People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University Mohamed KHider, BISKRA
Faculty of Exact, Natural, and Life Sciences
Computer Science Department

N° of order:
Series:

A thesis submitted in fulfilment of the requirements
for the degree of Magister in Computer Science

Option:

Data Mining and Multimedia

Theme:

Multi-Channel Communication with Effective Energy Consumption in Wireless Sensor Networks

By:

Ahlem Hannachi

Jury :

Kamal Eddine Melkemi	MCA	President
Yacine Challal	MCA	Examiner
Salim Bitam	MCA	Examiner
Abdelmalik Bachir	MCA	Supervisor

To my Parents,

To my brother and sisters,

To all the family,

To all my friends.

Acknowledgements

All praise and gratefulness is due to Allah. HE the Almighty gave me the strength to complete this work and made me lucky to be supervised by Dr.Abdelmalik Bachir.

I would like to express my deepest gratitude to my supervisor Dr.Abdelmalik Bachir for his guidance and invaluable advice. He with all patience and generosity taught me how to think intellectually and how to contribute to knowledge. His timely and efficient contribution helped me shape this thesis into its final form and I express my sincerest appreciation for his assistance in any way that I may have asked. I consider it my privilege to have accomplished this thesis under his right guidance.

I would also like to thank my committee members, Dr.Kamal Eddine Melkemi, Dr.Yacine Challal, Dr.Salim Bitam for serving as my committee members even at hardship.

I have enormous gratitude to Bouras Samia and Oizen Nabile for the time they spent with me and the patient answers to my questions on their respective indices.

I owe a deep sense of gratitude to my best friends. A special thanks to Abla, Imen, Mochira, Nabila and Sawsen, for their constant moral support and encouragement. They pushed me out through the difficult moments. Alhamdulillah I am blessed to have such lovely friends in my life, who love me for the sake of Allah.

Finally, an honorable mention goes to my family members. I am forever indebted to my mother and father for their unconditional love and endless dua. No words can actually describe their everlasting love to me. I owe a lot to them, they encouraged and helped me at every walk of my life. Their unwavering faith and confidence in my abilities always motivated me.

Abstract

The main three axes that determine the performance of WSN MAC protocols are energy consumption, throughput, and latency, while the major challenge that face WSN is the drain of energy.
Studies prove that energy drain is reduced by increasing the period of sleeping, and that throughput and latency are mainly affected by collisions, in addition to sleep delay. In our contribution, we tried to combine TDMA with FDMA to decrease energy consumption by increasing the sleep periods, and reduce interference by the use of the multi-channel. Our solution is based on the concept of Block Design. We combine the two types of Block Design: Latin Square/Rectangle and BIBD to develop a distributed and dynamic allocation of slots and channels. In order to get a dynamic allocation, we split the Latin Square into Latin Rectangles, where the Latin Square refers to a super-frame and every Latin Rectangle refers to a frame, and a frame is a set of slots. In our receiver-based method, every node generates the Latin Square following a unified formula and runs an Algorithm to determine the number of frames per super-frame, the number of slots per frame, its own slot-channel reservation and that of every neighbor so that we get a distributed allocation of slots and channels.
We used NS-3 simulator to validate the major performance aspects of our method: packet delivery ratio, end-to-end delay, and power consumption.

Keywords: WSN, Multi-channel, Energy consumption, Distributed MAC protocols, Dynamic channel allocation, Block Design.

ملخص

إن أهم ثلاث محاور يتم الاعتماد عليها لتقييم بروتكول الطبقة MAC في شبكات الاستشعار اللاسلكية هي: استهلاك الطاقة، نسبة التأخر و الانتاجية. لكن يعتبر استنفاذ الطاقة هو أهم عائق تواجهه شبكات الاستشعار اللاسلكية.

أثبتت الدراسات العلمية، أنه يمكن توفير الطاقة عن طريق إطفاء جهاز الاستشعار، و ان أكثر عامل يؤثر على التأخر و الانتاجية هو ارتطام الحزم مع بعضها. في هذه المذكرة، حاولنا الجمع بين TDMA مع FDMA و هذا لخفض نسبةاستهلاك الطاقة من خلال، زيادة المدة الزمنية التي تكون فيها وحدة الاستشعار منطفئة، و خفض نسبة ارتطام الحزم عبر استخدام عدة قنوات للارسال، معتمدين على الجبر التركيبي Block Design. لقد قمنا باستخدام نوعين من Block Design المربع / المستطيل الاتيني و BIBD لتوزيع أجزاء الزمن و القنوات بين أجهزة الاستشعار بطريقة متوازية و ديناميكية. في اقتراحنا هذا، قمنا بتقسيم المربع اللاتيني إلى عدة مستطيلات لاتينية، حيث أن المستطيل الاتيني يمثل الاطار الزمني الذي يتكون من أجزاء الوقت. كل جهاز استشعار، يستخدم الخوارزمية لمعرفة أجزاء الوقت و القنوات الخاصة به و بكل جهاز استشعار في مجال تردده. إن جهاز الاستشعار اثناء جزء الوقت الخاص به يكون مستيقظا و في القناة المناسبة (التردد) لاستقبال الحزم.

لقد استخدمنا برنامج المحاكاة NS-3 لإثبات أداء طريقتنا من عدة جوانب: استهلاك الطاقة، نسبة التأخر و الانتاجية.

الكلمات المفتاحية: شبكات الاستشعار اللاسلكية، استهلاك الطاقة، البروتكول الموزع للطبقة MAC ، ديناميكية استخدام القنوات، Block Design.

Abstract in Frensh

Les trois principaux axes qui déterminent la performance des protocoles de la couche MAC dans les réseaux de capteurs sans fils sont: la consommation d'énergie, le débit et la latence, quoique l'inconvénient majeur qui entrave le WSN est la perte (épuisement) d'énergie.
Des études ont prouvé que la perte d'énergie est réduite en augmentant la période de sommeil, et que le débit et la latence sont principalement affectée par les collisions. Dans notre contribution, nous avons essayé de combiner TDMA avec FDMA pour diminuer la consommation d'énergie en augmentant les périodes de sommeil et en réduisant les collisions par l'utilisation du multi-canal. Notre solution est basée sur le concept de Block Design. Nous combinons les deux types de bloc Design: Latin Square/Rectangle et BIBD pour développer une allocation distribuée et dynamique des slots et des canaux. Afin d'obtenir une allocation dynamique, nous avons divisé le Latin Square à Latin Rectangles, où le Latin Square fait référence à une super-trame et chaque Latin Rectangle se réfère à une trame, tandis que une trame est un ensemble de slots. Dans notre base-récepteur méthode, chaque nœud génère le Latin Square suivant une formule unifiée et exécute un algorithme pour déterminer le nombre de slots par trame, le nombre de trame par super-trame, et la réservation de canal et de slot de lui-même et celle de tous ces voisins, afin que nous ayons une allocation distribuée de slots et des canaux.
Nous avons utilisé le simulateur NS-3 pour valider les principaux aspects de la performance de notre méthode qui sont: la consommation d'énergie, le débit et la latence.

Mots-clés: RCSF, multi-canal, la consommation d'énergie, les protocoles distribués de la couche MAC, allocation dynamique des canaux, Block Design.

Contents

List of Figures

List of Tables

General Introduction

Wireless sensor networks (referred to as WSNs)are a specific category of wireless ad hoc networks. A WSN is composed of small sensors (nodes) deployed to monitor physical or environmental conditions such as temperature, sound, pressure, etc. It is used in a large number of applications, among which monitoring of a particular area, health care, air pollution and water quality, environmental and earth sensing, detection of forest fire, detection of landslides, etc. The structure of a sensor node varies according to the phenomena that it monitors, but the main components are: power unit, processing unit, and sensing unit.

Furthermore, every one of those applications has a protocol that is adapted with the problem dealt with. For example in the military application of intrusion detection, the installed protocol has to give a low delay so that the concerned service would react in real time. A MAC protocol such as Q-MAC is well suited under such conditions. In other applications such as the monitoring of air pollution and water quality, latency is not a critical constraint, the main problem in this case is energy consumption.

Therefore, the development of a WSN MAC protocol is strongly related to the problem dealt with, and the developer of a MAC protocol should focus on those factors that affect the concerned problem. Besides the energy drained in transmission and reception, the node consumes an important energy in listening to the channel (node is in busy state listening to an idle channel). In addition, radio interference and problems of hidden and exposed nodes cause the loss of a considerable number of packets.
In the context of wireless communication, the use of multi-channel (FDMA:Frequency Division Multiple Access) has a better performance in terms of reducing the interference (collisions), although it consumes more energy. The introduction of sleep mode saves energy, however, the alternation between sleep and awake states of the nodes has to be known by its neighbors, so that communication can be coherent and the deafness problem will be avoided. There are two approaches to make each node know the alternation (schedule) of their neighbors. The first, makes the node broadcast the schedule of its awake states timing, it causes a large overhead and consumes more energy. The second one, makes each node determine the awake timing of its neighbors by itself and thus consumes less energy.

TDMA(Time Division Multiple Access) ensures fairness among nodes by dividing time into frames each of which is divided into slots, where every node allocates one slot per

frame. The most important advantages of TDMA are the avoidance of collisions and the decreasing of overhearing.
Large number of WSN MAC protocols combine TDMA with FDMA to take advantage of the benefits of each one, such as Y-MAC(2008) [8], MuChMac(2010), and MC-LMAC(2011) [11]. In these protocols, the nodes switch the channels according to the slot, to increase the use of channel, and to improve the throughput.
The problem we provide refers to the portion of energy that will be saved by combining TDMA and FDMA concepts, considering the factors of performance: latency and throughput. Thus, we will not get a very high portion of saved energy, with a very low throughput.

Our objective is to propose a distributed hybrid method at MAC layer to save energy by the use of sleep mode, and to increase throughput by using the multi-channel.
In order to achieve our purpose, we have recourse to Combinatorial Design theory, particularly Block Design; that is an area of mathematics. Block Design is a system of subsets of a finite set which satisfies certain conditions related to the frequency of appearance of pairs of elements of the set in the subsets of the system. Block Design has several types, but among them we were interested in Latin Square/Rectangle, and BIBD.

According to our solution, node exchanges neighbors' lists with its neighbors. After that, node reserves slots and channels to itself, and determines the reserved slots and channels by every neighbor following the same Latin Square, which makes our solution lightweight in terms of message exchange and distributed.
Our proposed solution saves energy, increases the throughput, and completely avoids the deafness problem without a large overhead or overhearing.
To validate those performances, we used NS-3 simulator which has a large community base, and extensively used by the WSN community.

We have organized the structure of the thesis in 4 chapters:
The first chapter presents generalities about WSN, definitions, sensor node components, and performance criterion. In the second chapter we mention some challenges that face the performance of MAC protocols, after that we classify WSN MAC protocols according to time organization. Moreover, the third chapter, we include the development of our method in full details: definitions of Block Design, mapping Block Design to slot-channel distribution, and the presentation of our proposal. In the last chapter, we evaluate by simulation the performance of our method in terms of three factors: packet delivery ratio, end-to-end delay and energy consumption. Finally, we conclude our work by a general conclusion.

Chapter 1

Generalities about WSN

Contents

1.1 Introduction

Wireless sensor networks have a wide range of potential applications to industry, science, transportation, civil infrastructure, and security. They allow the observation of different phenomena with high accuracy in the environment where they are embedded. Nodes in

WSNs process the captured data and communicate it using radio signals.
In this chapter, we care for proffer generalities about wireless sensor networks, starting by definitions, after that we present node components, and conclude by providing the performance criteria of WSN.

1.2 Definitions

Definition 1.1 (Wireless Sensor Networks)**.** [1] A sensor network is an infrastructure comprised of sensing (measuring), computing, and communication elements that gives an administrator the ability to instrument, observe, and react to events and phenomena in a specified environment.

Definition 1.2 (Sensor Node)**.** [12] A sensor node is the basic component of WSN; it is the coalescence of node to one or several sensors. Sensors design is highly dependent on the monitored event. Many different types of sensors such as seismic, low sampling rate magnetic, thermal, visual, infrared, acoustic and radar are able to keep track of the targeted condition such as temperature, humidity, vehicular movement, lightning condition, pressure, soil makeup, noise levels, the presence or absence of certain kinds of objects, mechanical stress levels on attached objects, and the current characteristics such as speed, direction, and size of an object.

Nodes that are in the range of each others are named neighbors (first hop neighbors), while neighbors of a neighbor node are named second hop neighbors.

1.3 Sensor Node Components

According to the environment where sensor nodes are deployed, their architecture changes. Sensor nodes of under water wireless sensor networks have an architecture different from those deployed in a forest environment. However, the main components that are available in every sensor node are: sensing, processing, transceiver, and power units. According to the type of WSN other components will be present, such as the actuator that assets in the sensor node of mobile wireless sensor networks.
Figure 1.1 exhibits the typical architecture of the sensor node.

1.3.1 Sensing Unit

Sensor node can has one or more sensing unit. It is composed of two elements. The first one is the sensor that senses the environment and sends the analog signals to the ADC.

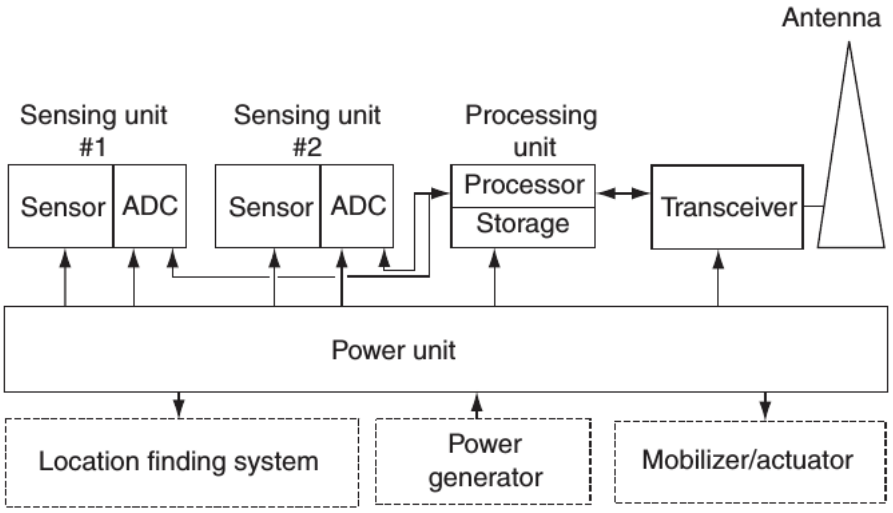


FIGURE 1.1: Typical sensing node [1]

The second one is ADC (Analog to Digital Converter) that is in-charge of converting the analog signals of the sensors to the digital format.

1.3.2 Processing Unit

It manages the correlation of network nodes to carry out the assigned tasks. It is the unit that process the data transfered form the sensing unit, where it runs protocols of communication.
The processing unit is associated with a small storage space. The types of embedded processors that can be used in a sensor node includes Microcontroller, Digital Signal Processor (DSP), Field Programmable Gate Array (FPGA) and Application-Specific Integrated Circuit (ASIC). Among them the Microcontroller has been the most used because of its flexibility to connect to other devices and its cheap price [13].

1.3.3 Transceiver Unit

It connects the node to the network, it is responsible for the wireless communication. It often switches between four states: transmit, receive, idle and sleep.
Wireless transmission media used by transceiver unit includes Radio Frequency (RF), Laser and Infrared, while RF based communication fits to most of WSN applications.

1.3.4 Power Unit

Batteries are the main source of power supply for sensor nodes. Power unit is the most important unit, it supplies the power for the system. Most of the energy is consumed by the transceiver, and the processing units. Thereby, the energy drain is the main concern in WSN. For that reason, developers recourse to energy-harvesting techniques for WSNs.

1.4 WSN Performance Criteria

To evaluate the performance of wireless sensor networks, different performance measures then the traditional ones are required because WSNs differ from the other networks. Among them we cite the following:

1.4.1 Quality of Service

It is strongly related to the service that the WSN provides. In WSN real-time application's, QoS requirements are: delay, loss ratio, and bandwidth.

1.4.2 Fault tolerance

It is the ability of the WSN to exceed the damage of certain nodes. Due to the energy constraint, the interference, or a physical damage, the node will not be effective. It is important that the WSN as a whole tolerates such faults, else a redundant deployment of nodes is the only solution. Thus, the objective of the WSN will be with higher number of nodes than it is needed.

1.4.3 Lifetime

This term can be defined in several ways: (a) the duration of time until some node depletes all its energy; or (b) the duration of time until the QoS of applications cannot be guaranteed; or (c) the duration of time until the network has been disjointed [1].

1.4.4 Scalability

WSN should considers potential expansion by adding new nodes. Thus, it has to maintain its performance even with large number of nodes. Its architectures and protocols must be able scale to large numbers.

1.4.5 Wide range of densities

There are many definitions of network density, but the most used is: the number of nodes deployed per unit area. The distribution of nodes is not homogeneous because, the deployment is casual, and nodes change their position in time in mobile network. Thus, WSN should be able to adapt with different values of density.

1.4.6 Programmability

It is the multitasking; nodes should be programmable, and switch to a new task during their operation time according to the program. Thus, node process tasks besides data.

1.4.7 Maintainability

The WSN is exposed to many internal and external changes. The internal changes could be depleted batteries, or new tasks, etc. The external one are related to the environment where the WSN nodes' are deployed. Thus, the system has to adapt with the different changes, it should be able to retain the system in, or restores it to a specified conditions. According to its status, WSN changes operational parameters or choose different trade-offs (e.g. to provide lower quality when energy resource become scarce). The WSN has to maintain itself, it could also be able to interact with external maintenance mechanisms to ensure its extended operation at a required quality [7].

1.4.8 Reliability

It is the success rate of the wireless sensor network to carry out its objective.
For applications that can tolerate packet loss, reliability can be defined as the ratio of successfully received packets over the total number of packets transmitted [1]. Reliability often comes at a cost in terms of energy consumption. An model for evaluating the cost of increasing reliability for a certain class of MAC protocols can be seen in [14].

1.5 Conclusions

In this chapter, we introduced generalities about wireless sensor network definitions and performance criteria.
It is very important for the developer who works in wireless sensor networks domain

to understand those concepts, as the first step toward the establishment of an effective solution. Because they are considered the gist of this type of network.

Chapter 2

WSN MAC Protocols

Contents

2.1 Introduction

Wireless sensor network is composed of thousands of sensors that communicate with each other where they are deployed in an environment to monitor a physical event. Most of the accommodation of the sensor node are configured at MAC layer to give the best transmission process of the monitoring results in terms of throughput, latency, or delay, etc.

MAC is the layer that controls the reliability and efficiency of a wireless sensor network; it controls the medium access, channel assignment, error control, energy consumption, latency, throughput, etc.

In this chapter we provide an overview about MAC protocols, challenges and attributes. According to time access medium control whether it is random, slotted, frame slotted or hybrid, we exhibit some MAC protocols.

2.2 WSN MAC Protocols Challenges

There are many factors that may degrade the performance of WSN MAC protocols, and among them, we will talk about factors that have the most considerable influence on the performance of MAC protocol.

2.2.1 Energy Efficiency

The most important events that make node drains its energy are:

1. Collisions
   There are many reasons that cause collisions, among them:
   - Hidden Node
     Hidden node problem occurs when node and its second hop neighbor send packet simultaneously to the same node and in the same frequency, it can be avoided by the use of RTS/CTS mechanism. In Fig. 2.1 nodes 1 and 3 are not in the transmission range of each other, when one of them transmits, the other one can not detect the transmission by sensing the channel.
     Fig. 2.1 illustrates the communication that faces the hidden node problem; where nodes 1 and 3 send almost at the same time to node 2, the packets will be collided at the level of node 2 (gray area), whereas nodes 1 and 3 complete their transmission because collision is not known by them. Hidden node problems are usually associated with exposed node problems which occur when a

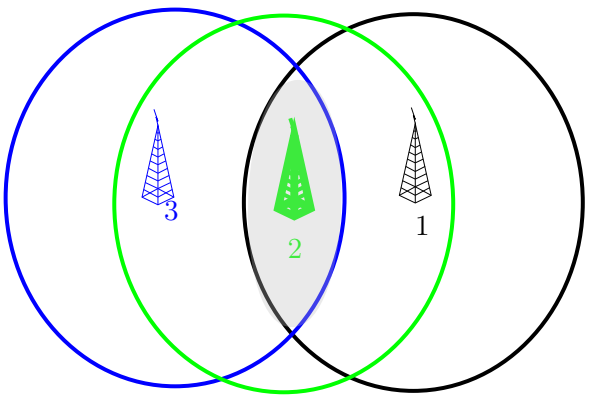


FIGURE 2.1: Illustration of hidden node problem.

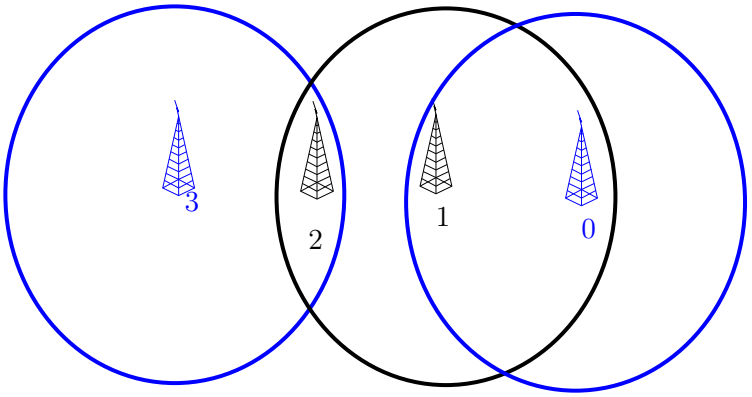


FIGURE 2.2: Illustration of exposed node problem.

node postpones its transmission, because of ongoing another transmission, despite the fact that:

- It is not concerned (it is not the intended receiver of that current transmission).
- The interference between current and future transmission will not lead to the collision, and packets of both transmission will be received correctly; the intended receiver of future transmission is not in the range of the current one.

Fig. 2.2 illustrates node communication that face the exposed node problem, where node 1 is sending to node 0. When node 2 senses the channel to send to node 3, it will find the channel busy. Node 2 holds on believing that its transmission will collide with the current one, while it will not because node 3 is not in the range of node 1.

- Deafness

  Deafness problem takes place in receiver-based protocols, where node and its first hop neighbor receive in the same time. Nodes a and b are neighbors which have the same receiving time. When node a send to node b (or the inverse), packets of nodes that were transmitting to a will be dropped because node a is transmitting and could not be heard.

2. Overhearing

   Overhearing is related on receiving irrelevant packets. In some cases, node receives packet which is not concerned, or receives duplicate packets.

   Overhearing problem can be reduced by adopting a receiver-based protocol, thus node will receive only during a short period.

3. Control Packet Overhead
   Packet overhead problem occurs when control information take a considerable portion from the size of the packet or consume a considerable energy. Control information could be about synchronization,or routing, etc.
   Overhead problem can be reduced by avoinding RTS/CTS mechanism that generates large overhead, although its use avoids collisions. Or by using distributed protocol.

4. Idle listening
   Idle listening is state of node in which keeps listening to the channel without sending or receiving.
   It can be alleviated by the use of receiver-based protocols and scheduling the wake up time of nodes, thus node will listen to the channel only when it is awake.

2.2.2 Latency

Latency is the time that a packet takes from the sender to the receiver. It is negligibly affected by the physical distance but mostly determined by the delay caused by the MAC protocol to organize the access to the shared medium. Some MAC protocols force the node to backoff its transmission to avoid potential collisions.

2.2.3 Throughput

Throughput is the average rate of successful message delivery over a communication channel. The throughput is usually measured in bits per second (bit/s or bps), and sometimes in data packets per second or data packets per time slot.

2.3 WSN MAC Protocol Classification

Authors used different criteria to classify WSN MAC protocols, such as the problem dealt with (see [11]), synchronization (see [5]), and multichannel (see [15]).
We followed MAC protocol classification that is based on time allocation, which means how time is distributed between nodes. There are four types: random, slotted, frame slotted and hybrid time allocation, as it is mentioned in Figure 2.3.

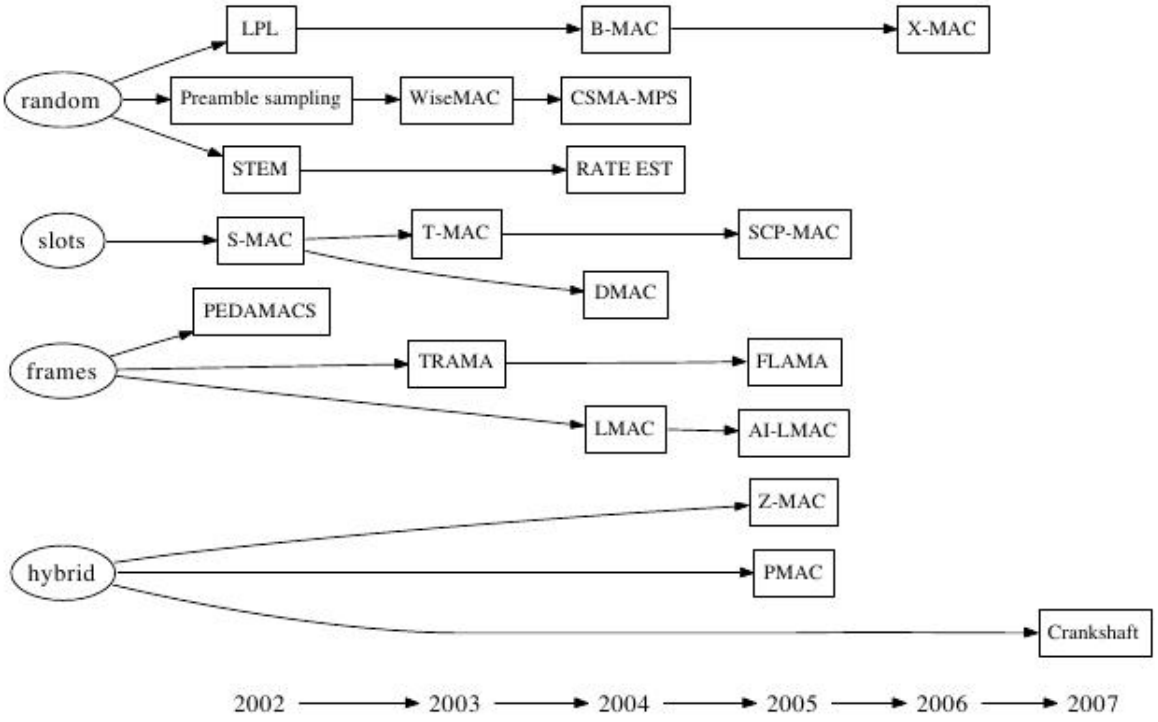


FIGURE 2.3: Taxonomy of MAC protocols according to time organization and historic development [2].

2.3.1 Random Time Allocation

In Random time allocation protocols, nodes choose their own schedule autonomously. This type of protocols does not require synchronization, thus it will saves the energy drained caused from the overhead of synchronization. Random protocols such as STEM, X-MAC, RC-MAC, SCP-MAC, B-MAC and WiseMAC are based on one of the following techniques: Preamble Sampling or Low Power Listening.

Definition 2.1 (Clear Channel Assessment (CCA))**.** Is a fundamental mechanism in MAC protocols. CCA determines whether the wireless channel is idle or busy by sampling the energy level in the channel and comparing it with predetermined threshold (the noise floor). If the sample is higher than the threshold, then announce the channel busy else it is idle.

CCA is most used in two important strategies. First, CSMA/CA protocols that have been used to avoid collisions on shared wireless channels, by sampling the channel activity just before the transmission. Second, CCA has been used in Low Power Listening (LPL)[16].

Definition 2.2 (Preamble Sampling (El-Hoiydi 2002))**.** Is a technique that stands on alert the receiver node before starting current data transmission by preceding it with a preamble. The size of the preamble is the same as sampling period.

In preamble sampling, node is scheduled between sleep and awake states to decrease

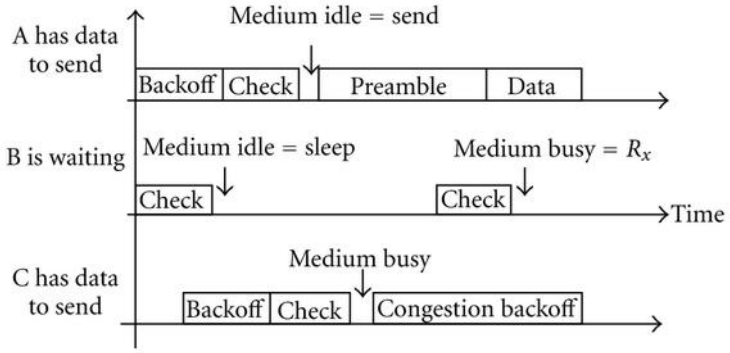


FIGURE 2.4: B-MAC communication example. All nodes are within range of each other. [3].

energy consumption because when traffic is light the most of energy is wasted in idle listening. When the node samples the channel, if the channel is busy, it will keep listening until it receives a packet or the channel becomes idle. Preamble Sampling is also called Low Power Listening in the literature. Similarly, in LPL, every node wakes up periodically to perform CCA, and if the channel is idle, node returns to sleep state otherwise it keeps waiting to receive the packet.

1. B-MAC (Berkeley MAC)

   B-MAC uses Low Power Listening . Node in B-MAC waits a back-off time before starting the transmission and sample the channel. If the sample is smaller than the noise floor (this means that the node has detected outliers), it will declare the channel as clear (because efficient transmission will never has such signal strength) and it will start the transmission (see Fig. 2.4).

   In B-MAC, each node estimates the noise floor by measuring the signal strength when the channel is supposed to be clear (e.g., immediately after transmitting a frame) an enqueue results measurement into FIFO queue and takes its median [5]. B-MAC is efficient in collision avoidance but energy is drained because of the overhead created by the preambles.

2. Wise-MAC (Wireless Sensor)

   WiseMAC is the first protocol that refines preamble sampling.
   In WiseMAC each node interchanges its wake up time with neighbors to construct its internal dynamic table. Before starting transmission, node estimates clock drift to add it to the wake up time.
   After a successful reception node piggyback its wake up schedule in the acknowledgment. If node doesn't know about wake time of the receiver, it has to send a full length preamble. Every node will sample the channel periodically during check interval to check whether it is intended for next transmission (see figure 2.5).

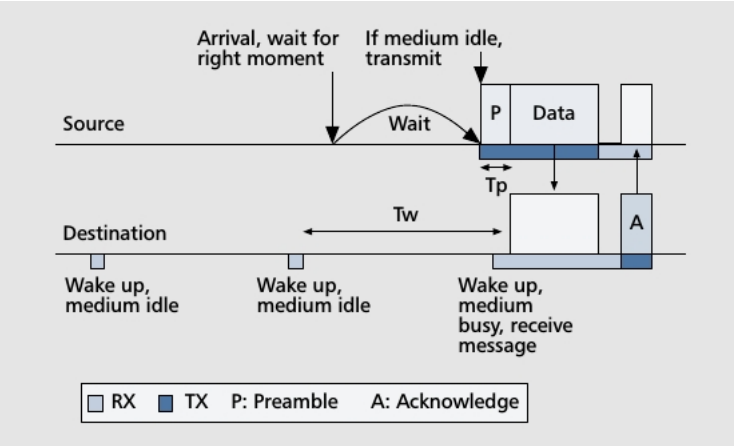


FIGURE 2.5: The WiseMAC Concept [4].

WiseMAC implements energy efficiency by scheduling sleep/awake periods. While it drains energy in broadcasting because preamble will be largely long to cover the wake up time of all receivers.

WiseMAC does not implement RTS/CTS but it can be added in the higher layer, besides, it suffers from hidden node problem.

2.3.2 Slotted Time Allocation

In slotted time allocation protocols, nodes need to be synchronized, thus they generate an overhead that consumes more energy. Most of the slotted protocols such as S-MAC, T-MAC, R-MAC, DW-MAC, and Q-MAC are based on Adaptive Listening technique.

1. S-MAC (Sensor MAC)

   S-MAC uses a fixed duty cycle. Nodes are organized in virtual clusters, where each cluster has a separated schedule that contains three periods: SYNC, DATA and SLEEP. Nodes of the same cluster are locally synchronized and have the same schedule. They wake up every SYNC period to synchronize with each other and adjust their clock if there is a drift.

   When a node starts up, it keeps listening to the channel during an initial phase period to receive SYNC packet, thus it can join the cluster associated to the packet. If the initial period elapses and node did not receive any SYNC packet, it starts constructing new virtual cluster by broadcasting SYNC packets, so other nodes can join it later.

   Node can join more than one virtual cluster (see Figure 2.6), it depends on the number of different received SYNC packets. The node should run schedules of every virtual cluster in which it is a member leading it to waist its energy.

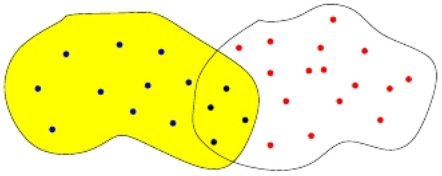


FIGURE 2.6: Overlapping virtual clusters with bridging nodes running both schedules [2].

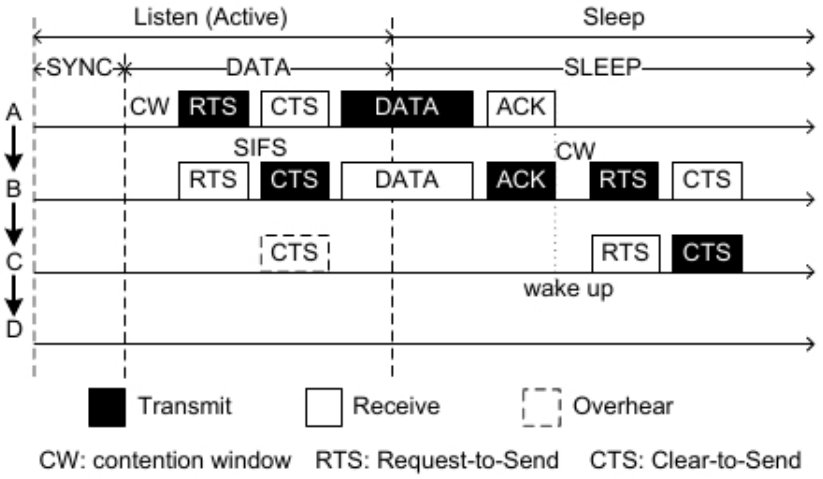


FIGURE 2.7: The adaptive listening in S-MAC [5].

After synchronization and during DATA period, if node has to transmit, then it has to contend to access the medium by using RTS/CTS mechanism. In sleep period, if node is not concerned in the current transmission, it switches off its transmitter, otherwise it stays active till the end of the transmission and the acknowledgment. In every cycle, packet can forward only one hop. The principle of the establishment of virtual clusters has also been used in [17] to create virtual structures for efficient broadcasting in WSN.

2. S-MAC with Adaptive Listening

   S-MAC protocol has been optimized by inducing adaptive listening where packet can forward two hops during one cycle as it is exhibited in Figure 2.7.

   In S-MAC with adaptive listening, a node can hear only its first hop neighbor. During DATA period, if node sends CTS, all of its immediate neighbors that are not involved in current transmission will overhear it, and will schedule to wake up in SLEEP period after the transmission is finished. So, if the node is the next hop of packet, it will forward it, else it will return to sleep again.

   S-MAC is not efficient in low traffic load, because even if node is not sending or receiving any packet. Node will stay active during all of DATA period, witch

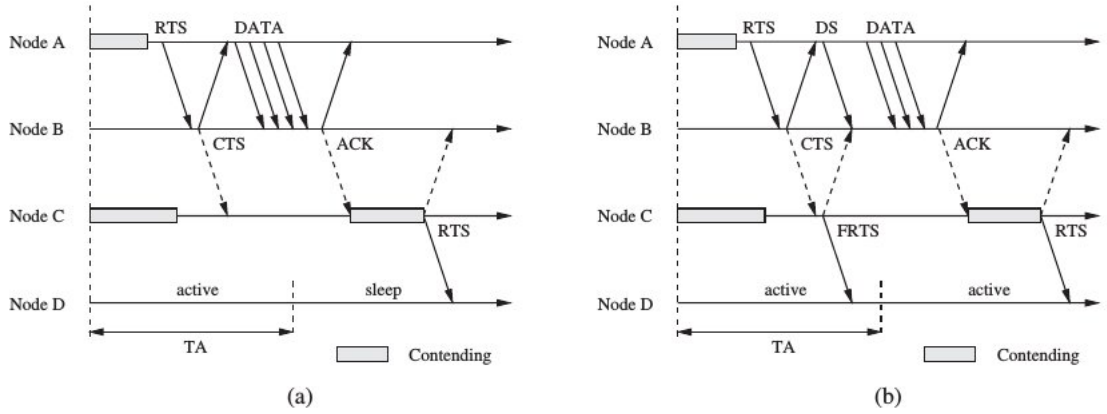


FIGURE 2.8: Data exchange in T-MAC, showing the early sleeping problem in (a) and the future-request-to-send technique in (b) [6].

create idle listening that drain energy. Even adaptive listening is induced in S-MAC, idle listening, latency and energy consumption are increased in low traffic load networks.

3. T-MAC (Timeout MAC)

   Fixed duty cycle makes S-MAC not adaptive to different traffic load. Besides, the interval of duty cycle is constrained. T-MAC protocol mitigates effect of those problems by introducing adaptive active period and decreases the interval of DATA period from 300 ms in S-MAC to 15 ms [2] to make node return to sleep as quickly as possible.

   When transmission operation ends, participated nodes keep listening to the channel for a short timeout waiting for new transmission. The timeout should be long en aught to allows neighbors to send and receive CTS.

   The Figure 2.8(a) shows an example of data transmission using T-MAC, where TA represents the DATA period. Nodes A and C compete to access the medium. The node A wins and start sending the data to node B, but node C keeps listening to the channel waiting to overhear the ACK of its immediate neighbor. When node A receives ACK, node C returns to compete on the access to the channel.
   The node C wins but the future receiver (node D) will not receive RTS of node C, because node D is in SLEEP period.
   So to avoid such problem, T-MAC introduces FRTS (Future Request To Send). When node looses the access to medium, then it sends a FRTS to inform receiver to stay awake and to not switch to sleep state, as it exhibited in Figure 2.8(b).

2.3.3 Frame Slotted Time Allocation

In this type of protocols, nodes are globally synchronized. TRAMA, ZMAC, TreMAC, Crankshaft are frame slotted MAC protocols that are based on TDMA.

1. TRAMA(Traffic-Adaptive Medium Access)
   The TRAMA protocol itself composed of 3 protocols: Neighborhood, Schedule Exchange and the Adaptive Election algorithm. TRAMA assumes that nodes are time synchronized.
   Time is divided into periodic random access intervals (signaling slots), and scheduled access intervals (transmission slots). While cycle is signaling slot followed by a transmission slot.
   During random access intervals, nodes use Neighbor Protocol to discover their first and second hop neighbors. Node chooses randomly a slot, and send without any carrier sensing a short control packet to gather information about neighborhood. The control packet maintains a set of the new and the deleted neighbors. In case there are no changes, this packet serves as "keep-alive" beacon [6].
   Relying on the gathered information and for every slot of scheduled access intervals, node runs distributed scheduling algorithm to determine from nodes those that will send, those that will receive and those that can switch to sleep mode in the current slot. Node that has the highest priority can transmit, and in case it does not need to send, then it announces that with all determined parameter by piggybacking them in packet called schedule packet [7].
   During scheduled access intervals, nodes use Schedule Exchange Protocol (SEP) to interchange schedule packet. From the received schedules, if in slot s, node x is future receiver of y and y have the highest priority within 2 hops neighbors of x, node x will stay awake during slot s to receive packets of y, otherwise it will return to sleep.
   In complicated cases, such as the example in Figure 2.9. Node D has the highest priority in 2 hops neighbors of B, and node A has the highest in 2 hops neighbors of C. Adaptive Election Algorithm is used to overcome such situation and also to manipulate unused slots of neighbors (see [7]).

2. Crankshaft
   Crankshaft is receiver-based protocol that combine TDMA with CSMA.
   In Crankshaft time is divided into frames and frame to slots, and there are two types of slots. Each frame starts by slots that are dedicated to unicast traffic, followed by slotted dedicated to broadcast traffic.
   Because Crankshaft is receiver-based protocol, each node has its specified slot to

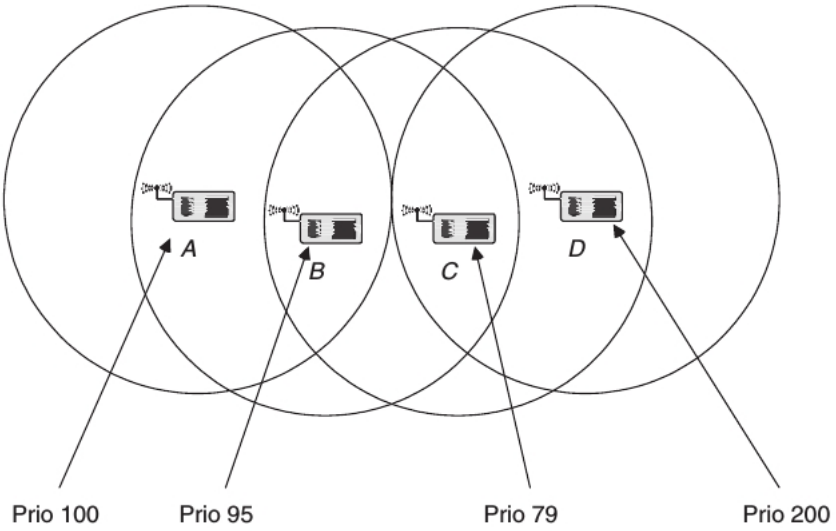


FIGURE 2.9: TRAMA: conflict situation [7].

receive data and this slot is determined by calculating the modulo of MAC address on n, where n is the number of unicast slots. It is not forbidden that two nodes share the same slot.

Crankshaft is similar to WiseMAC in the method they used to access the medium [11]. To mitigate contention, nodes that loose the access in current frame try with higher probability of 70% portion in the next frame.
Slot interval should be large enough to hold contention, transmission and acknowledgment.
Although, Crankshaft decreases overhearing that consume energy, it drains energy during broadcast slots of the channel sampling.

2.3.4 Hybrid Time Allocation

Hybrid protocols combine TDMA with FDMA, their main issue is channel allocation and cross-channel communication. Time allocation for some hybrid protocols is by assuming that time is synchronized, and divided into fixed length beacon intervals, such as MMSN, TMCP, and Y-MAC.

1. MMSN
   Multi-frequency Media access control for wireless Sensor Networks (MMSN) is receiver-based protocol that does not use RTS/CTS mechanism to reduce the overhead generated by this mechanism. In MMSN node uses exclusive frequency assignment to allocate one frequency among four frequencies (see [18]), to avoid having the same frequency as its second hop neighbor. The node that has the smallest Id

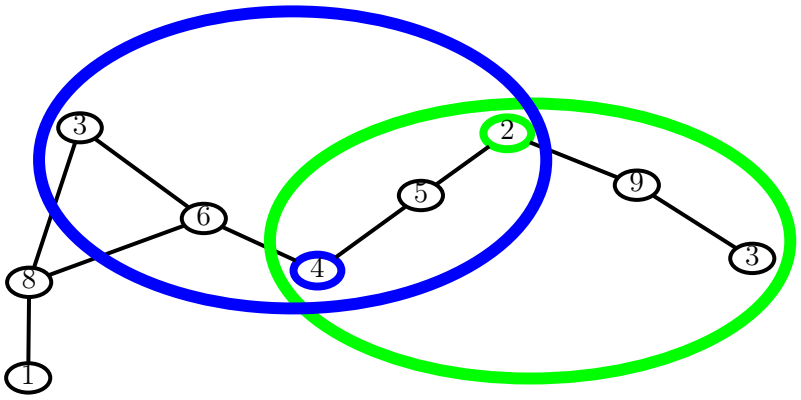


FIGURE 2.10: Channel allocation in MMSN MAC protocol.

within its first and second hop neighbors allocates the smallest available frequency, then beacons the chosen frequency in two communication hops. Nodes that do not have the smallest Id in its two communication hops, wait for the decision of neighbors that have an Id smaller than its Id to select the smallest free frequency. In Fig. 2.10, node 2 has the smallest Id withing its two communication hops (in green oval). We suppose that the smallest available frequency is f_0, thus node 2 broadcast in two communication hops that it allocates frequency f_0, while node 4 have to wait for the frequency allocation of node 2 and 3 to decides its frequency and broadcast its decision in two communication hops (the blue oval).

In some cases, where there is not enough of frequencies, node find that every frequency is allocated by at least one of its second hop neighbors. Thus, node select the lesser used frequency and broadcast its frequency.

2. Y-MAC

   Y-MAC concept can be used as receiver-based or sender-based protocol, but in generally it is a receiver-based protocol, because this one gives better performance in terms of energy consumption. Y-MAC node use low power listening to access medium during its slot.

   Time is divided to frames and frame is composed of two periods: broadcast and unicast. The unicast period is divided to slots, while the broadcast period is not divided. Node wakes up during its slot, and all nodes wake up at the beginning of the broadcast period to wait for broadcast packets, and if there is not, every node returns to sleep till its wake up slot.

   Y-MAC uses its own method of synchronization that based on exchanging the remaining time of the current super frame, thus it reduces overhead generated from synchronization.

   Node in Y-MAC maintains a vector of slot allocation of its first hop neighbor including itself, where the byte at position i refers to the slot i.

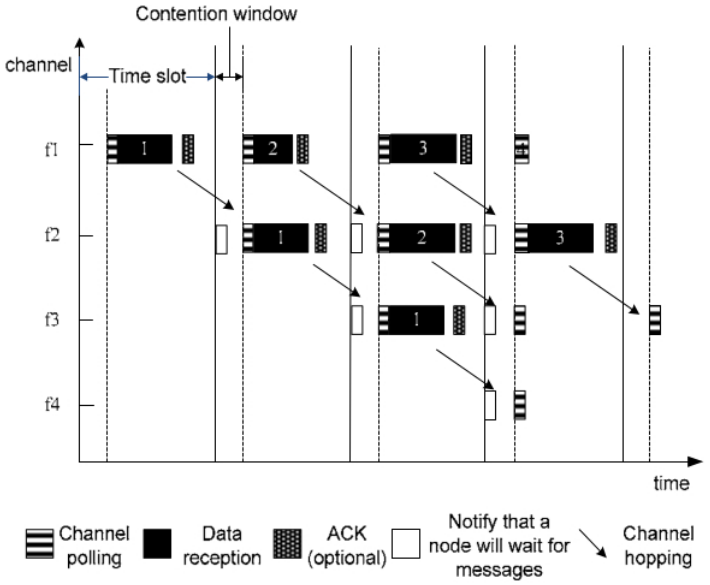


FIGURE 2.11: A light weight-channel hopping mechanism [8].

Acknowledgment can be enabled or disabled. Every node at the beginning of its slot switches to the base channel and also all potential senders too.
The mechanism of channel hopping in Y-MAC is illustrated in Fig. 2.11. After that node receives a unicast message in the base channel ($f1$), it determines the channel of communication by using the hopping sequence generation algorithm (see [8]).

At the start of the slot, the receiving node sends an independent packet, so node can know whether it wins in the contention to send or not. Thus, notifying loser node in the contention by overhearing the packet whether the receiving node will be waiting during the next time slot or not.

2.4 Conclusions

In this chapter, we presented the main constraints faced in the conception of MAC protocol of wireless sensor network. We also exhibited the reasons that led the drawbacks and how to mitigate its effect. Then we classified some MAC protocols according to time allocation and explained its concept of communication as well.

Chapter 3

Block Design Slot-Channel Distribution for MAC Protocols

Contents

3.1 Introduction

Block design is a special type of combinatorial designs. Its concept is to gather randomly elements of a set in subsets considering different parameters (number of subsets, number

of elements per subset). It is used in large number of applications such as experimental design, cryptography [19], finite geometry, software testing, and algebraic geometry [20].

Block design has many types, however we are mainly interested in Latin Square Design to develop a pattern in the reservation of the slot and the channel.

In the core of multichannel WSN communication, many studies used Latin Square such as [21], [22], [23] and [24]. Each one has a different purpose and exploits properties of Latin Square in a special manner.
In the course of this chapter, we start by sitting definitions of some types of Block Design that are basically introduced, followed by the mapping to slot-channel distribution, and propose at the end a pattern of distributed allocation method of slots in multi channel (including one channel) networks.

3.2 Block Design

3.2.1 Pair Design

Definition 3.1. [25] For positive integers $t \leq k \leq v$ and λ, a $t-(v,k,\lambda)$ design is a pair (X,B), satisfying the following properties:

1. X is a set of v elements, called *points*,
2. B is a family of k-subsets of X, called *blocks*,
3. every t-subset of X is contained in exactly λ blocks. By convention $b = |B|$ denotes the number of blocks.

Definition 3.2. [20] An (r,λ)-design is a pair (V,B) where V is a set of v elements and B is a collection of b subsets (blocks) of V such that every distinct pair of elements occurs in precisely λ blocks and every element occurs in precisely r blocks.
Let $n = r$ - λ. Let k_i be the size of the ith block in the (r,λ)-design. We have the following definition:

If $k \leq v$, then the design is called: Incomplete Design,
If λ is constant, then the design is called: Balanced Design,
If $b = v$, then the design is called: Symmetric Design.

3.2.2 Balanced Incomplete Block Design (BIBD)

Definition 3.3. [19] Let v, k and λ be positive integers such that $v > k \geq 2$. A (v,k,λ)-Balanced Incomplete Block Design (which we abbreviate by (v,k)-BIBD) is a design (X,A) such that the following properties are satisfied:

1. $|X| = v$,
2. Each block contains exactly k points.
3. Every pair of distinct points is contained in exactly λ blocks.

Proposition 3.4. [20]
The trivial necessary conditions for the existence of a BIBD(v, b, r, k, λ) are:

$$vr = bk \tag{3.1}$$

$$r(k-1) = \lambda(v-1) \tag{3.2}$$

Parameter sets that satisfy 3.1 and 3.2 are admissible.

3.2.3 Splitting Designs (SD)

Definition 3.5. For positive integers t, v, b, c, u, λ with $t \leq u$ and $cu \leq v$, a $t-(v, b, l = cu, \lambda)$ Splitting Design D is a pair (X, B), satisfying the following properties:

1. X is a set of v elements, called points,
2. B is a family of l-subsets of X, called blocks, suck that every block $B_i \in B(1 \leq i \leq |B| = b)$ is expressed as a disjoint union

   $$B_i = B_{i,1} \cup \ldots \cup B_{i,u}$$

   with $|B_{i,1}| = \ldots = |B_{i,u}| = c$ and $|B_i| = l = cu$,
3. every t-subset $\{x_m\}_{m=1}^{t}$ of X is contained in exactly λ blocks

   $$B_i = B_{i,1} \cup \ldots \cup B_{i,u}$$
   such that
   $x_m \in B_{i,j_m}$ (j_m between 1 and u) for each $1 \leq m \leq t$, and $j_1 \ldots j_t$ are mutually distinct.

3.2.4 Row-Column Block Design (RCBD)

Row-Column Block Design is a type of Block Design that is used to treat problems with two dimensions, that unidimensional Block Designs cannot deal with.

Definition 3.6. A Latin Square is a $n*n$ table filled with n different symbols, in such a way that each symbol occurs exactly once in each row and exactly once in each column. Here is an example:

Example 3.1. $3*3$ *Latin Square*

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{bmatrix}$$

Definition 3.7. If each entry of an $n*n$ Latin Square is written as a triple (r, c, s), where r is the row, c is the column, and s is the symbol, we obtain a set of n^2 triples called the orthogonal array representation of the Latin Square.
See the following example:

Example 3.2. *The Orthogonal Array Representation of the Latin Square of the Example 3.1 displayed above is:*
$\{(1,1,1),(1,2,2),(1,3,3),(2,1,2),(2,2,3),(2,3,1),(3,1,3),(3,2,1),(3,3,2)\}$

Definition 3.8. A Latin Rectangle is a $n*r$ table, with n greater than r in which each row is a permutation of the numbers $1, 2, \ldots, n$. No number appears in a column more than once, and no number appears in a row more than once. See the example below:

Example 3.3. $3*4$ *Latin Rectangle.*

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \\ 3 & 4 & 1 & 2 \end{bmatrix}$$

3.3 Mapping Block Design To Slot-Channel Distribution

In the mapping to slot-channel distribution, set of points X can be considered as a set of nodes in a network, or as vertices (existing link between two nodes). Although, vertices represent the spatial distribution of sensors, we did not use it because, we intend to develop receiver-based protocol, and it fit much more link-based protocol.

In this section we try to apply the concepts and properties of Block Design to develop a dynamic distribution of time slots and channels on nodes. Our work is inspired by [26].

Remark 3.3.1. *We did not use the Symmetric Design ($b = v$ $(k = r)$) because it assumes that the number of points in a source state set is equal to the number of blocks. Our purpose from using Block Design is to group nodes in sets where each set will receive in one slot (channel) so:*
$b = v \implies$ *number of slot (channel = number of nodes.)*
It is very constrained , it does not fit our problem, and it may reduce scalability, because for every new node we add slot(channel).

3.3.1 Mapping Balanced Incomplete Block Design To Slot-Channel Distribution

BIBD is unidimensional design that has one parameter (cardinality of block). It is the first type of Block Design that we used to resolve our problem. We set the mapping and present it in the Table 3.1.
BIBD is included in our solution, but we do not use it as the core of our method because our work has two dimensions: slots and channels.

Remark 3.3.2. *We set $k = 2$ because we scheduled communication links between 2 nodes, but k can take other values. For example, if we want to schedule receivers; k the is number of nodes that receive in the current slot. And if we want to schedule senders k is the number of nodes that send in the current slot. But to settle the correct value of k that will reduce collisions, deafness, and saves energy, it is inescapable to set a central node that has the topology of the entire network and decides the best value for k. Beside that, there will be bottleneck problem when all nodes send their list of neighbors to the central node.*
The value of k is constant and depend, on the number of neighbors of the node, so node's distribution should be highly symmetric to avoid making node awake worthless. When k represent number of nodes that send at the same time, to avoid collision neighbors have to send in separate slots or channels.
Because k is constant, the number of times that every node will receive is constant too, then when nodes i and j have respectively 7 and 2 neighbors, i will be scheduled to receive as much as j, while i have 7 neighbors and j have only 2. So node j will wake up and drain energy listening to the channel uselessly.

From proposition 3.4 the ratios b/λ and r/λ are determined by v and k.
The proposition 3.4 give the following equation:

$$b = \frac{v(v\text{-}1)\lambda}{k(k\text{-}1)} \tag{3.3}$$

Symbol	BIBD	Slot Distribution
$\|X\| = v$	X is the set of points (source states), v is the number of source states	Set of nodes of WSN and v is the number of nodes
$\|B\| = b$	Set of blocks which is a subset of points from X	Set of slots which is a combination of nodes so the number of slots is b
k	The number of points in a block	The number of nodes that have the same slot, so $k = 2$
λ and t	Every t-subset of X is contained in exactly λ blocks	Every combination of two nodes (slot) is contained exactly in one block so $\lambda = 1$ and $t = 2$
r	Constant such that each point lies in exactly r blocks	Each node appear exactly in r slots

TABLE 3.1: Table of mapping Balanced Incomplete Block Design to Slot Distribution in WSN

Example 3.4. *In Mapping Table 3.1, we have assumed that* $k = 2$, $t = 2$ *and* $\lambda = 1$, *so our design will be:* $2 - (v, b, r, 2, 1)$.
Because our issue is time distribution with energy efficiency and r *represent the wake up times of a node, we will build our Block Design in function of* r.

Therefore, Equation (3.3) will be rewritten as:

$$b = \frac{v * (v - 1)}{2} \tag{3.4}$$

and if we apply Equation (3.2), we find:

$$r = v\text{-}1 \tag{3.5}$$

This is the largest value of r, *it gives all the possible combinations in a design.*

Therefore, if $v = 8$ *then* $b = 28$ *and* $r = 7$.
The Block Design becomes $D = 2 - (v, b, r, 2, 1)$, *thus we have:*
In case $r = 1$*:*
$D = \{\{1, 2\}, \{3, 4\}, \{5, 6\}, \{7, 8\}\}$
In case $r = 2$*:*
$D = \{\{1, 2\}, \{3, 4\}, \{5, 6\}, \{7, 8\}\ \{2, 3\}, \{4, 5\}, \{6, 7\}, \{8, 1\}\}$
In case $r = 3$*:*
$D = \{\{1, 2\}, \{3, 4\}, \{5, 6\}, \{7, 8\},\ \{2, 3\}, \{4, 5\}, \{6, 7\}, \{8, 1\},\ \{1, 3\}, \{2, 4\}, \{7, 5\}, \{8, 6\}\}$
In case $r = 4$*:*

$D = \{\{1,2\},\{3,4\},\{5,6\},\{7,8\}\ \{2,3\},\{4,5\},\{6,7\},\{8,1\},\ \{1,3\},\{2,4\},\{7,5\},\{8,6\},$
$\{3,5\},\{4,6\},\{1,4\},\{2,7\}\}$
In case $r = 5$*:*
$D = \{\{1,2\},\{3,4\},\{5,6\},\{7,8\}\ \{2,3\},\{4,5\},\{6,7\},\{8,1\},\ \{1,3\},\{2,4\},\{7,5\},\{8,6\},$
$\{3,5\},\{4,6\},\{1,4\},\{2,7\},\ \{1,5\},\{2,6\},\{3,7\},\{4,8\}\}$
In case $r = 6$*:*
$D = \{\{1,2\},\{3,4\},\{5,6\},\{7,8\},\ \{2,3\},\{4,5\},\{6,7\},\{8,1\},\ \{1,3\},\{2,4\},\{7,5\},\{8,6\},$
$\{3,5\},\{4,6\},\{1,4\},\{2,7\},\ \{1,5\},\{2,6\},\{3,7\},\{4,8\},\ \{1,6\},\{2,5\},\{3,8\},\{4,7\},\ \{1,7\},$
$\{2,8\},\{3,6\},\{5,8\}\}$

In such time distribution, we will have 28 slots of time, all nodes can reach each other in different slot it schedules links between pair of nodes.

3.3.2 Mapping Splitting Design To Slot-Channel Distribution

Splitting Design seems more flexible than BIBD. It has more parameters that generate hierarchy of subsets, because it has b number of blocks, u number of sub blocks, and c number of elements per sub block. We have found that SD can be used to distribute only slots, or slots and channels. We resume the mapping in the Table 3.2.

Example 3.5. *[25].*
A $3-(10,15,6=2*3,1)$ *Splitting Design can be obtained by taking as point set* X *such as:*

$$X = \{1,2,3,4,5,6,7,8,9,0\}$$

and as block set B *such as:*

$$B = \{B_1,\ldots,B_{15}\}$$

with

$$B_1 = \{\{1,2\},\{4,0\},\{5,9\}\}$$
$$B_2 = \{\{1,3\},\{2,8\},\{5,0\}\}$$
$$B_3 = \{\{1,4\},\{3,8\},\{6,9\}\}$$
$$B_4 = \{\{1,5\},\{4,7\},\{6,8\}\}$$
$$B_5 = \{\{1,6\},\{2,3\},\{4,8\}\}$$
$$B_6 = \{\{1,7\},\{2,5\},\{6,9\}\}$$

$$B_7 = \{\{1,8\},\{6,7\},\{9,0\}\}$$
$$B_8 = \{\{1,9\},\{2,5\},\{3,7\}\}$$
$$B_9 = \{\{1,9\},\{3,4\},\{7,0\}\}$$
$$B_{10} = \{\{2,4\},\{5,6\},\{7,9\}\}$$
$$B_{11} = \{\{2,5\},\{4,7\},\{3,0\}\}$$
$$B_{12} = \{\{2,9\},\{6,8\},\{3,0\}\}$$
$$B_{13} = \{\{2,0\},\{4,5\},\{6,8\}\}$$
$$B_{14} = \{\{3,7\},\{4,6\},\{8,0\}\}$$
$$B_{15} = \{\{3,9\},\{5,7\},\{6,0\}\}$$

So we have 10 nodes and 15 channels and 3 slots, in each channel there are 3 pairs of nodes that can communicate in different slot, or 15 slots and 3 channels where every pair of nodes communicate in different channel.
Some nodes can never communicate with each other, for example node 1 *with* $10, 11, 12, 13, 14$ *and* 15.

Example 3.6. $2-(8,7,8=2*4,1)$
$D = \{B_1, B_2, B_3, B_4, B_5, B_6, B_7\}$

$$B_1 = \{\{1,2\},\{3,4\},\{5,6\},\{7,8\}\}$$
$$B_2 = \{\{2,3\},\{4,5\},\{6,7\},\{8,1\}\}$$
$$B_3 = \{\{1,3\},\{2,4\},\{7,5\},\{8,6\}\}$$
$$B_4 = \{\{3,5\},\{4,6\},\{1,4\},\{2,7\}\}$$
$$B_5 = \{\{1,5\},\{2,6\},\{3,7\},\{4,8\}\}$$
$$B_6 = \{\{1,6\},\{2,5\},\{3,8\},\{4,7\}\}$$
$$B_7 = \{\{1,7\},\{2,8\},\{3,6\},\{5,8\}\}$$

In this example, the number of slots will be reduced to 7 instead of 28 in BIBD. We present this slot allocation in Figure 3.1, where we can see when (in which slot) each two nodes can communicate.
We did not fill the other half because it is the same (permutation in Block Design and in our study does not make difference. Blocks $\{1,5\}$ and $\{5,1\}$ are the same).

Our allocation of time slots seems like it is collision-free. In fact it is not, because we did not take in our consideration the position of nodes.
As shown in Figure 3.2, the 3^{rd} slot for every time that Node 3 is sending and Node 5 is receiving, there will be collision at node 3 level (problem of hidden nodes). But if they use different channels (multichannel), there will not be such a problem at all.

TABLE 3.2: Table of mapping Splitting Design To Slot-Channel Distribution in WSN

Symbol	**BD-SD**	**Slot distribution**	**Slot-Channel distribution**
$\|X\| = v$	$\|X\|$ is the set of points (source states), v is the number of source states	Set of nodes of WSN and v is the number of nodes	
$\|B\| = b$	Set of blocks where each block is a set of sub-blocks and every sub-block is a subset of points. From X	set of b slots, where every slot refers, to a combination of nodes that will communicate during it.	
c, u	c is the cardinality (number of points) of sub-block while u is the cardinality of block (number of sub-block)	$c = 2$ because we put nodes in a couple and u is the number of couples that have the same slot.	c is the number of nodes that have the same channel and u is the number of orthogonal channels.
λ and t	every t-subset of X is contained in exactly λ blocks	Every combination of nodes is contained exactly in one block, so $\lambda = 1$ and $t = 2$.	
r	a constant such that each point lies on exactly r blocks	Each node wakes up exactly in r slots.	

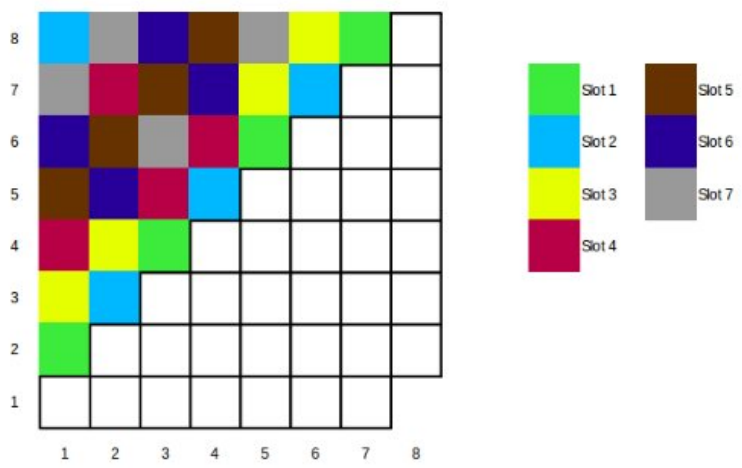


FIGURE 3.1: Slot distribution based on Split Design.

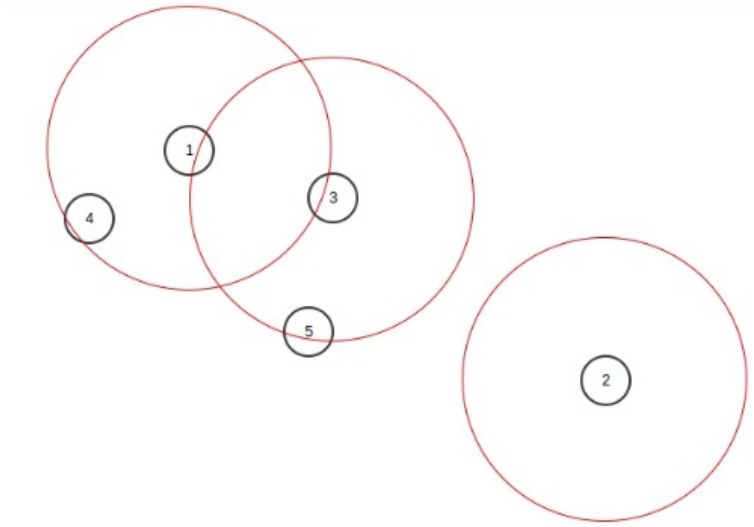


FIGURE 3.2: Networks where nodes face hidden and exposed nodes problem.

The 1^{st} slot of time is allocated to $\{1,2\},\{3,4\},\{5,6\}$ and $\{7,8\}$ but nodes 1 and 2 can not hear each other, but both will wake up each slot and this will consume energy uselessly.

3.3.3 Mapping Latin Square to Slot-Channel Distribution

The main feature of the Latin Square design is that there are the two blocking factors. Each treatment is present at each level of the first blocking factor and is also present at each level of the second blocking factor.
In our studies the two factors are channel and time. Although the formulas associated with a Latin Square are fairly simple, it is an unbalanced design (every pair occurs in exactly λ blocks, where λ is not constant).

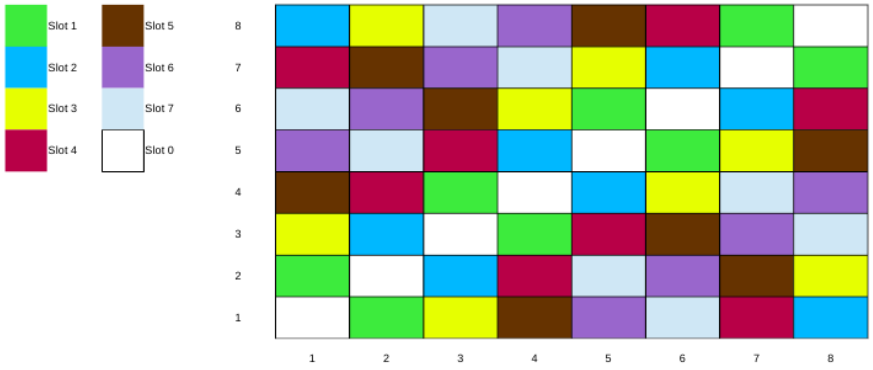


FIGURE 3.3: Time Slots Distribution based on Latin Square.

Latin Square can be represented as a set of triples (r, c, s). In this case, entries (source set) will be links (vertices's) and not nodes, r and c represent nodes whereas s is the time slot that will be used for each combination of nodes.
For example, nodes 1 and 2 have a direct link between them and they can communicate in time slot 1, we will represent this by using the orthogonal array representation as $\{(1, 2, 1)\}$ and for nodes that can not communicate the value of s will be 0.
If we take the same example we used in Splitting Design and re-organize slot distribution so that we respect Latin Square rules, it will look like Figure 3.3 and the corresponding Latin Square will be:

$$\begin{bmatrix} 2 & 3 & 7 & 6 & 5 & 4 & 1 & 0 \\ 4 & 5 & 6 & 7 & 3 & 2 & 0 & 1 \\ 7 & 6 & 5 & 3 & 1 & 0 & 2 & 4 \\ 6 & 7 & 4 & 2 & 0 & 1 & 3 & 5 \\ 5 & 4 & 1 & 0 & 2 & 3 & 7 & 6 \\ 3 & 2 & 0 & 1 & 4 & 5 & 6 & 7 \\ 1 & 0 & 2 & 4 & 7 & 6 & 5 & 3 \\ 0 & 1 & 3 & 5 & 6 & 7 & 4 & 2 \end{bmatrix}$$

The orthogonal array representation of Row 1 is:
$\{(1, 1, 0), (2, 1, 1), (3, 1, 3), (4, 1, 5), (5, 1, 6), (6, 1, 7), (7, 1, 4), (8, 1, 2)\}$
And for Row 8 is:
$\{(1, 8, 2), (2, 8, 3), (3, 8, 7), (4, 8, 6), (5, 8, 5), (6, 8, 4), (7, 8, 1), (8, 8, 0)\}$

3.4 Latin [Square/Rectangle] Based MAC Protocols

Based on Block Design concept our method performs a distributed slot and frequency assignment respecting design drivers for WSN MAC protocols mentioned in [11], with a particular attention to the deafness problem. Based on the resulted reservation, node stays awake (and switch to the appropriate channel) or turns to sleep mode to conserve energy.

In this section, we explain how we use BIBD and Latin Square/Rectangle, to reserve channel and slot to nodes. We extract the Latin Rectangles from the Latin Square adopting a mathematical basis, so that we make sure that all nodes have the same opportunity in assignment, and no constraints are on the number of nodes or channels in the network.

3.4.1 Slot-Channel Distribution

3.4.1.1 Latin [Square/Rectangle] Generation

References [23],[24], and [27] use different methods to generate the Latin Square that suits their study. We thought to exploit the properties of Latin Square as much as possible to provide a fair and dynamical slot and channel reservation, as well as avoid randomness by making all nodes generate the same Latin Square. We start by forming Latin Square from which we extract a set of Latin Rectangles, where a Latin Square represents a super frame (set of frames), and every Latin Rectangle represents a frame (set of slots), whereas, column refers to channel (see Figure 3.6). Our method to generate these Latin Squares is described hereafter.

The core of our procedure is the position of node id in the Latin Square (Rectangle). If we use the method of [21] described in Equation (3.6), channel and slot assignment, will be almost static because nodes show in the same order.

Equation used by [21] to generate Latin Square is:

$$\mathrm{LS}(i,j) = (i+j) \mod n \tag{3.6}$$

Where i is the index of the row and j is the index of the column.
See below, an example of a Latin Square generated by using this method:

$$\begin{bmatrix} 7 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 7 & 0 & 1 & 2 & 3 & 4 & 5 \\ 5 & 6 & 7 & 0 & 1 & 2 & 3 & 4 \\ 4 & 5 & 6 & 7 & 0 & 1 & 2 & 3 \\ 3 & 4 & 5 & 6 & 7 & 0 & 1 & 2 \\ 2 & 3 & 4 & 5 & 6 & 7 & 0 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{bmatrix}$$

To make our allocation of slots and channels more dynamic, we thought to generate our Latin Square by following those steps:

- Node classification in sets.
- Generation of the partial Latin Squares.
- Column permutation (Extraction of Latin Rectangles).

1. Node classification in sets

   We classify nodes in s sets using the following **BIBD** Block Design:

   $D = 2 - (v, b, k, r)$ where:

   v = number of source state $\rightarrow$ n= $\left\lceil \frac{\text{number of nodes}}{\text{number of channels}} \right\rceil$ * number of channels.

   b = number of blocks $\rightarrow$ number of channels.

   k = number of source states in each block $\rightarrow \left\lceil \frac{\text{n}}{\text{number of channels}} \right\rceil$ = number of frames.

   $r = 1$ because we need that each source state appear only in one block.

   $vr = bk$ is always true for our design $\Rightarrow D$ is BIBD.

   We use the notation described in Table 3.3 in our analysis.

Symbol	Description
n	Number of nodes as multiplier of number of channels
nb_channels	The number of orthogonal channels in the WSN
nb_nodes	The number of nodes in the WSN
nb_frames	The number of frames in a super frame
frame_id	Identifier of a frame in a super frame
node_id	Identifier of a node
slot_id	Identifier of a slot
channel_id	Identifier of a channel

TABLE 3.3: Notation used in the analysis.

Example 3.7. *In a network of 10 nodes and 3 orthogonal channels we get:*
D = 2-(12, 4, 4, 1)
D = (0, 1, 2, 3), (4, 5, 6, 7),(8, 9, 10, 11)

2. Generation of the partial Latin Squares
   After organizing nodes in sets, we will deal with block id instead of node id, to generate an initial Latin Square using the method in [21], that will give a $b * b$ matrix.

   Example 3.8. *The initial Latin Square of Example 3.7 is:*

$$\begin{bmatrix} b_2 & b_0 & b_1 \\ b_1 & b_2 & b_0 \\ b_0 & b_1 & b_2 \end{bmatrix}$$

   Now, in the resulted Latin Square, we replace every Block i in the initial Latin Square with a partial Latin Square.
   Partial Latin Square of block i is the Latin Square of its k elements. It is generated using the same method described in the Equation 3.6.

   Example 3.9. *The initial Latin Square of Example 3.8 after replacing every block i with its Partial Latin Square is exhibited in the Fig. 3.4.*

11 8 9 10 10 11 8 9 9 10 11 8 8 9 10 11	3 0 1 2 2 3 0 1 1 2 3 0 0 1 2 3	7 4 5 6 6 7 4 5 5 6 7 4 4 5 6 7
7 4 5 6 6 7 4 5 5 6 7 4 4 5 6 7	11 8 9 10 10 11 8 9 9 10 11 8 8 9 10 11	3 0 1 2 2 3 0 1 1 2 3 0 0 1 2 3
3 0 1 2 2 3 0 1 1 2 3 0 0 1 2 3	7 4 5 6 6 7 4 5 5 6 7 4 4 5 6 7	11 8 9 10 10 11 8 9 9 10 11 8 8 9 10 11

FIGURE 3.4: Initial Latin Square of b partial Latin Squares, where b= number of channels.

3. Column permutation (Extraction of Latin Rectangles)
   Now, our purpose is to extract the Latin Rectangles from the Latin Square.
   We could use a simple method, where every k columns represent one Latin Rectangle, to get b Latin Rectangles.

As we have seen before, b refers to the number of channel which has a very limited value, so whatever is the value of n, we will have b frames. And we will note that there is not a considerable change of node's position between two successive columns. Thereby, we thought to make a column permutation to increase the number of frames in a super frame by using the feature that: $k > b$. Thus, we generate k Latin Rectangles instead of b to make the reservation of slot-channel more dynamic.

To generate Latin Rectangle LR_i that will be used by nodes to reserve slot and frequency, we gather columns j satisfying Eq. (3.7):

$$j \mod \text{nb\_frames} = i \tag{3.7}$$

Figure 3.5 shows column permutation of the introduced Latin Square in example 3.9, where columns that have the same color form one Latin Rectangle.

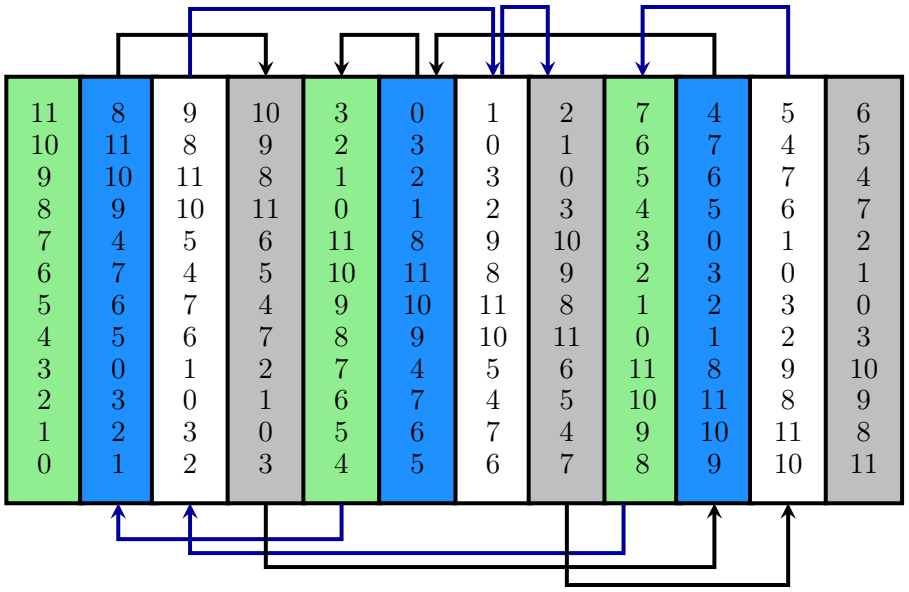


FIGURE 3.5: Columns permutation of a Latin Square to make a Latin Rectangle.

Figure 3.6 shows a view of the final Latin Square and its Latin Rectangles with the significance of every column.

In the resulted Latin Rectangle, if we consider that column identifier refers to channel number, channel allocation will be almost static.
In the example of Fig. 3.6, node 1 has the second position in the first column of the frame 0 and the first position in the first column of frame 1. If the column identifier refers to the channel number, node 1 will allocate the same channel in frames 0 and 1.
For that reason, in networks of nb_channels orthogonal channels, we set Id of

channel represented by the column j in the Latin Rectangle of frame_id is c where:

$$c = (j + \text{frame\_id}\ ) \bmod \text{nb\_channels} \tag{3.8}$$

11	3	7	8	0	4	9	1	5	10	2	6
10	2	6	11	3	7	8	0	4	9	1	5
9	1	5	10	2	6	11	3	7	8	0	4
8	0	4	9	1	5	10	2	6	11	3	7
7	11	3	4	8	0	5	9	1	6	10	2
6	10	2	7	11	3	4	8	0	5	9	1
5	9	1	6	10	2	7	11	3	4	8	0
4	8	0	5	9	1	6	10	2	7	11	3
3	7	11	0	4	8	1	5	9	2	6	10
2	6	10	3	7	11	0	4	8	1	5	9
1	5	9	2	6	10	3	7	11	0	4	8
0	4	8	1	5	9	2	6	10	3	7	11
f0	f1	f2	f1	f2	f0	f2	f0	f1	f0	f1	f2
frame 0			frame 1			frame 2			frame 3		

FIGURE 3.6: The final Latin Square and its Latin Rectangles.

We abbreviate steps listed above, and set Algorithm 1 used by node to generate Latin Square of nb_frames Latin Rectangles in networks of nb_nodes and nb_channels.

We define the function Round as the following:

$$\text{Round}(\text{nb\_nodes}, \text{nb\_frames}) = \left\lceil \frac{\text{nb\_nodes}}{\text{nb\_frames}} \right\rceil * \text{nb\_frames} \tag{3.9}$$

Algorithm 1: LS_Generation

Input: nb_nodes, nb_channels

Output: Latin Square matrix

```
1 nb_nodes ← Round (nb_nodes, nb_channels) ;
2 nb_frames ← nb_nodes / nb_channels;
3 for frame_id ← 0... nb_frames - 1 do
4     for i ← 0...nb_nodes - 1 do
5         for j ← 1...nb_channels do
6             ls_column_id ← frame_id * nb_frames + j ;
7             LS(i, ls_column_id) ← (Round(i, nb_frames) + (i +
              frame_id)%nb_frames + (j − 1) * nb_frames)%nb_nodes ;
```

3.4.1.2 Slot-Channel Allocation Description

The first concern of our slot-channel reservation approach is to avoid deafness problem by preventing the reservation of the same slot by a node and its first hop neighbor.
We set an initial phase, in which every node sends to its first hop neighbors the list of its 1^{st} hop neighbors. The second concern is to reduce idle listening by scheduling the sleep mode, so node listen to the channel only during its slot and spend the rest of time in sleeping.

At first, node runs Algorithm 3. After that, every node broadcasts the max value of slots of every frame. So nodes will have the same number of slots in a super frame and synchronize with each other.

In some cases, the reservation that node predicts for its first hop neighbor will not much the reservation obtained by the neighbor itself. For that reason, every node will re-run Algorithm 3, at this time the node takes node_id as the id of its first hop neighbor to adjust reservation. While node adjusts reservation, if in Frame i it finds that it receive in the same slot of its first hop neighbor, then node deletes allocation of this neighbor in Frame i to prevent deafness.
The details of our method are expressed in Algorithm 2 and Figure 3.7.

Remark 3.4.1. *We manage our method to make node does not keep Latin Square/Rectangles in memory because it has a restricted memory space.*

As it is exhibited in Algorithm 3, SF_Allocation procedure reserves channel_id in slot_id of frame_id to node_id, only if slot_id of frame_id is not allocated to a first hop neighbor of node_id, so deafness problem is completely avoided.
Note that to avoid deafness problem, the slot should not be allocated to a first hop neighbor what ever was the channel.

Algorithm 2: Global SF_Allocation

Input: nb_nodes, nb_channels
Output: Latin Square matrix

```
1  nb_nodes ← Round (nb_nodes, nb_channels) ;
2  nb_frames ← nb_nodes / nb_channels ;
   /* Initialization of the Assignment table                              */
3  for  frame_id ← 0 ... nb_frames - 1 do
4      for every neighbor from list(1st_hop_neighbors) do
5          Assignment[frame_id][neighbor_id] ← (-1, -1) ;
6      Assignment[frame_id][node_id] ← (-1,-1) ;
7      Neighbor_Assignment[frame_id][node_id] ← (-1,-1) ;
8  Assignment ← SF_Allocation (current_node_id) ;
9  for every neighbor from list(1st_hop_neighbors) do
10     Neighbor_Assignment ← SF_Allocation (neighbor) ;
11     for frame_id ← 0 ... nb_frames - 1 do
12         if ( slot(Assignment[frame_id][neighbor]) ≠
           slot(Neighbor_Assignment[frame_id][neighbor]) ) then
13             if (slot(Assignment[frame_id][current_node_id]) ≠
               slot(Neighbor_Assignment[frame_id][neighbor]) then
14                 slot(Assignment[frame_id][neighbor]) ←
                   slot(Neighbor_Assignment[frame_id][neighbor]);
15             else
                   /* Delet allocatin of this neighbor in frame_id        */
16                 Assignment[frame_id][neighbor] = (-1, -1) ;
17 for  every neighbor from list(1st_hop_neighbors) do
       /* Test whether the current neighbor has at least one allocation in a super fame  */
18     if  ∀ frame_id ( Assignment[frame_id][neighbor] = -1 ) then
           /* Add a slot at the end of (nb_frames-1) and allocate it to the current neighbor
              and the current_node_id in channel 0                        */
19         Assignment[nb_frames-1][neighbor] = number max of slots in (nb_frames-1)
           +1 ;
20         Assignment[nb_frames-1][current_node_id] = number max of slots in
           (nb_frames-1) +1 ;
```

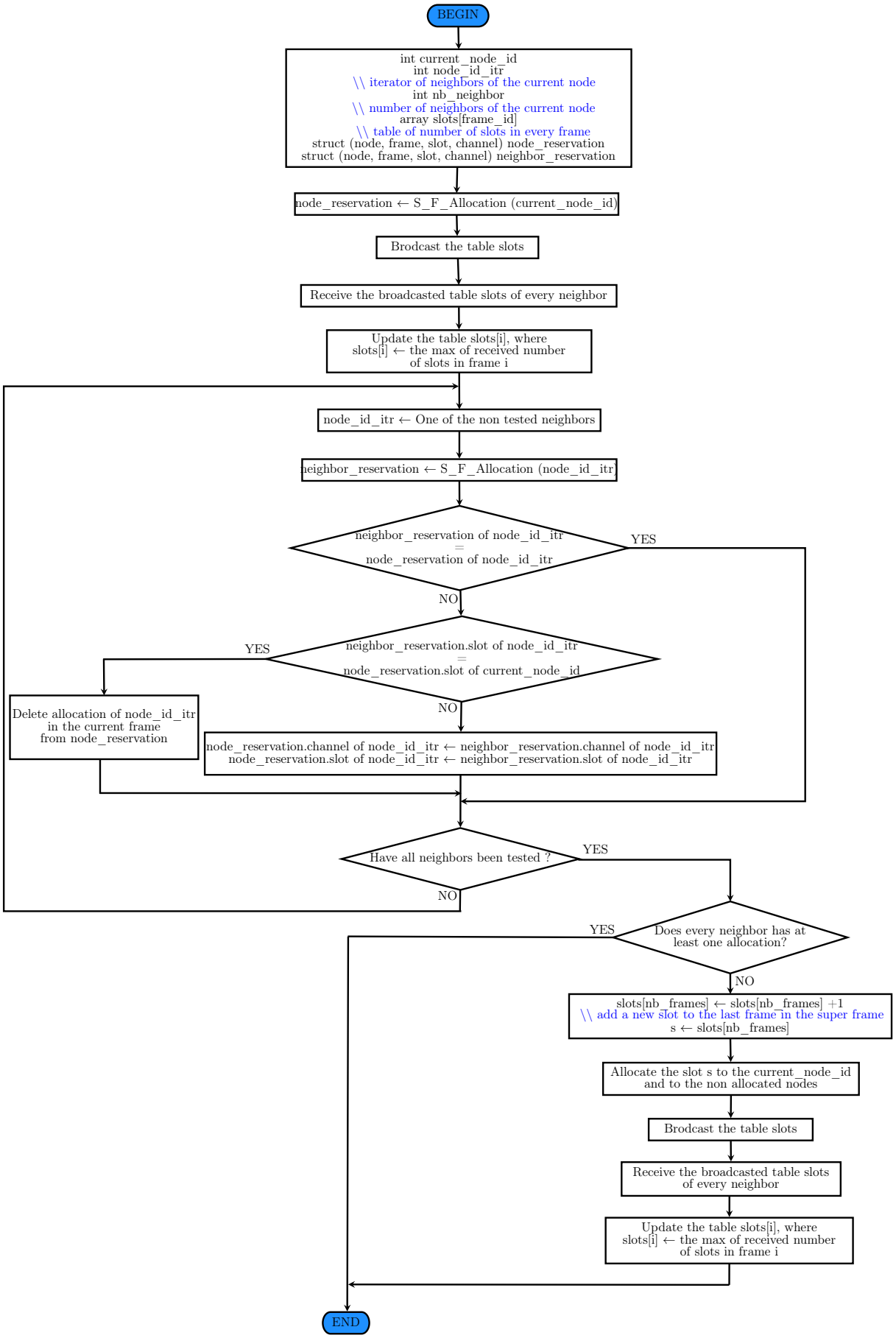


FIGURE 3.7: Reservation of channel and slot process (executed by every node).

Algorithm 3: SF_Allocation

Input: node_id
Output: (node_id, frame_id, slot_id, channel_id)
Data: **Constants**: nb_nodes, nb_channels
Data: Toplogy table (1st and 2nd hop neighbor)

```
1  n ← Round(nb_nodes, nb_channels) ;            // Our Latin Square is a n * n matrix
2  nb_frames ← ⌈n/nb_channels⌉ ;     // Our Latin Rectangle is a n * nb_channels matrix
3  for frame_id ← 0 ... nb_frames -1 do
4      for i ← 0 ... n - 1 do
           /* We set this loop at first to make node switches between channels as much as
              possible. To provide more dynamism in channel allocation. Because when we
              set the next loop (j goes from 1 to nb_channels) at first, most of allocation
              in every frame will be in the channel referred by the first column of the
              Latin Rectangle that represent the frame.                                  */
5          for j ← 1 ... nb_channels do
6              node_id_to_be_allocated ← (Round(i, nb_frames) +
               (i+frame_id)%nb_frames + (j-1)*nb_frames)%n;
7              list(1st_hop_neighbors) ← 1st hop neighbors of node_id;
8              if node_id_to_be_allocated = node_id or node_id_to_be_allocated ∈
               list(1st_hop_neighbors) then
9                  channel_id ← (column_id+frame_id)%nb_channels;
                   /* Test whether node_id_to_be_allocated is already allocated   */
10                 if Assignment[frame_id][node_id_to_be_allocated] = (-1, -1) then
11                     slot_id ← 1;
                       /* because we start from the first slot every time and jump to next
                          slot only if present slot is assigned to a 1st hop neighbor  */
12                     is_slot_free ← false;
13                     while (is_slot_free = false) do
14                         is_slot_free ← true;
15                         list(1st_hop_neighbors) ← 1st hop neighbors of
                           node_id_to_be_allocated;
16                         for every neighbor from list(1st_hop_neighbors) do
17                             if slot(Assignment[frame_id][neighbor]) = slot_id then
18                                 slot_id ← slot_id +1;
19                                 is_slot_free ← false;
20                                 Break;
21                     Assignment[frame_id][node_id_to_be_allocated] =
                       (slot_id,channel_id);
```

Example 3.10. *When we apply our method to the network exhibited in Fig. 3.8. we have to consider that number of nodes is n, so we get m frames where:*
$n = \lceil \frac{8}{3} \rceil * 3$
$m = \lceil \frac{n}{3} \rceil$

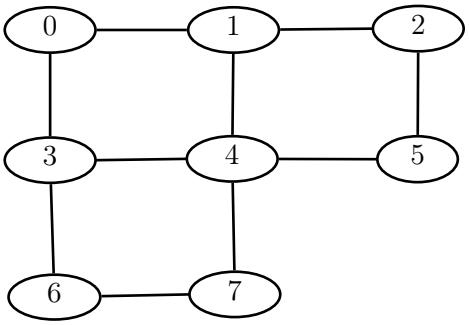


FIGURE 3.8: Example of WSN.

So every node runs Algorithm 2 that follows this Latin Square to reserve slots, and channels in each frame.

$$\begin{array}{ccccccccc} \overbrace{}^{Frame\ 0} & & & \overbrace{}^{Frame\ 1} & & & \overbrace{}^{Frame\ 2} & & \\ f_1 & f_2 & f_0 & f_2 & f_0 & f_1 & f_0 & f_1 & f_2 \\ \hline 0 & 3 & 6 & 1 & 4 & 7 & 2 & 5 & 8 \\ 1 & 4 & 7 & 2 & 5 & 8 & 0 & 3 & 6 \\ 2 & 5 & 8 & 0 & 3 & 6 & 1 & 4 & 7 \\ 3 & 6 & 0 & 4 & 7 & 1 & 5 & 8 & 2 \\ 4 & 7 & 1 & 5 & 8 & 2 & 3 & 6 & 0 \\ 5 & 8 & 2 & 3 & 6 & 0 & 4 & 7 & 1 \\ 6 & 0 & 3 & 7 & 1 & 4 & 8 & 2 & 5 \\ 7 & 1 & 4 & 8 & 2 & 5 & 6 & 0 & 3 \\ 8 & 2 & 5 & 6 & 0 & 3 & 7 & 1 & 4 \end{array}$$

Node determines assignment for itself and every first hop neighbor.
In our example, nodes 0, 4, and 7 get the slot-channel assignment as in Table 3.4.

Remark 3.4.2. *In our example, we set number of nodes to 8 and channels to 3, because with such values we get all cases that node my face during allocation.*
For example, nodes 7 and 4 will get exactly the same slot in every frame, thus, they can not communicate with each other. So they have to add another slot (in this case Slot 5 of Frame 2) and specify it only for (4, 7) link transmission, as it is shown in Table 3.4.
Node 7 can not communicate with node 5 in frame 0 because it receives in the same slot, but they can communicate in Frames 1 and 2.
Nodes 3, 5, 6 receive in Slot 4 of Frame 2 but nodes 3, 5 in Channel f_0 and Node 6 in Channel f_1 therefore, hidden node problem will be avoided.

For Node 0:

Frame 0

	S_1	S_2	S_3	S_4
f_0				
f_1	0		1	
f_2		3		4

Frame 1

	S_1	S_2	S_3	S_4
f_0		4		3
f_1				
f_2	1		0	

Frame 2

	S_1	S_2	S_3	S_4	S_5
f_0	0		1		
f_1		3		4	
f_2					

For Node 4:

Frame 0

	S_1	S_2	S_3	S_4
f_0		6		
f_1	0		1,2	
f_2		3		4

Frame 1

	S_1	S_2	S_3	S_4
f_0		4		3,5
f_1				6
f_2	1		0,2	

Frame 2

	S_1	S_2	S_3	S_4	S_5
f_0	0	2	1		
f_1		3,5		4	
f_2		6			7

For Node 7:

Frame 0

	S_1	S_2	S_3	S_4
f_0		6		
f_1				
f_2		3		7

Frame 1

	S_1	S_2	S_3	S_4
f_0		7		3, 5
f_1				6
f_2				

Frame 2

	S_1	S_2	S_3	S_4	S_5
f_0					4
f_1	5	3			
f_2	6			7	

TABLE 3.4: Slot and Channel Assignment.

3.4.2 Improvement

Our method can be applied to Mobile Wireless Sensor Networks too by scheduling initial phase at the beginning of every super frame.
In Mobile WSNs that have nodes with constant position, nodes need to discover their neighborhood only once. The initial phase is scheduled after the deployment.
We illustrate the cycle in constant network in the Figure bellow:

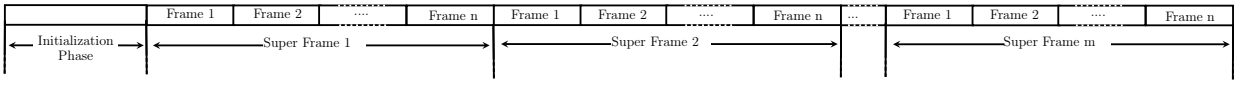


FIGURE 3.9: Cycle in WSNs with constant nodes.

While in Mobile WSNs, nodes' position change in time, thus the neighborhood of the node changes. So they have to schedule the initial phase periodically to update its neighbors list.
According to the velocity of the node, the cycle length is determined. The cycle in Mobile WSNs that have high velocity have to be shorter than the cycle in Mobile WSNs that have low velocity.
We illustrate the cycle in Mobile WSNs in the Figure bellow:

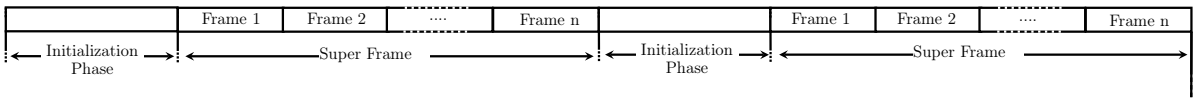


FIGURE 3.10: Cycle in Mobile WSNs, where, the cycle length is equal to the super frame length.

The same principle of our method can be used to avoid hidden node problem in sender-based allocation, but in this case nodes need to know about their first, second and third hop neighbors. And node reserves slot_id of frame_id to node_id, only if channel_id in slot_id of frame_id is not allocated to a second hop neighbor of node_id.

3.5 Conclusions

In this chapter we proposed a new method of reservation of slots in a multi channel wireless sensor network or even in networks which can also be used in the case of one channel. We showed the development of our method from the first phase, where we used the Block Design BIBD in the distribution of channels and slots to the phase where we used combination of BIBD, Latin Square, and Latin Rectangle.
This method of combining TDMA and FDMA takes the advatanges of TDMA of reducing collisions and overcomes its drawbacks, such as the hidden node problem, thanks to FDMA. At the same time, our method overcomes the drawback of FDMA of power consumption by using a sleep mode state that makes nodes save energy.

Chapter 4

Implementation and Experimentation

Contents

4.1 Introduction

Simulation of networks allow emulating the real world communications, thereby allowing a developer to discover potential limitations, problems, design errors, and fix them.
There are many simulators to simulate WSN. NS-2 and NS-3 seem to be the most popular to simulate wireless communication protocols. Since NS-3 is not backward compatible with NS-2, we had to select one of them, and although the number of models and contributed codes in NS-3 is limited in comparison with NS-2, we chose NS-3 because it acts better than NS-2 in many aspects as explained below:

1. In NS-2, bi-language (C++/Tcl) system make debugging complex, but for NS-3 only knowledge of C++ is required (single-language architecture is more robust in the long term).
2. The total computation time required to run a simulation scales better in NS-3 compared to NS-2. This is due to the removal of the overhead associated with interfacing oTcl with C++, and the overhead associated with the oTcl interpreter.
3. NS3 performs better than NS-2 in terms of memory management.
4. NS3 has an emulation mode, which allows for the integration with real networks.

4.2 NS-3 MAC Based Components

The NS-3 is a discrete-event network simulator. Its core and models are implemented in C++.
There are several modules such Mesh, WiMAX and WiFi that can set wireless communication, but the nearest one that fits our situation is WiFi. Therefore, in this section we will describe the main components that are responsible on WiFi packet transmission/reception in NS-3 simulator, particularly WifiNetDevice and WifiChannel.

4.2.1 WifiNetDevice

Fig. 4.1 shows WifiNetDevice architecture and the role of every components in transmission/reception process with its corresponding stage as well.
In order to clarify Fig. 4.1, we use the following explanation.
WifiNetDevice components are:

1. WifiPhy
   WifiPhy behaves as a state machine as illustrated in Fig. 4.2 and clarified below:

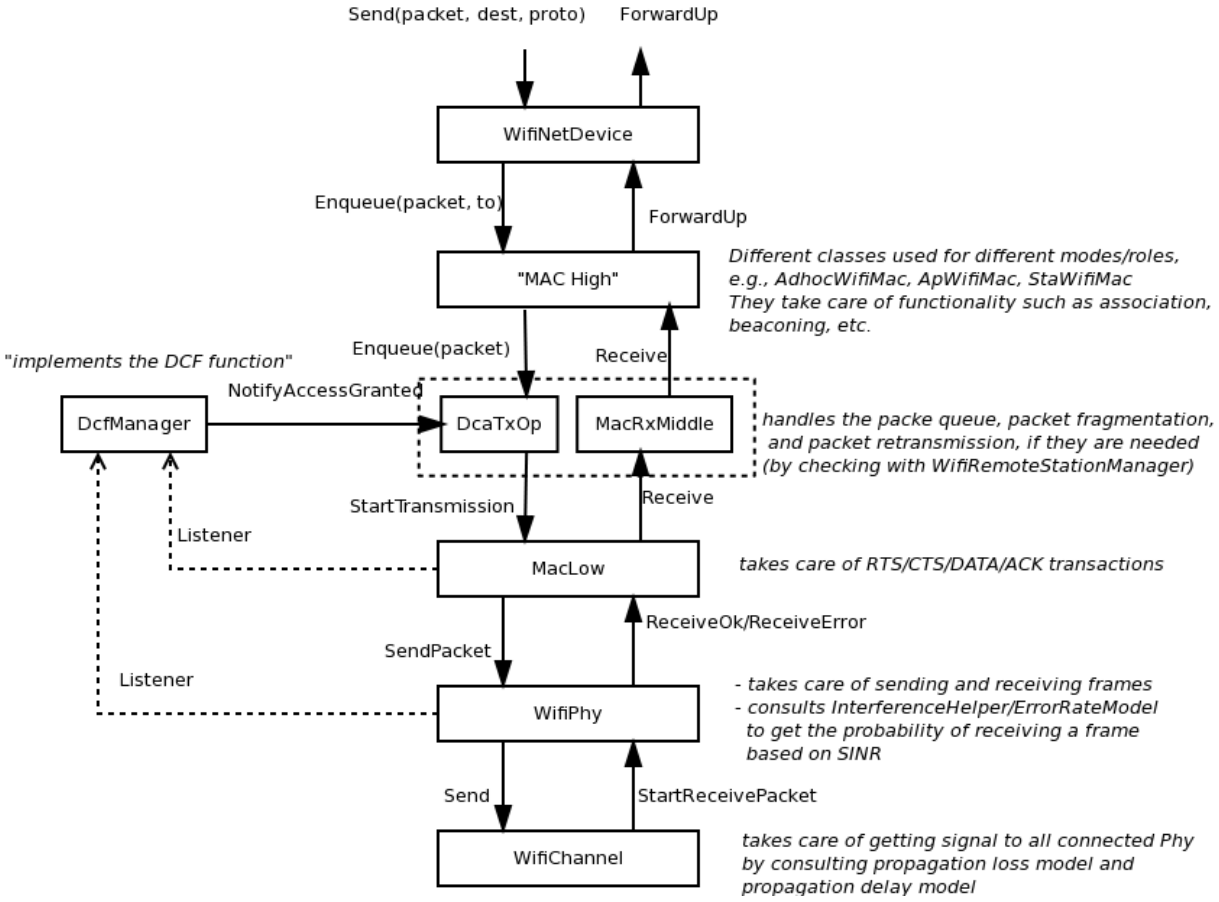


FIGURE 4.1: WifiNetDevice architecture [9].

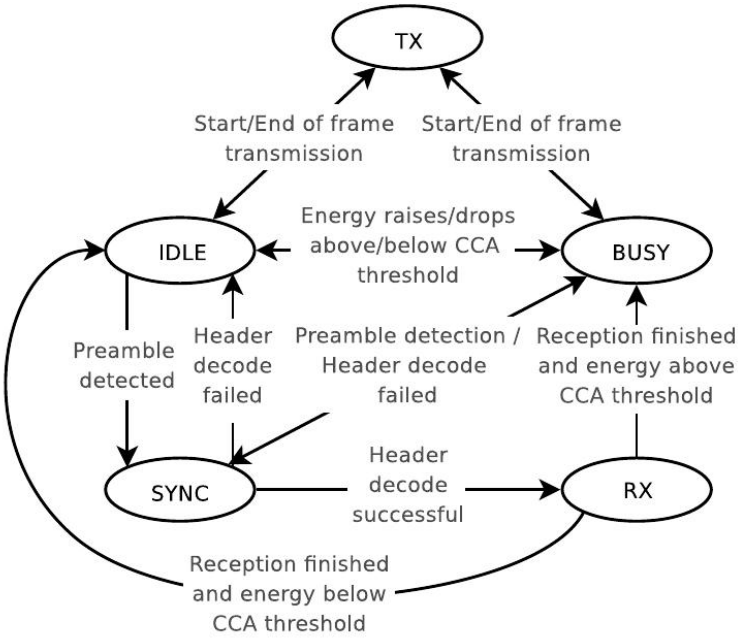


FIGURE 4.2: The state machine of the physical layer emulator distinguishes between Tx, Idle, Busy, Sync and Rx states [10].

(a) Idle: as long as the energy detected in the receiver is below CCA threshold, or no header is successfully decoded.

(b) Sync: detects preamble. It is eventually followed by Rx state if the corresponding header is decoded successfully.

(c) Busy: The energy detected in the receiver is more than CCA threshold. While no signal header is successfully decoded.

(d) Tx: The node is transmitting.

(e) Rx: The node is receiving.

The YansWifiPhy is the only implementation of the physic layer model, (see [28]). It works seamlessly with YansWifiChannel and interference is considered by InterferenceHelper that is based on ErrorRateModel.

2. WifiMac
   There are four models of MACLayer which are inherited from the regular WifiMac and they are:

   (a) AdhocWifiMac

   (b) ApWifiMac

   (c) StaWifiMac

   (d) MeshWifiInterfaceMac

3. WifiRemoteStationManager
   It maintains a list of connected stations to the wireless network, information on its states, and it incarnates an algorithm of adaptation of throughput.

4.2.2 WifiChannel

WifiChannel components are:

1. PropagationLossModel
   This model allows simulating the power loss (attenuation) of signal operating in a transmission channel. It actually allows calculating the power of receiving of destination node, which is used to determine whether it can receive the signal. This calculation is based on the transmission power of the source node and the position of both the source and destination nodes (which depends on a mobility model).

2. PropagationDelayModel
   This model allows simulating delay of propagation in a transmission channel, it is based on the position of the source and destination nodes (which depends on a mobility model).

3. List of pointers to every node.

4.3 Implemented Modules

After studying and understanding wireless networks in NS-3, we found that the best module that will contain our method is WiFi module. So we adapt it on our needs and inject a new layer (SlotChannelMannager) between MAC and Phy layers.
In this section, we present the adaptation that we made on WiFi module and different steps of the implementation of the new layer.

4.3.1 Implementation of SlotChannelManneger

All the changes that we have made on WiFi module keep respect to class architecture (class aggregation) of NS-3.

To inject the new layer, we added a function AllocatorInstall to NetDeviceContainer class to be used instead of Install function. At this stage we determine the number of channels, and the use of sleep mode whether it will be enabled or disabled.

Al the level of Dynamic Channel Assignment (DCA), we make the operation of enqueue packet delayed until the wake up time of the receiver. So when node i sends to j at second 1, but the wake up slot of j is 3, then DCA layer schedules the enqueue operation of the packet after 2 seconds.

At the beginning of simulation (Initial phase), nodes elaborate their table of slot-channel allocation. So all of aforementioned modification are not effective until the end of initial phase, where nodes recognize their neighbors and finish elaborating table of slot-channel allocation. Thus, in initial phase nodes DCA does not delay the enqueue of packets.

4.3.2 Implementation of Neighbor Discovery

Like we have seen in Chapter 3, nodes of the network that will follow our method have to know about their first, second and third hop neighbors. So we have to add these functionalities to node, because it is not implemented in the version NS-3.21.

Neighbor discovery can be done at the routing layer and it is already implemented in AODV protocol. There is a class neighbor and we can use the member TTL to know neighbor's degree, whether it is 1^{st} or 2^{nd} hop. But we did not use it, because it is available at routing layer, while we settle our method at MAC layer.

The node distinguishes its neighbors by using the MAC or IP address, but in our reservation procedure of slot and channel, node needs its neighbors id. We can get them by adding a function that downcast MAC address to int, but we can not use MAC address at low layer (Phy layer). For that reason, we thought to set it with another method.
So in initial phase, when node succeeds in packet reception, it updates a shared global table of neighbors and sets packet's sender as a 1^{st} hop neighbor.
At the end of this phase, the shared global table of neighbors contains topology of the entire network. Now, each node extracts from this table a local table of neighbors that contains only its first, second and third hop neighbors. According to this table, node runs the process of channel and slot reservation.

4.3.3 Implementation of Multichannel

Multi channel can be implemented by using a feature that nodes in NS-3 can have more than one interface (net device), unlike NS-2 where node can only have one interface.
In the reference [29], authors create a multi radio wireless node that has more than one NetDevice to implement the multichannel.

In order to set a multichannel communication, we can make nodes have net devices as much as number of channels and we will set for every net device a different parameter like channel id and frequency. However, we did not use that option because the switching of channels will be scheduled at high layers (and may be at routing layer), while we have to implement it at low layer (MAC).

Channels are distinguishable only by their numbers (Ids). Current NS-3 doesn't account for inter channel interference. SetChannelNumber is a function that sets Id of channel according to the given value.
After initial phase, nodes determine their slots and channels. So every wake up slot, node switches to the appropriate channel to start receiving the transmitted data of its neighbors.

4.3.4 Implementation of Sleep mode

We used the sleep mode to reduce energy consumption and the depletion of energy is not handled; node does not switch to sleep mode when energy is depleted.

Sleep mode is scheduled to make node resume from sleep mode only during its slot or when transmitting. Setting sleep mode or resuming from it, is done by using the functions SetSleepMode and ResumeFromSleep.
So after initial phase (neighbor discovery), node sets sleep mode to reduce energy consumption. Node resumes from sleep mode only during its slot or when transmitting as it is showed in Fig. 4.3.

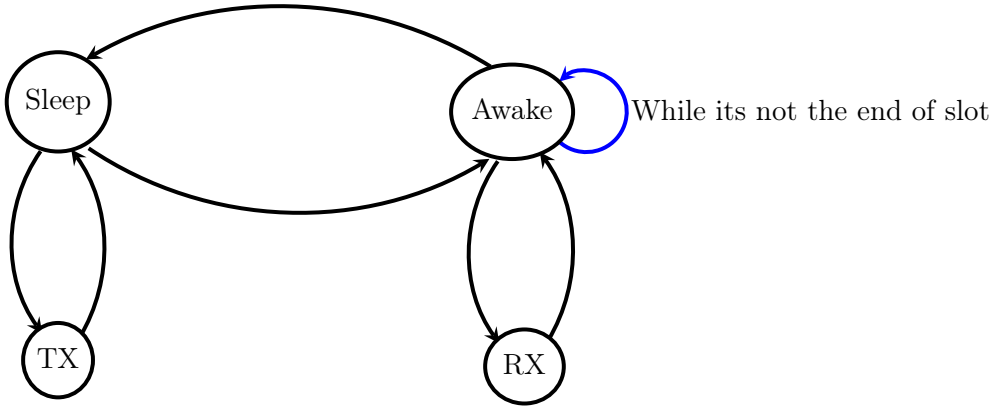


FIGURE 4.3: Abstract states of Phy layer that distinguishes between Sleep, Awake, RX and TX states after the installation of SlotChannelMannager layer.

4.4 Experimentation Environment

4.4.1 Types of Topologies

We experiment in two different topologies, grid and random:

1. Grid Tpopology
   We used grid topology where the maximum number of neighbors is 8 and the minimum number of neighbors is 3, as shown in Fig. 4.4. Where neighbors of node 0 are: 1, 10, and 11, while neighbors of node 11 are: 0, 1, 2, 10, 12, 20, 21, and 22.

2. Random Topology
   In the random topology, the number of neighbors is unpredictable, but it varies in function of the network density.
   We considered the network density as the number of nodes in a m^2. We used two random networks. In the first one, the network of x nodes will be distributed in $x * 50m^2$, and in the second, nodes will be distributed in $x * 20m^2$ as shown in Fig. 4.5.

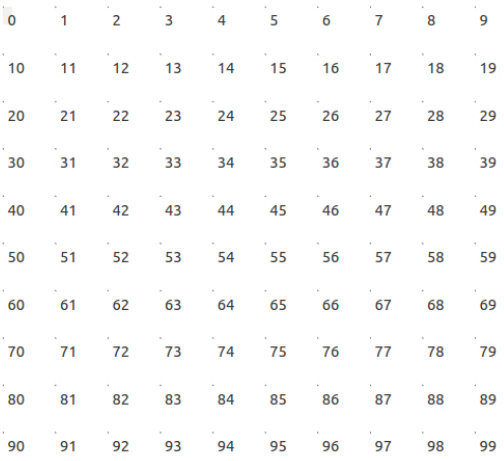
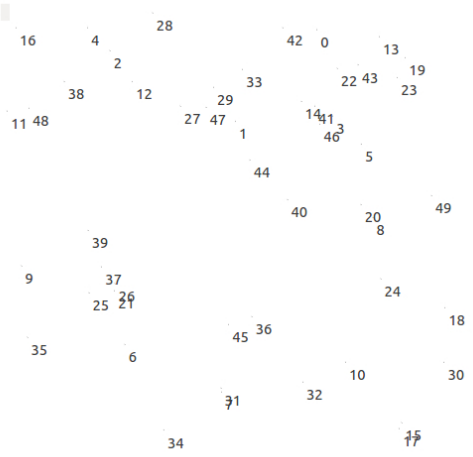
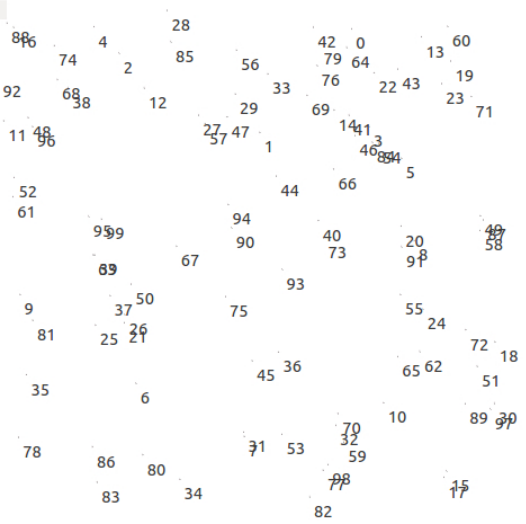


FIGURE 4.4: Grid topology of network of 100 nodes.



(A) Random topology of network, where 100 nodes are distributed in $5000m^2$, and all nodes have the same range = 120m.



(B) Random topology of network, where 100 nodes are distributed in $2000m^2$, and all nodes have the same range = 120m.

FIGURE 4.5: Random topology of network where 100 nodes distributed in different areas.

4.4.2 Traffic Generation

To evaluate our approach, we generate traffic using two different methods. The first one is based on a loop from 1 to the number of nodes in the network, and for 5 neighbors of random node schedule transmission operation as it is exhibited in Algorithm 4.
We set that the node send 5 packets in 1 second with packet interval equal to 0.2 second, Because node can receive at most 5 packets per second.
The second procedure schedules neighbors to send simultaneously to nodes that have the same wake up slot, as it is exhibited in Algorithm 5.

Algorithm 4: Traffic Generation_1

Input: nb_nodes
Output: traffic of k packets

```
1  k ← 0 ;
2  for n ← 0 ... nb_nodes - 1 do
3      sender ← random non isolated node;
       /* Isolated node is the node that have 0 neighbor.                    */
4      list_receivers ← list of neighbors of node sender ;
5      t ← 0 ;
6      for i ← 1 ... 5 do
7          receiver ← random neighbor from list_receivers ;
8          sender send to receiver second t ;
9          k ++ ;
10         t ← t + 0.2 ;
```

We used this way of generating traffic to validate the performance of our method when neighbors transmit to the receiver node in the same slot where it wakes up. Thus, we generate a high amount of traffic in the entire network.
So, when we use the procedure Traffic Generation_1 in a network of 100 nodes, the

Algorithm 5: Traffic Generation_2

Input: id_nodes
Output: traffic of k packets

```
1  k ← 0 ;
2  super_frame_length ← length of super frame ;
3  n ← random node from network ;
4  for i ← 0 ... super_frame_length do
5      list_receivers ← list of nodes that receive at slot i in the reservation table of node n
       ;
       /* We set 4 because we assumed that node can receiver 5 packets during its slot   */
6      foreach element of list_receivers do
7          for j ← 1 ... 4 do
8              sender ← random neighbor of receiver ;
9              sender send to receiver at second i ;
10             k ++ ;
```

generated traffic will be 500 packets per 100 seconds, and the use of the procedure Traffic Generation_2 in a network of 100 nodes and 3 channels in grid topology, generates traffic of 544 packets. Then, to generate considerable traffic for the evaluation of our method, we schedule both of procedures several times during simulation.

The configuration of the simulation's context is shown in Table 4.1.

Parameter	**Value**
Standard	ns3::WIFI_PHY_STANDARD_80211b
Remote station manager	ns3::ArfWifiManager
Propagation delay model	ns3::ConstantSpeedPropagationDelayModel
Propagation loss model	ns3::LogDistancePropagationLossModel
Time of simulation	number of nodes in network * 25 seconds
Node Initial Energy	100 J
Transmission gain	1.0 dB
Reception gain	1.0 dB
Channel Switch Delay	250 Micro Second
The radio Idle current	0.273 Ampere
The radio CCA Busy State current	0.273 Ampere
The radio Tx current	0.380 Ampere
The radio Rx current	0.313 Ampere
The radio Channel Switch current	0.273 Ampere
The radio Sleep current	0.033 Ampere
The radio Rx current	0.313 Ampere

TABLE 4.1: Parameters used in simulation.

4.4.3 Evaluation Metrics

We evaluate the performance according to three metrics: Packet Delivery Ratio, End-to-End Delay, and Power Consumption.

1. Packet Delivery Ratio
   In this experiment, we calculate the ratio of the number of correctly received packets compared to the number of transmitted packets, as shown in the Equation (4.1). Packets of initial phase are not considered.

$$\frac{\text{number of received packets}}{\text{number of transmitted packets}} \tag{4.1}$$

2. End-to-End Delay
   In this experiment, we calculate the average time taken by a data packet to arrive at the destination. It also includes the queue in data packet transmission.The only data packets that have been successfully delivered to destinations that counted, using the following formulas:

$$\frac{\sum \text{ end to end delay}}{\text{number of received packets}} \tag{4.2}$$

3. Power Consumption
   In this experiment, we calculate the total energy consumption of the device at least

nine seconds before the end of the simulation time. We use the following formulas:

$$\text{Power consumption} = \sum \text{Total energy consumption of the device.} \tag{4.3}$$

4.5 Performance Evaluation

4.5.1 Performance in Terms of PDR Metric

Collisions are the main reason why packets do not get received correctly. Collisions occur when two nodes choose to transmit to the same node. This is exacerbated in receiver-based slot distribution as all traffic intended for the nodes gets synchronized at the beginning of the wake-up slot of the destination node.

The frame length may play a role in the amount of collisions, because when node sends a packet from application layer. The operation of transmitting will be delayed at lower layer until the nearest slot of the receiver.

For example, nodes 1, 2, 3, and 4 transmit packets to node 5 respectively at seconds: 1, 2, 3, and 4. If the length of frame 1 is 4 and node 5 receives in slot 4, all packets will be transmitted in the same time to node 5. If frames 1 and 2 have the same length, which is 2 and node 5 receives in the 2^{nd} slot of frames 1 and 2 then, the packets of nodes 1 and 2 will be transmitted in the second slot of frame 1 and the packets of nodes 3 and 4 will be transmitted in the second slot of frame 2.

The frame length is strongly related to the number max of first hop neighbors of every node in the network.

The resulted plots from our simulation that represent packet delivery ratio are shown in Fig. 4.6.
In Figures 4.6b and 4.6c, we show that PDR is not equilibrated, because of the traffic generating procedure in Algorithm 5, that is based on the table of slot-channel allocation of a randomly chosen node. If we run this on Example 3.10, and suppose that the randomly chosen node is 4, Table 3.4 shows that in Frame 1 and 2, Nodes 3 and 6 receive at the same slot and in the same channel so the probability of collision will be high. However, if the randomly chosen node is 0, there is not any nodes that receives in the same slot and channel so probability of collision will be very low.
This plots show that a higher number of channels gives a better performance.

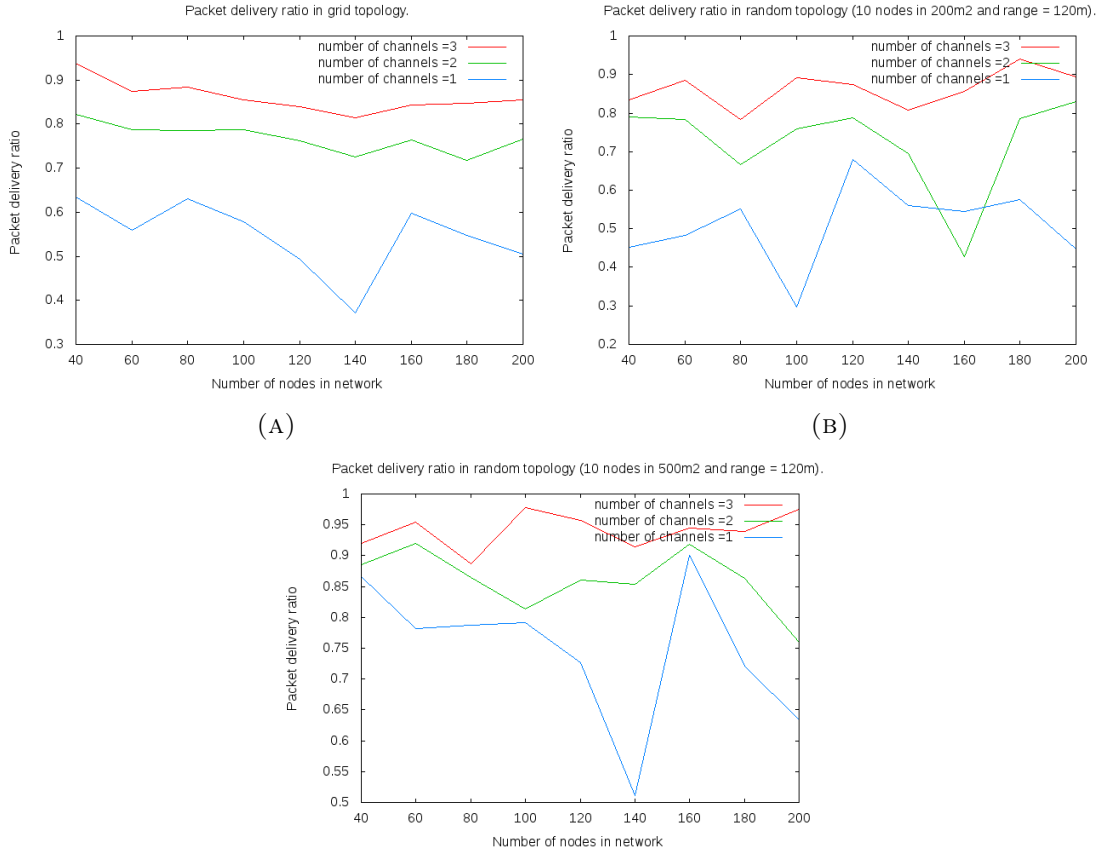


FIGURE 4.6: Packet delivery ratio in grid and random network topologies.

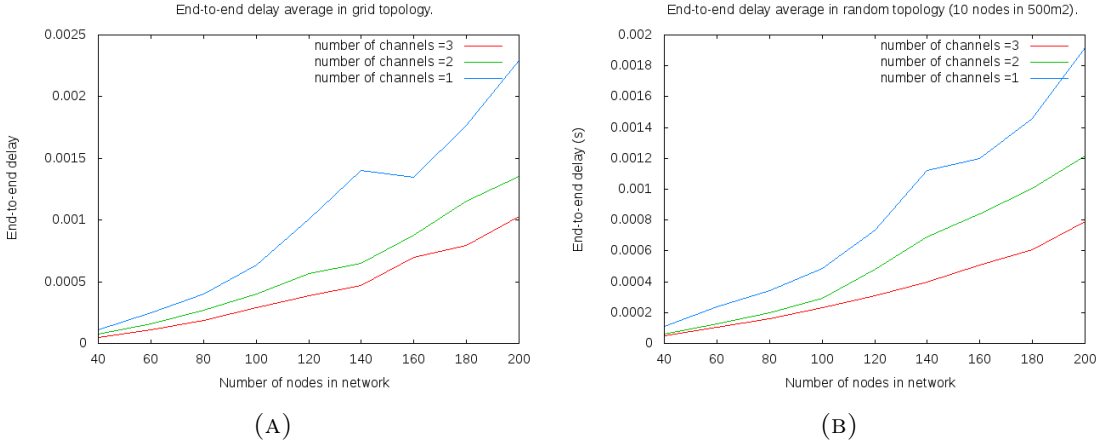
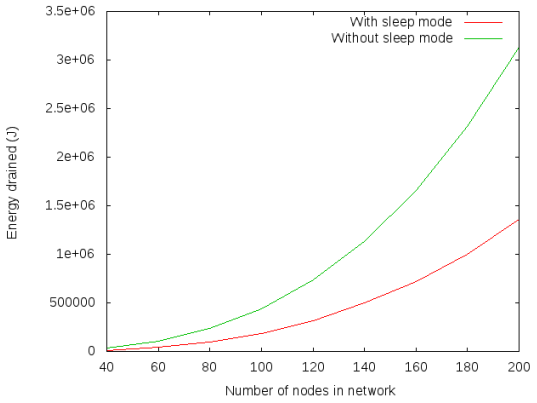


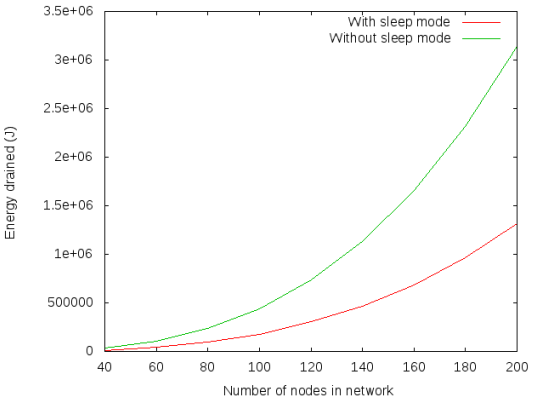
FIGURE 4.7: End-to-end delay average in grid and random network topologies.

4.5.2 Performance in Terms of End-to-End Delay Average Metric

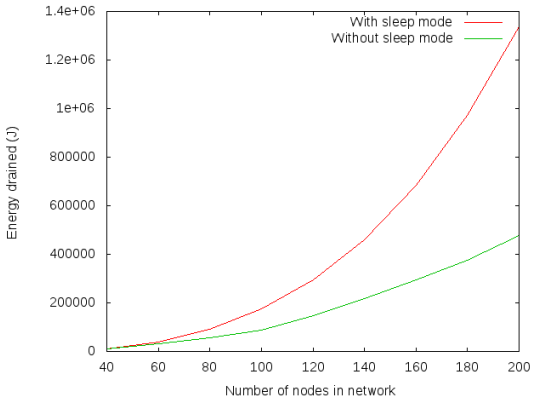
The resulting plots that represent end-to-end delay are shown in Fig. 4.7.



(A) Drained energy in a network of 3 channels in a random topology (10 nodes in 500m2).



(B) Drained energy in a network of 2 channels in a random topology (10 nodes in 500m2).



(C) Drained energy in a network of 1 channel in a random topology (10 nodes in 500m2).

FIGURE 4.8: Drained energy in a network of a random topology.

4.5.3 Performance in Terms of Power Consumption Metric

Figures 4.9 and 4.8 represent the performance of our method in terms of energy consumption. We run our simulation in two different configurations. The first one enable the sleep mode, thus node wakes up during its slots or when it is transmitting. The second disable sleep mode, thus node stay awake during all the simulation time.

In Fig. 4.10 where we combined power consumption of the 3 channels together, we remark that when the network has either 1, 2 or 3 channels, the results are almost the same.

Energy consumption in our method has two sides. The first one saves energy through the use of sleep mode, which consumes only 0.033 Ampere. The second one drains energy in the switching between the different channels which consumes 0.273 Ampere, as it is mentioned in the Table 4.1.

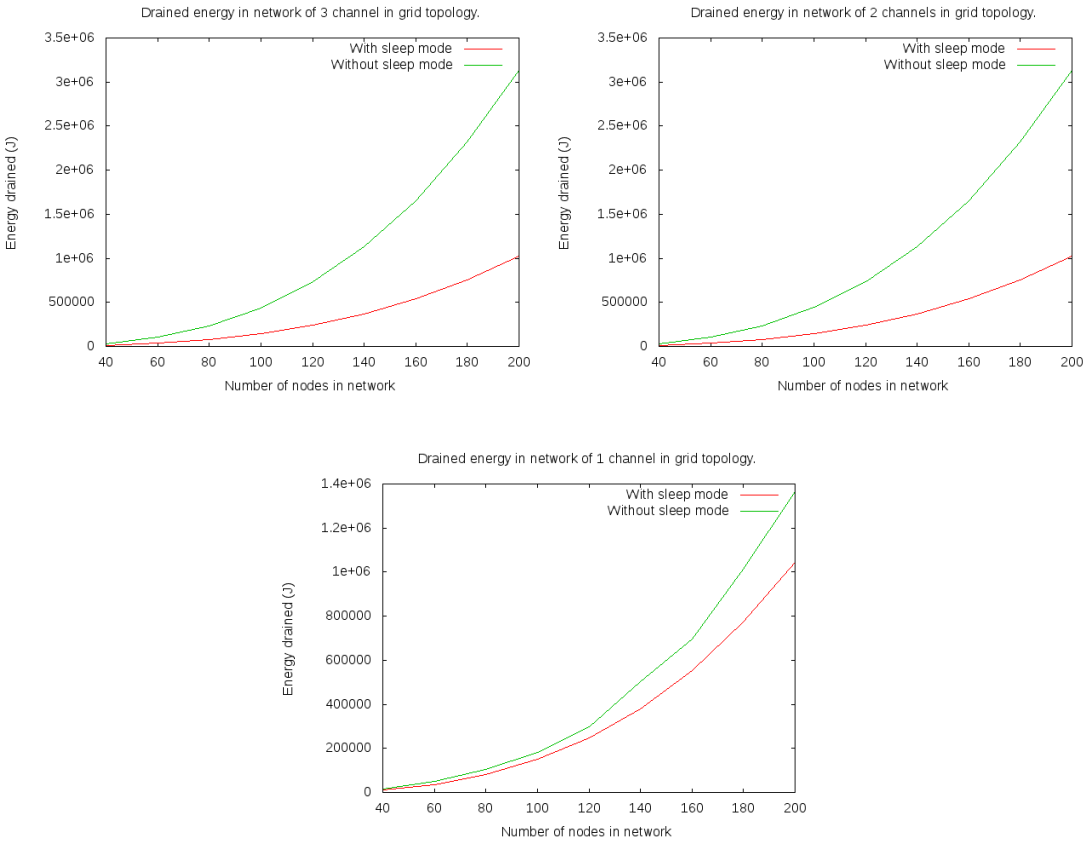


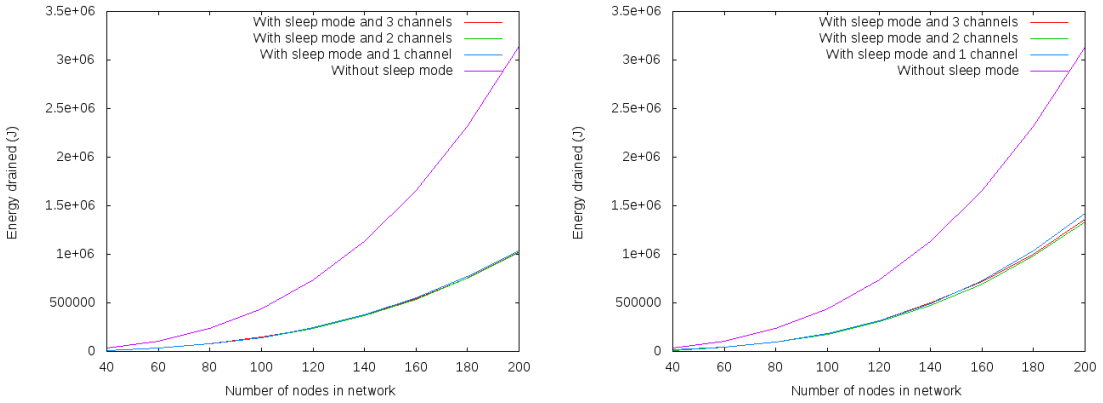
FIGURE 4.9: Drained energy in a network of grid topology.

In-multi channel network, node drains the energy during the switch between the channels. While it economizes energy because it has a low number of wake up slot. As we defined in Chapter 3, every node wakes up nb_frames time in a super frame, where:

$$\text{nb\_frames} = \left\lceil \frac{\text{n}}{\text{number of channels}} \right\rceil$$

For that reason, node in multi channel network have lower number of wake up slots than node in network that have one channel.

On the other hand, the nodes in the network of one channel lose energy since it has a higher number of wake up slots, and it reduces energy consumption by the absence of the channel switching.
From Algorithm 3, we find that node in the network of 10 nodes and 1 channel will wake up 10 times in a super frame, whereas node in multi channel network (3 channels) of 10 nodes will wake up 4 times in a super frame.
So in multi-channel networks, energy drained in switching will be saved in reduced number of awake slots and the inverse for network of one channel, energy saved without the switching will be drained in high number of awake slots, this is the reason that



(A) Drained energy in a network of a grid topology of 3 cases of the number of channels.

(B) Drained energy in a network of a random topology of 3 cases of the number of channels.

FIGURE 4.10: Drained energy in different network topologies.

makes drained energy when nodes use the sleep mode in different number of channels convergent.

4.6 Conclusions

In this chapter, we validated our method of slot-channel reservation by simulation. The resulted plots represent the performance in the two types of topologies: grid and random, with different numbers of channels starting from one channel to three channels.
Graphs show that a higher number of channels gives better results and that our method is not so much affected by the number of nodes in the network. And it achieves good results even in large networks.

General Conclusions

Due to the ease of deployment of wireless sensor networks in various environments, they are prominent in several domains such as military, environmental, etc. Although, WSN have been researched and deployed since decades, energy consumption remains one of the main problems that WSNs face.

Our objective was to propose a new method to overcome one of the most important problems of wireless sensor networks, which is energy consumption.

The layer that drains most of energy is MAC, for that reason, we were interested in MAC protocols, and addressed our method to be implemented at MAC layer.

Our method uses TDMA, and FDMA to allocate slots and channels to nodes in a dynamic, and distributed manner. It is based on the concepts of Block Design for the distribution of slots and channels in WSN. The combination between TDMA with FDMA helps solve these problems and achieves better performance in terms of either throughput or energy consumption.

Our method is validated by using NS-3 simulator. The resulted plots from simulation show that our method is effective in terms of three factors: PDR (Packet Delivery Ratio), EED (End-to-End Delay), and power drain, even in large network. The results we have obtained show that as much as the number of channels increased, the network gives better performance.

The core of our method could be tuned to achieve different goals, according to what is needed. For example, if we aim to avoid hidden node problem, we can develop a sender-based protocol where the node will reserve a slot in a channel only if those have not already been allocated to a second hop neighbor. Thus, node will never send at the same slot and channel with its 2^{nd} hop neighbor, and hidden node problem will be completely avoided.

Moreover, we could improve this method to be used on mobile networks by only rescheduling the initial phase after every super-frame.

Bibliography

[1] K. Sohraby, D. Minoli, and T. Znati. *Wireless Sensor Networks: Technology, Protocols, and Applications*. Wiley, 2007. ISBN 9780470112755.

[2] K.G. Langendoen. Medium access control in wireless sensor networks. In H. Wu and Y. Pan, editors, *Medium Access Control in Wireless Networks*, pages 535–560. Nova Science Publishers, Inc., may 2008. ISBN 1-60021-944-6.

[3] J. Kabara and M. Calle. Mac protocols used by wireless sensor networks and a general method of performance evaluation. *International Journal of Distributed Sensor Networks*, 2012.

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

[5] Pei Huang, Li Xiao, S. Soltani, M.W. Mutka, and Ning Xi. The evolution of mac protocols in wireless sensor networks: A survey. *Communications Surveys Tutorials, IEEE*, 15(1):101–120, First 2013. ISSN 1553-877X.

[6] W.W. Dargie and C. Poellabauer. *Fundamentals of Wireless Sensor Networks: Theory and Practice*. Wireless Communications and Mobile Computing. Wiley, 2010. ISBN 9780470975688.

[7] H. Karl and A. Willig. *Protocols and Architectures for Wireless Sensor Networks*. Wiley, 2005. ISBN 9780470519233.

[8] Youngmin Kim, Hyojeong Shin, and Hojung Cha. Y-mac: An energy-efficient multi-channel mac protocol for dense wireless sensor networks. In *Information Processing in Sensor Networks, 2008. IPSN '08. International Conference on*, pages 53–63, April 2008.

[9] https://www.nsnam.org/docs/models/html/wifi. Last updated on Mar 30, 2015.

[10] S. Papanastasiou, J. Mittag, E.G. Strom, and H. Hartenstein. Bridging the gap between physical layer emulation and network simulation. In *Wireless Communications and Networking Conference (WCNC), 2010 IEEE*, pages 1–6, April 2010.

[11] A. Bachir, M. Dohler, T. Watteyne, and K.K. Leung. Mac essentials for wireless sensor networks. *Communications Surveys Tutorials, IEEE*, 12(2):222–248, Second 2010. ISSN 1553-877X.

[12] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38:393–422, 2002.

[13] Q. Wang and I. Balasingham. *Wireless Sensor Networks - An Introduction*. INTECH Open Access Publisher, 2010. ISBN 9789533073217.

[14] Abdelmalik Bachir, Martin Heusse, Andrzej Duda, and Kin K Leung. Preamble sampling mac protocols with persistent receivers in wireless sensor networks. *Wireless Communications, IEEE Transactions on*, 8(3):1091–1095, 2009.

[15] R. Soua and P. Minet. A survey on multichannel assignment protocols in wireless sensor networks. In *Wireless Days (WD), 2011 IFIP*, pages 1–3, Oct 2011.

[16] Mo Sha, Gregory Hackmann, and Chenyang Lu. Energy-efficient low power listening for wireless sensor networks in noisy environments. In *Proceedings of the 12th International Conference on Information Processing in Sensor Networks*, IPSN '13, pages 277–288, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1959-1.

[17] Abdelmalik Bachir, Fabrice Theoleyre, Andrzej Duda, and Kin K Leung. Energy-efficient broadcasts in wireless sensor networks with multiple virtual channels. In *Wireless Communications and Networking Conference (WCNC), 2010 IEEE*, pages 1–6. IEEE, 2010.

[18] Gang Zhou, Chengdu Huang, Ting Yan, Tian He, J.A. Stankovic, and T.F. Abdelzaher. Mmsn: Multi-frequency media access control for wireless sensor networks. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–13, April 2006.

[19] D.R. Stinson. *Combinatorial Designs*. Springer, 2004. ISBN 9780387954875.

[20] C.J. Colbourn and J.H. Dinitz. *Handbook of Combinatorial Designs, Second Edition*. Discrete Mathematics and Its Applications. CRC Press, 2007. ISBN 9781439832349.

[21] Chih-Min Chao and Hsiang-Yuan Fu. Providing complete rendezvous guarantee for cognitive radio networks by quorum systems and latin squares. In *Wireless Communications and Networking Conference (WCNC), 2013 IEEE*, pages 95–100, April 2013.

[22] Ji-Her Ju and V.O.-K. Li. Tdma scheduling design of multihop packet radio networks based on latin squares. *Selected Areas in Communications, IEEE Journal on*, 17(8): 1345–1352, Aug 1999. ISSN 0733-8716.

[23] Lichun Bao. Mals: multiple access scheduling based on latin squares. In *Military Communications Conference, 2004. MILCOM 2004. 2004 IEEE*, volume 1, pages 315–321 Vol. 1, Oct 2004.

[24] Lichun Bao and Shih-Hsien Yang. Latin square based channel access scheduling in large wlan systems. In *Wireless Communications and Networking Conference (WCNC), 2011 IEEE*, pages 375–380, March 2011.

[25] M. Huber. *Combinatorial Designs for Authentication and Secrecy Codes*. Foundations and trends in communications and information theory. Now Publishers, 2010. ISBN 9781601983589.

[26] Walid Bechkit, Yacine Challal, Abdelmadjid Bouabdallah, and Vahid Tarokh. A Highly Scalable Key Pre-Distribution Scheme for Wireless Sensor Networks. *IEEE Transactions on Wireless Communications*, 12(2):948–959, 2013.

[27] Chih-Kuang Lin, V. Zadorozhny, P. Krishnamurthy, Ho-Hyun Park, and Chan-Gun Lee. A distributed and scalable time slot allocation protocol for wireless sensor networks. *Mobile Computing, IEEE Transactions on*, 10(4):505–518, April 2011. ISSN 1536-1233.

[28] Mathieu Lacage and Thomas R. Henderson. Yet another network simulator. In *Proceeding from the 2006 Workshop on Ns-2: The IP Network Simulator*, WNS2 '06, New York, NY, USA, 2006. ACM. ISBN 1-59593-508-8.

[29] Maryam Amiri-Nezhad, Manel Guerrero-Zapata, Boris Bellalta, and Llorenç Cerdà-Alabern. Simulation of multi-radio multi-channel 802.11-based mesh networks in ns-3. *EURASIP Journal on Wireless Communications and Networking*, 2014(1): 118, 2014.

[30] A. Bachir, F. Theoleyre, A. Duda, and K.K. Leung. Energy-efficient broadcasts in wireless sensor networks with multiple virtual channels. In *Wireless Communications and Networking Conference (WCNC), 2010 IEEE*, pages 1–6, April 2010.

[31] F. Theoleyre, A. Bachir, N. Chakchouk, A. Duda, and K.K. Leung. Energy efficient network structure for synchronous preamble sampling in wireless sensor networks. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1–6, May 2010.

[32] Wei Ye, Fabio Silva, and John Heidemann. Ultra-low duty cycle mac with scheduled channel polling. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, SenSys '06, pages 321–334, New York, NY, USA, 2006. ACM. ISBN 1-59593-343-3.

[33] M. Kuorilehto, M. Kohvakka, J. Suhonen, P. Hämäläinen, M. Hännikäinen, and T.D. Hamalainen. *Ultra-Low Energy Wireless Sensor Networks in Practice: Theory, Realization and Deployment*. Wiley, 2007. ISBN 9780470516799.

[34] Tomas Nilson. Some matters of great balance. Master's thesis, Mid Sweden University, 2013.

[35] Lei Gao. Latin squares in experimental design. Technical report, Michigan State University, December 2005.

[36] Beshr Al Nahas, Simon Duquennoy, Venkatraman Iyer, and Thiemo Voigt. Low-power listening goes multi-channel. In *Proceedings of the 2014 IEEE International Conference on Distributed Computing in Sensor Systems*, DCOSS '14, pages 2–9, Washington, DC, USA, 2014. IEEE Computer Society. ISBN 978-1-4799-4618-1.

[37] A.S. Hedayat, N.J.A. Sloane, and J. Stufken. *Orthogonal Arrays: Theory and Applications*. Springer Series in Statistics. Springer New York, 1999. ISBN 9781461214786.

[38] Shaohua Qin, Dongyan Chen, Xu Huang, and Leilei Yu. Rtmac a random time slot multi-channel mac protocol. *Journal of Computational Information Systems*, 2014.

[39] https://www.nsnam.org/docs/release/3.21/models/html/index.html. Last updated on Dec 05, 2014.

[40] https://www.nsnam.org/docs/release/3.21/tutorial/singlehtml/index.html. Last updated on Dec 05, 2014.

[41] Ted Herman and Sébastien Tixeuil. A distributed tdma slot assignment algorithm for wireless sensor networks. In SotirisE. Nikoletseas and JoséD.P. Rolim, editors, *Algorithmic Aspects of Wireless Sensor Networks*, volume 3121 of *Lecture Notes in Computer Science*, pages 45–58. Springer Berlin Heidelberg, 2004. ISBN 978-3-540-22476-1.