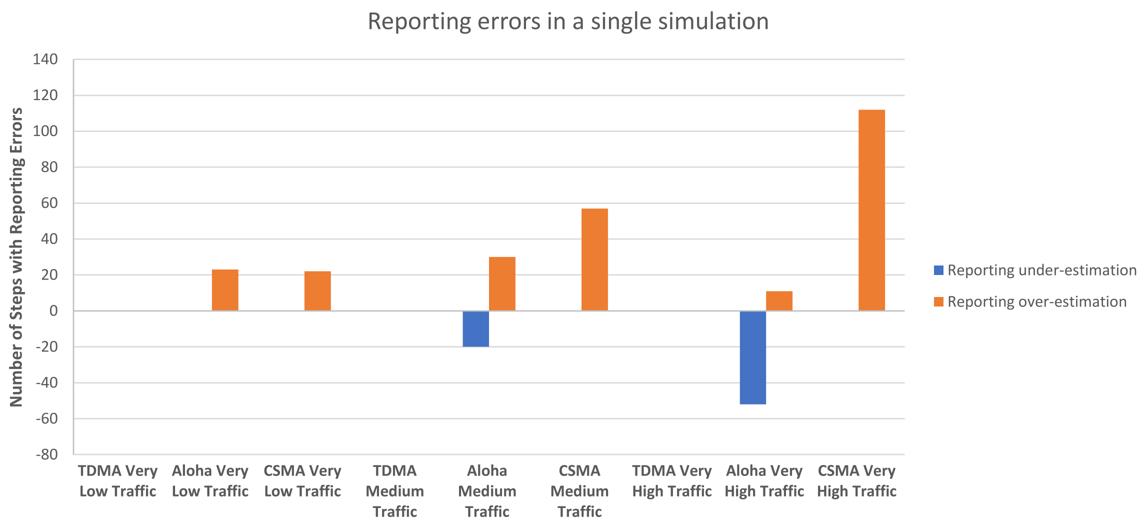


Fig. 5. Information reporting error for the single intersection of roads scenario. The neighbor radius is set to 15.

When there is a low traffic, the model using the slotted Aloha protocol is seen to have a number of reporting over-estimation errors. Such errors are due to backed off sensor packets being transmitted when they no longer reflect the state of the system (i.e., a collision happens, all involved sensors decide to retransmit the packets and in the meantime, the traffic lights change state and the number of vehicles waiting changes). The DM is not aware that the received information is stale and therefore continues to control the timing based on inaccurate information. The information reporting error for the slotted Aloha protocol changes as the number of vehicles waiting grows. This is due to the fact that with the increasing number of waiting vehicles the number of sensors trying to transmit their measurements increases.

The increased number of transmissions results in an increasing number of collisions, leading to the case in which the DM does not have the information about a significant number of vehicles waiting on the lights ($N_{\text{DM}} << N_S$). This is further exemplified through results shown in Fig. 6 on the section of each Aloha simulation when there is at least one car waiting. When there is very little traffic there is very little congestion in the channel and one sensor packet is transmitted at every single slot. This indicates that there is only one car waiting during this time. As the number of cars waiting increases, the amount of collisions increases, leading to less sensors transmitting at every slot.

The model using the CSMA/CA protocol does not suffer from any reporting underestimation error. As expected the number of reporting overestimation errors increases with the increasing level of traffic. The reason for this stems from the increased number of sensors attempting to transmit packets when there is a higher traffic level. When the traffic lights change state, there is a sudden reduction in the number of packets competing for spectrum access as the vehicles begin to move. This leads to an increased amount of packets reaching the DM with inaccurate information ($N_{\text{DM}} >> N_S$).

As previously mentioned, we also calculate the spectrum utilization as shown in (5). The results in Fig. 7 are obtained over 10^4 simulation steps on a two-intersection

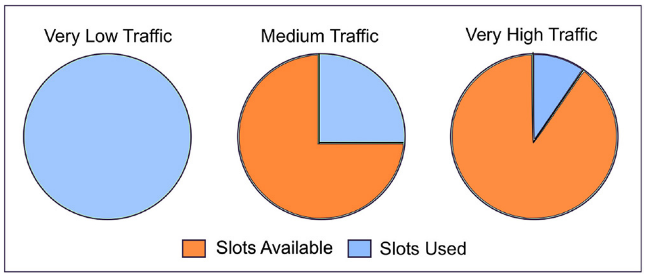


Fig. 6. Section of each Aloha simulation when there is at least one car waiting, representing an uncoordinated scenario where increased number of transmissions causes rise in number of collisions.

model. Considering the uncoordinated nature of the scenario (two DMs that are not aware of each other's scheduling decisions) 40 ticks are assigned for the sensors to transmit the vehicle detection information per cycle. We upscaled the model (from one to two intersections) in order to increase the range of neighboring radii. We vary the neighbor radius from 5 to 25. All the scenarios assume an intermediate traffic level, i.e., a 0.5 probability of a car being generated each cycle.

Fig. 7, as expected, shows that TDMA exhibits the lowest spectrum utilization. However, it is not affected as much by neighbor radius. When the neighbor radius is large, there is a low probability that two sensors from different intersections within the same neighbor radius would be scheduled for the same tick, both having vehicles waiting at them. Hence, the spectrum utilization for the TDMA protocol remains fairly constant regardless of the neighbor radii. CSMA/CA displays the greatest spectrum utilization in all variations of neighbor radii. This is due to the sensing "first—transmitting if available" policy of the CSMA/CA protocol. This allows sensors to avoid collisions of packets by sensing the collision before it occurs and backing off for a random period of time. In comparison to this, slotted Aloha demonstrates a relatively high spectrum utilization when the neighbor radius is low. The increase of the neighbor radius leads to the increasing number

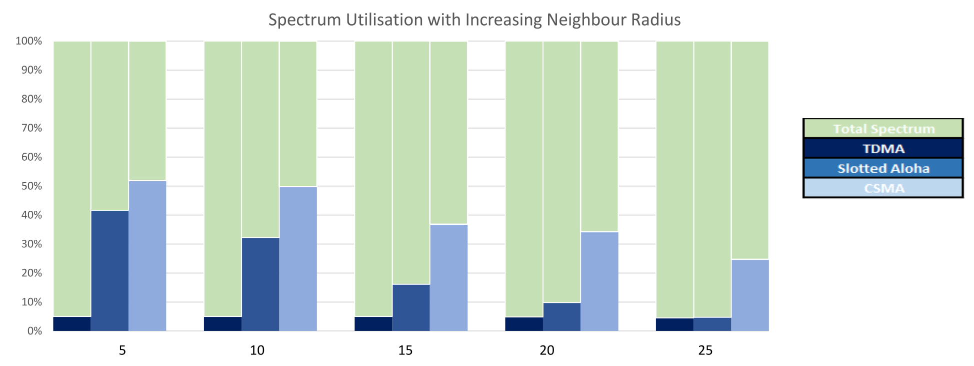


Fig. 7. Spectrum utilization—showing how much of the available information in the system is actually transferred to the DM in order to make a more informed decision about the traffic light system.

of collisions (due to lack of coordination and sensing), which results in lower spectrum utilization.

B. Coordinated Scenario

The coordinated scenario assumes that the DMs are aware of each other's scheduling. As previously explained, in order to coordinate the scheduling mechanisms of the DMs, we introduce a higher MAC layer. The coordination is achieved by either a CDM, the DESYNC, or the L-MAC protocol. The operation of the CDM is obvious—manually configured time slots for each DM. Therefore, we are going to explain in more detail how the DESYNC and L-MAC protocols achieve the best time slot assignment on the higher MAC layer.

1) Desync Algorithm: The DESYNC algorithm relies on the fact that each node in the system performs a task periodically. Depending on the length of the cycle, the convergence of the system can display different behavior. We explain the algorithm with the help of some numerical examples and illustrations in Fig. 8. The ring represents the value of T_r, the total time taken for a full round to be completed. A colored circle with a number in the middle represents the DM that assigned a time slot. An empty circle represents a slot where no DM is assigned and therefore remains idle. We are going to use (2) and (3) to explain the illustrations in Fig. 8. Two different scenarios can arise:

1) number of DMs = T_r [refer to Fig. 8(a)];
2) number of DMs $< T_r$ [refer to Fig. 8(b) and (c)].

Fig. 8(a) is slotted into four time slots that can be chosen by the nodes. Hence, in (2) and (3), $T_r = 4$. Each node is given a unique starting point. Considering that the number of nodes is equal to the number of available time slots, the spacing between the time slots is already maximal and (2) is always satisfied, resulting in no further readjustments of the initially chosen time slots. Therefore, (3) is not applicable in this case.

To summarize the assumptions and conditions for the Desync algorithm, we enlist as follows.

Assumptions:

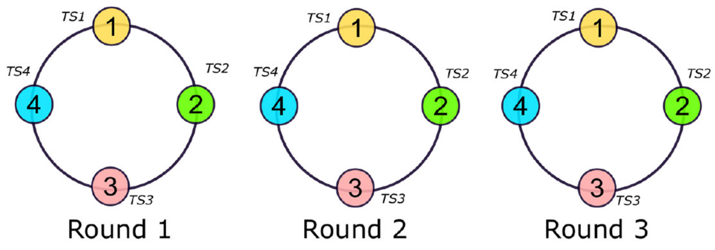
1) Time slots are fixed and the DMs move around.
2) We assume that the DMs are sequentially placed in the first round without any gaps between them.
3) T_p is the firing slot of the previous DM in the present round (since it is a memoryless process).
4) T_n is the firing slot of the next DM in the previous round.
5) *Exception:* In order to calculate T_m for DM1 in the present round, we consider T_p as the firing slot of the DM4 from the previous round.

Conditions:

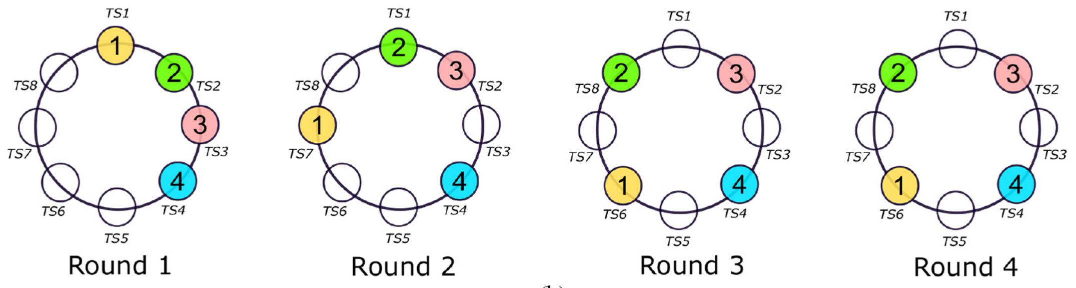
1) If $T_p < T_n$, then use (2) to calculate T_m.
2) If $T_p > T_n$, then use (3) to calculate T_m.
3) If $T_m > T_r$ from (3), then to calculate the right firing slot, we use $T'_m = T_m \bmod T_r$.
4) Convergence is ensured if, the number of DMs = $T_r/2^n$ where $n = 0, 1, 2, 3, \ldots$ is a positive integer.

To illustrate the execution of the Desync algorithm for the case where the number of DMs is less than T_r, we are going to analyze an example with $T_r = 8$. We choose an initial configuration [shown in the first circle in Fig. 8(b): all the nodes representing the DMs start next to each other]. According to (2) and (3), each DM listens to its neighbor's firing and adjusts the chosen slot accordingly. Let us consider DM2 in round 2 as shown in row 2 of Table II. To calculate the position of DM2 in round 2, let us first find out T_m according to (2). In this case, $T_m = 5$, where $T_p = 7$ is the position of DM1 in round 2 and $T_n = 3$ is the position of DM3 in round 1. Now in this case, $T_p > T_n$, hence, we use (3) to calculate $T_m = 9$ which is greater than $T_r = 8$, as shown in Table II. Hence, the final position of DM2 in round 2 is $T'_m = T_m \bmod T_r = 1$. In round 3, for DM2, $T_p = 6$ and $T_n = 2$. Since, $T_p > T_n$, T_m is calculated according to (3) to be equal to 8. Hence, we place DM2 on the eighth slot in round 3. At round 4, for DM2, $T_p = 6$, $T_n = 2$. Again, since $T_p > T_n$, $T_m$is found using (3) as $T_m = 8$. Therefore, DM2 again fires at the eighth time slot for round 4. Following this, for the case where $T_r = 8$ and the number of DMs is 4, the DMs finally converge in the fourth round. This is because the spacing between the DMs is maximal and none of the DMs wants to readjust its firing slot. We summarize the steps and observations of this scenario in Tables II–IV for rounds 2–4, respectively.

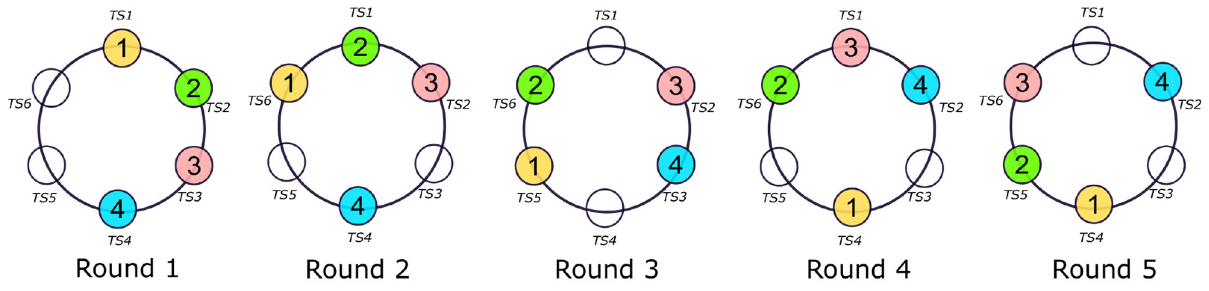
Now, we will look into an example that never converges [Fig. 8(c)]. In this example the chosen $T_r = 6$, and the number of DMs is still 4. Obviously, there is no configuration for which the time slot spacing between the DMs would be equal and maximal, and therefore the system never converges. Fig. 8(c) shows that the nodes continue to rearrange themselves in such a way that a circular pattern emerges. Let us



(a)



(b)



(c)

Fig. 8. Numerical examples implementing the DESYNC algorithm that relies on the fact that each node in the system performs a task periodically at a fixed period. (a) $T_r = 4$. (b) $T_r = 8$. (c) $T_r = 6$.

TABLE II
DESYNC ALGORITHM FOR $T_r = 8$: **ROUND 2**

DM	T_p	T_n	T_m $Eq.(2)$	$T_p > T_n$	T_m $Eq.(3)$	$T_m > T_r$	$T'_m = T_m$ mod 8	Convergence
1	4	2	3	Yes	7	No	Not Applicable	No
2	7	3	5	Yes	9	Yes	1	No
3	1	4	2	No	Not Applicable	No	Not Applicable	No
4	2	7	4	No	Not Applicable	No	Not Applicable	No

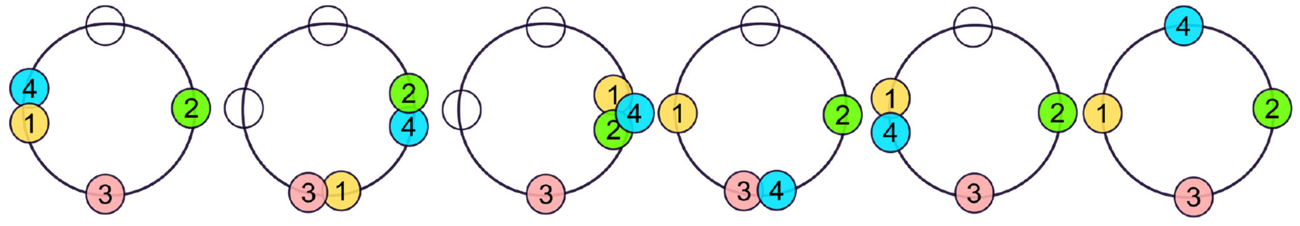
TABLE III
DESYNC ALGORITHM FOR $T_r = 8$: **ROUND 3**

DM	T_p	T_n	T_m $Eq.(2)$	$T_p > T_n$	T_m $Eq.(3)$	$T_m > T_r$	$T'_m = T_m$ mod 8	Convergence
1	4	1	2	Yes	6	No	Not Applicable	No
2	6	2	4	Yes	8	No	Not Applicable	No
3	8	4	6	Yes	10	Yes	2	No
4	2	6	4	No	Not Applicable	No	Not Applicable	No

TABLE IV
DESYNC ALGORITHM FOR $T_r = 8$: **ROUND 4**

DM	T_p	T_n	T_m $Eq.(2)$	$T_p > T_n$	T_m $Eq.(3)$	$T_m > T_r$	$T'_m = T_m$ mod 8	Convergence
1	4	8	6	No	Not Applicable	No	Not Applicable	Yes
2	6	2	4	Yes	8	No	Not Applicable	Yes
3	8	4	6	Yes	10	Yes	2	Yes
4	2	6	4	No	Not Applicable	No	Not Applicable	Yes

consider the firing slots for DM3. In round 1, it starts on slot 3. In round 2, T_m can be calculated according to (2) since $T_p = 1 < T_n = 4$ and $T_m = 2$. If we jump to the fourth round, in case of DM3, $T_p = 6$, $T_n = 3$, T_m can be calculated according to (3), where $T_m = 7$. Since $T_m > T_r$, we have final position as $T'_m = T_m \bmod 6 = 1$. Next in the fifth round, T_m


(a)

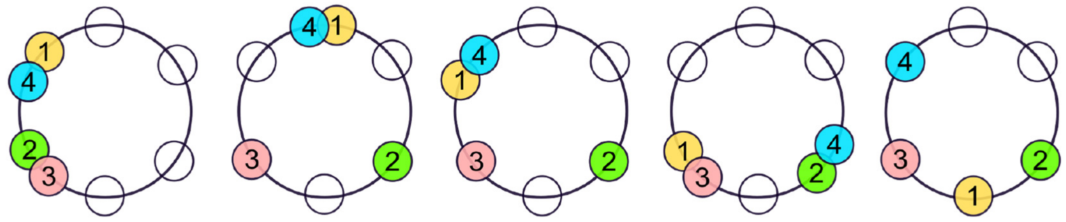

(b)

Fig. 9. Two numerical examples implementing the L-MAC protocol where any node that experience collision continues to rearrange themselves until a collision-free schedule is reached: (a) $C = 4$ and (b) $C = 6$.

for DM3 becomes equal to 6 [from (3)]. The reconfiguration continues in this way and never converges. Theoretically, it is not possible that the nodes can be arranged in such a way that they are equally spaced with maximal difference between them. The reason can be attributed to the fact that, number of DMs $\neq T_r/2^n$ where $n = 0, 1, 2, 3, \ldots$ and the system never converges.

2) L-MAC Protocol: The method used by the L-MAC protocol to implement a TDMA schedule is quite different from the DESYNC protocol. Initially, the DMs choose a random slot in the schedule with equal probability. This is in contrast to the DESYNC protocol where nodes are assigned a starting point. If there is a collision in a slot, the DM will choose a slot again in the next round with updated probabilities. If the DM is successful in a slot, it will choose the same slot again in the next round with a higher probability.

Fig. 9 shows the convergence of the system to a collision-free configuration. Each node that experiences collisions continue to rearrange itself, until a collision-free schedule is reached. In Fig. 9(a), if we take a closer look at DM3, we see that due to a collision-free assignment in a previous slot, the node decides to stick with the chosen slot even after it experiences collisions.

As proven in [29], the system can converge with any round time that is greater than the number of available DMs. For the sake of comparison with the DESYNC protocol, in Fig. 9(b), we show that the L-MAC protocol can converge with a round time of 6. However, the L-MAC protocol does not consider the spacing of the nodes around the ring.

Increasing the round time leads to an increasing number of idle slots, which implies that the probability of a node initially choosing slots without collisions increases as well that results in a shorter convergence time. Though very unlikely, it is still possible that the DMs randomly choose a collision-free schedule on the initial selection with any number of slots in a round greater or equal to the number of DMs. To summarize, the L-MAC protocol compared to the DESYNC can converge with any T_r greater or equal to the number of DMs, whereas the DESYNC protocol requires that the T_r should be chosen such that the number of DMs is a factor of it. On the other hand, the DESYNC protocol will assure maximal spacing between the time slots, whereas the L-MAC protocol does not take into account the spacing.

Similar to the approach adopted to analyze the uncoordinated scenario, we will focus on the analysis of the information reporting error and the spectrum utilization for the coordinated scenario. The coordinated scenario can be implemented by using any of the above-mentioned high layer MAC protocols. The number of time slots available on the higher MAC layer is set to four, meaning that all mentioned higher level MAC protocols would converge to the same arrangement. We used L-MAC in our simulations. The higher lower MAC level time slot length has the ratio of 1:20, meaning that for every higher level MAC time slot allocated to a DM, the equivalent of 20 lower MAC ticks for sensor transmissions is available. Fig. 10 shows the absolute error (absolute value of the information reporting error). The data used in Fig. 10 are obtained from simulations of the four intersections model. The traffic level is set to medium, i.e., a vehicle will be generated with the probability 0.5 in each cycle. The neighbor radius is set to 10. Fig. 10 depicts the spread of the absolute reporting error averaged over ten simulations and over 5000 simulation steps for each simulation.[2] The absolute error is used in this case as it is not our intention to imply a median error close to zero. The TDMA results are not shown in Fig. 10, because TDMA results in an average spread of zero for both uncoordinated and coordinated scenarios. The data in Fig. 10 are shown in the form of a box plot. The upper extreme of the error bars shows the average maximum absolute error, the lower extreme of the error bar shows the minimum average absolute error. The upper lines of the boxes in the graph represent the upper

[2]Averaging over ten already averaged samples is performed to remove the effect of random initialization of each simulation.

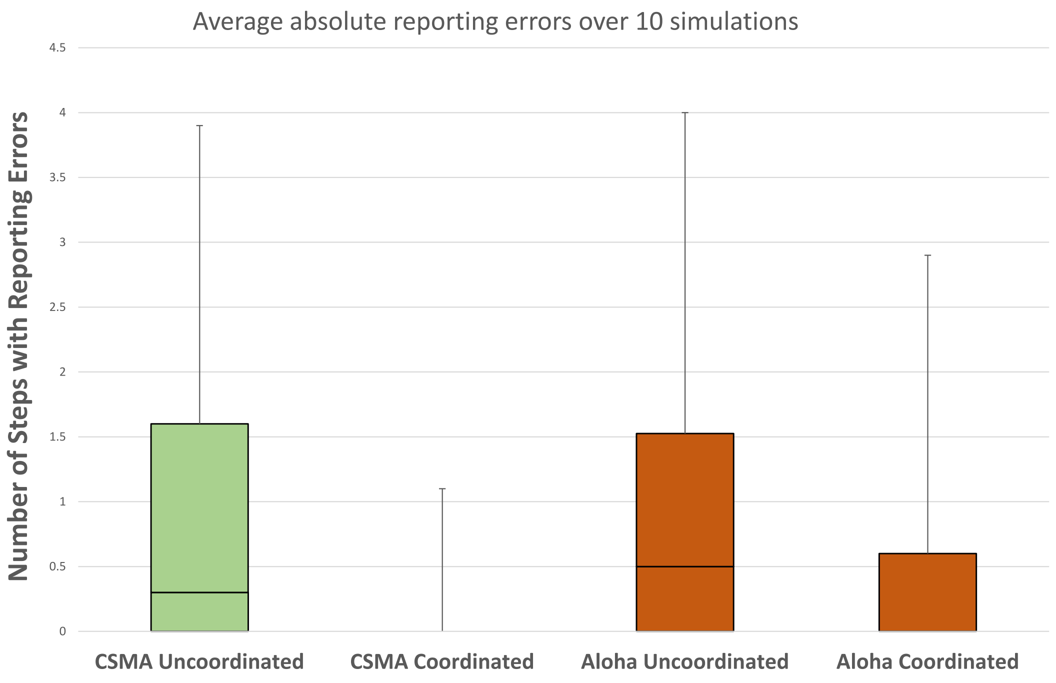


Fig. 10. Average absolute information reporting error—the data are obtained from simulations of the four intersections model; traffic level is set to medium; and the neighbor radius is set to 10.

quartiles, the lower lines represent the lower quartiles. The lines in the centers of each box represent the median values of the absolute error.

Fig. 10 shows that the introduction of the higher MAC layer can greatly reduce the average absolute error for both CSMA/CA and slotted Aloha. The introduction of coordination reduced the average maximum absolute error from 3.9 to 1.1 for CSMA/CA and from 4 to 2.6 for slotted Aloha. The average median absolute error was also reduced from 0.3 to 0 for CSMA/CA and from 0.5 to 0 for slotted Aloha. The reason for the reduction in the absolute error lies in the influence of the higher MAC layer scheduling procedures. Sensors can only transmit data to the target DM when the target DM is selected by the higher MAC layer. This decreases the problem of stale data. Since the ABM approach allows us to analyze each time step of the discrete simulation, the analysis of the log files shows that the majority of stale data is received immediately after a car moving step, mostly affecting the earliest ticks in each cycle. When a higher MAC layer is introduced, stale data primarily affects the first DM that is selected after a movement step. Previously, all four DMs would be affected by this stale information. The stale information will indeed only affect the sensors' packets that are being sent to the first DM selected after a movement step, as the selected DM will reject all other sensor packets being transmitted to other DMs. Because each DM is allocated its own slot, the interference from sensors transmitting to other DMs is decreased, and thus the reporting error of information received by the DMs is improved.

Fig. 11 shows the comparison between the spectrum utilization for the coordinated and uncoordinated scenario. The results are gathered from the simulations of the two intersections model over 10^4 simulation steps. The traffic level is again configured to be medium. This is the same configuration that was used to obtain the spectrum utilization of the uncoordinated model shown in Fig. 7. The L-MAC protocol is used with a round time of two, and a higher to lower ratio of 1:20. Therefore, after convergence these should be no idle slots in the higher MAC layer. This choice of parameters ensures fairness between the uncoordinated and coordinated scenarios. Two higher MAC layer ticks in the coordinated scenario with the ratio of 1:20 results in a total of 40 time slots for the communication between the sensors and the DMs. In the uncoordinated scenario, 40 time slots are assigned for sensor transmissions in each cycle.

The results obtained from these simulations are overlaid with the results obtained from the uncoordinated model as shown in Fig. 11. It should be noted that Fig. 11 is not a stacked bar chart, meaning that the ratio between each scenario and the total spectrum is being analyzed. As shown in Fig. 11, the spectrum utilization is reduced when coordination is introduced. This is due to the limitation that only sensors transmitting to the selected DM are able to send in each slot. The neighbor radius has a similar effect on the slotted ALOHA and CSMA/CA protocols in the uncoordinated and coordinated scenarios (i.e., the spectrum utilization decreases with increasing neighbor radius). Again TDMA is not affected by the neighbor radius. As previously explained, in the coordinated TDMA scenario, each DM is assigned its own higher MAC TDMA slot to transmit where each sensor will then be given its own lower level MAC tick to transmit. The spectrum utilization of TDMA in the coordinated scenario is approximately half the spectrum utilization achieved in the uncoordinated scenario. This is due to the rejection of packets attempting to transmit to DMs that are not selected by the higher MAC layer.

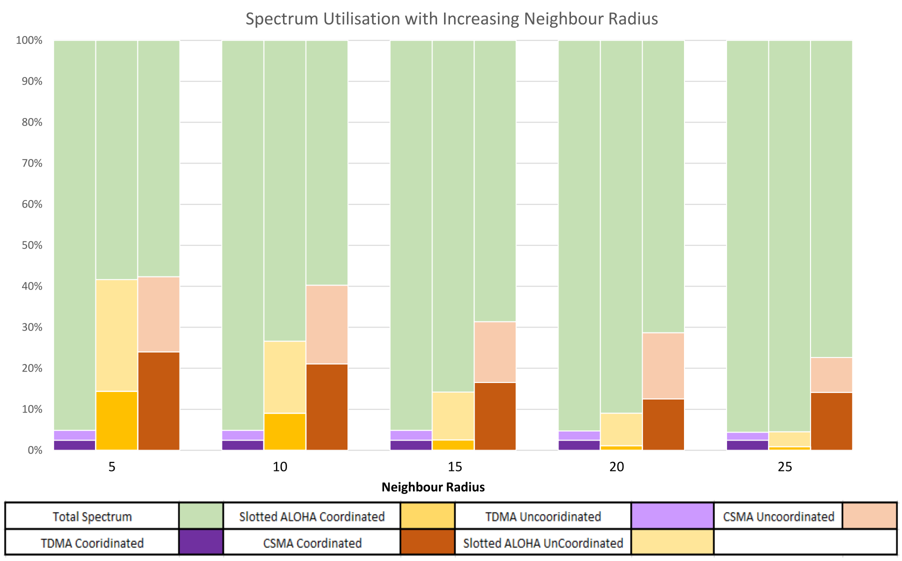


Fig. 11. Spectrum utilization—comparison between the coordinated and uncoordinated scenario, showing how much of the available information in the system is actually transferred to the DM.

V. Discussion and Conclusion

The increasing complexity of the next generation of communication networks leads to a need to change the tools we use to model and analyze them. The primary purpose of this article was to investigate the possibility of using ABM as a method of modeling an IoT network. We showed that ABM is an effective way to model the complex behavior of heterogeneous nodes (e.g., simple sensors, traffic lights, and more powerful decision-making nodes). One of the main advantages of ABM is its flexibility in modeling networks that does not scale up exponentially with the size (e.g., from the simple one intersection model to the more complex two and four intersections model). The complexity of the solution depends on the number of agents (sensors in each intersection) and the intersections in the neighborhood. As the number of neighboring intersections is usually small, complexity scales with the number of agents per intersection. The frequency of message exchange between intersections is another factor influencing the complexity of the computation of the solution. However, it is worth noting that complexity does not really scale with the agents that are not in the neighborhood, therefore, ABM provides a pretty much localized distributed decision-making platform. Besides, it provides an opportunity to add new features for decision coordination (e.g., the higher MAC layer protocol to ensure coordination between the DMs). Human interactions, such as vehicles, are as easily configurable as the network agents in the model. This feature allowed the level of traffic and behavior of the vehicles to be modeled as well as the operation of the network. Another appealing quality of the ABM modeling approach is its ability to model and collect information on a more granular level (i.e., from all agents within the system in any time step of the simulation).

In the models where CSMA/CA was used as the selected MAC protocol, we assumed there are no hidden nodes. If hidden nodes were introduced, the behavior and performance of the models using the CSMA/CA protocol could change. Moreover, we assumed that the only interference in the model is generated from the agents within the IoT network itself. Although the results in Section IV suggested TDMA to be an extremely effective MAC protocol, the limitations of our present model did not highlight the areas where TDMA can fail. For example, if a sensor fails to transmit successfully due to the external source of interference, it has to wait for its slot in the next cycle to transmit. As explained in Section II, the DM uses a method of polling for a certain amount of time before making timing decisions. If this was more of a continuous decision-making process, TDMA could be found to be slower than the other protocols as each sensor must wait for its time slot and for the polling phase to be over, before transmitting. Therefore, more investigations are necessary before concluding in a definite way that TDMA is the best MAC protocol for the kind of application considered in this article and a subject of future research.

The motivation for this article was to investigate ABM as a tool for modeling the complexity of future networks. Building on this article, we envision the future research to be about modeling of complex networks where ABM can help to reduce overhead and complexity of distributed decision making.

References

[1] J. Grady, *System Synthesis: Product and Process Design*. Boca Raton, FL, USA: CRC Press, 2010.

[2] I. Macaluso, H. D. Cornean, N. Marchetti, and L. Doyle, "Complex communication systems achieving interference-free frequency allocation," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2014, pp. 1447–1452.

[3] I. Macaluso, C. Galiotto, N. Marchetti, and L. Doyle, "A complex systems science perspective on wireless networks," *J. Syst. Sci. Complexity*, vol. 29, no. 4, pp. 1034–1056, 2016.

[4] C. M. Macal and M. J. North, "Tutorial on agent-based modeling and simulation," in *Proc. IEEE Conf. Winter Simulat.*, 2005, pp. 2–15.

[5] U. Wilensky and W. Rand, *An Introduction to Agent-Based Modeling: Modeling Natural, Social, and Engineered Complex Systems With NetLogo*. Cambridge, MA, USA: MIT Press, 2015.

[6] C. M. Macal and M. J. North, "Tutorial on agent-based modeling and simulation—Part 2: How to model with agents," in *Proc. IEEE Winter Simulat. Conf.*, Dec. 2006, pp. 73–83.

[7] M. Pogson, R. Smallwood, E. Qwarnstrom, and M. Holcombe, “Formal agent-based modeling of intracellular chemical interactions,” *Biosystems*, vol. 85, no. 1, pp. 37–45, 2006.

[8] I. Benenson, K. Martens, and S. Birfir, “PARKAGENT: An agent-based model of parking in the city,” *Comput. Environ. Urban Syst.*, vol. 32, no. 6, pp. 431–439, 2008.

[9] C. Savaglio, G. Fortino, M. Ganzha, M. Paprzycki, C. Bădică, and M. Ivanović, *Agent-Based Computing in the Internet of Things: A Survey*. Cham, Switzerland: Springer, Oct. 2017, pp. 307–320.

[10] S. Laghari and M. A. Niazi, “Modeling the Internet of Things, self-organizing and other complex adaptive communication networks: A cognitive agent-based computing approach,” *PLoS ONE*, vol. 11, no. 1, 2016, Art. no. e0146760.

[11] P. Twomey and R. Cadman, “Agent-based modeling of customer behavior in the telecoms and media markets,” *Digit. Policy Regul. Governance*, vol. 4, no. 1, pp. 56–63, 2002.

[12] C. C. Douglas, H. Lee, and W. Lee, “A computational agent-based modeling approach for competitive wireless service market,” in *Proc. Int. Conf. Inf. Sci. Appl.*, 2011, p. 6.

[13] A. Tonmukayakul and M. B. Weiss, “An agent-based model for secondary use of radio spectrum,” in *Proc. New Front. Dyn. Spectr. Access Netw. (DySPAN)*, 2005, pp. 467–475.

[14] D. Horváth, V. Gazda, and J. Gazda, “Agent-based modeling of the cooperative spectrum management with insurance in cognitive radio networks,” *EURASIP J. Wireless Commun. Netw.*, vol. 2013, no. 261, p. 261, 2013.

[15] M. Niazi and A. Hussain, “Agent-based tools for modeling and simulation of self-organization in peer-to-peer, ad hoc, and other complex networks,” *IEEE Commun. Mag.*, vol. 47, no. 3, pp. 166–173, Mar. 2009.

[16] M. Liu, Y. Xu, and A.-W. Mohammed, “Decentralized opportunistic spectrum resources access model and algorithm toward cooperative ad-hoc networks,” *PLoS ONE*, vol. 11, no. 1, 2016, Art. no. e0145526.

[17] M. A. Niazi and A. Hussain, “A novel agent-based simulation framework for sensing in complex adaptive environments,” *IEEE Sensors J.*, vol. 11, no. 2, pp. 404–412, Feb. 2011.

[18] M. P. Hernández, B. Alturki, and S. Reiff-Marganiec, “FABIoT: A flexible agent-based simulation model for IoT environments,” in *Proc. IEEE Int. Conf. Internet Things*, Oct. 2018, pp. 66–73.

[19] A. Zappone, M. D. Renzo, and M. Debbah, “Wireless networks design in the era of deep learning: Model-based, AI-based, or both?” *IEEE Trans. Commun.*, vol. 67, pp. 7331–7376, 2019.

[20] S. Coleri, S. Y. Cheung, and P. Varaiya, “Sensor networks for monitoring traffic,” in *Proc. Allerton Conf. Commun. Control Comput.*, 2004, pp. 32–40.

[21] M. Tubaishat, Y. Shang, and H. Shi, “Adaptive traffic light control with wireless sensor networks,” in *Proc. IEEE Consum. Commun. Netw. Conf.*, 2007, pp. 187–191.

[22] S. Y. Cheung, S. C. Ergen, and P. Varaiya, “Traffic surveillance with wireless magnetic sensors,” in *Proc. 12th World Congr. Intell. Transp. Syst. (ITS)*, Nov. 2005, pp. 173–181.

[23] J. Ma, B. L. Smith, and X. Zhou, “Personalized real-time traffic information provision: Agent-based optimization model and solution framework,” *Transp. Res. C Emerg. Technol.*, vol. 64, pp. 164–182, Mar. 2016.

[24] K. Hager, J. Rauh, and W. Rid, “Agent-based modeling of traffic behavior in growing metropolitan areas,” *Transport. Res. Procedia*, vol. 10, pp. 306–315, Sep. 2015.

[25] A. Lansdowne, *Traffic Simulation Using Agent-Based Modelling*. Univ. West England, Bristol, U.K., 2006.

[26] I. Jang, D. Kim, D. Lee, and Y. Son, “An agent-based simulation modeling with deep reinforcement learning for smart traffic signal control,” in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2018, pp. 1028–1030.

[27] K. Kravari and N. Bassiliades, “StoRM: A social agent-based trust model for the Internet of Things adopting microservice architecture,” *Simulat. Model. Pract. Theory*, vol. 94, pp. 286–302, Jul. 2019.

[28] N. J. Kaminski, M. Murphy, and N. Marchetti, “Agent-based modeling of an IoT network,” in *Proc. IEEE Int. Symp. Syst. Eng. (ISSE)*, Oct. 2016, pp. 1–7.

[29] M. Fang, D. Malone, K. R. Duffy, and D. J. Leith, “Decentralised learning MACs for collision-free access in WLANs,” *Wireless Netw.*, vol. 19, no. 1, pp. 83–98, 2013.

[30] J. Degesys, I. Rose, A. Patel, and R. Nagpal, “DESYNC: Self-organizing desynchronization and TDMA on wireless sensor networks,” in *Proc. 6th Int. Conf. Inf. Process. Sensor Netw.*, 2007, pp. 11–20.

[31] D. Masad and J. Kazil, “MESA: An agent-based modeling framework,” in *Proc. 14th Python Sci. Conf.*, 2015, pp. 51–58, doi: 10.25080/Majora-7b98e3ed-009.