Master of Science

EFFICIENT BROADCASTING METHOD OF SCHEDULE TABLE FOR A CENTRALIZED SCHEDULING ALGORITHM IN IEEE802.15.4E TSCH

The Graduate School of the University of Ulsan

Department of Electrical and Computer Engineering

RUGAMBA Jules Pascal Gilles

EFFICIENT BROADCASTING METHOD OF SCHEDULE TABLE FOR A CENTRALIZED SCHEDULING ALGORITHM IN IEEE802.15.4E TSCH

Supervisor: Professor KIM Myung Kyun, Ph.D

A Thesis

Submitted to

The Graduate School of the University of Ulsan

In partial Fulfillment of the Requirements for the Degree of

Master of Science

By

RUGAMBA Jules Pascal Gilles

Department of Electrical and Computer Engineering

University of Ulsan, Korea

February 2020

DEDICATED TO

My Parents, My twin Sisters,

Those who find satisfaction in the fulfillment of this work

EFFICIENT BROADCASTING METHOD OF SCHEDULE TABLE FOR A CENTRALIZED SCHEDULING ALGORITHM IN IEEE802.15.4E TSCH

This certifies that this Thesis of RUGAMBA Jules Pascal Gilles is approved by:

Committee Chair: Professor Seok Hoon Yoon

Committee Member: Professor Insoo Koo

Committee Member: Professor Myung Kyun Kim

Department of Electrical and Computer Engineering

University of Ulsan, Korea

February 2020

ACKNOWLEDGEMENT

I would like to express my gratitude to the Korean People through the National Institute for International Education (NIIED) by Korean Government for giving me the opportunity to benefit from a scholarship to pursue my studies in the Republic of Korea and improve my skills and knowledge.

I would like also to deeply thank my advisor Prof. KIM Myung Kyun for his paralleled and continuous support during my study period. His tireless desire to inculcate the knowledge was more than helpful throughout my period of study.

I also would like to thank all the lab members of CNLAB at UoU for support each other and a great collaboration in whatever circumstances we were working through.

Last but not least, I express my special thanks to my parents, my sisters for their kind support, love and sacrifice to be what I am**.**

Republic of Korea, Ulsan, November 2019

RUGAMBA Jules Pascal Gilles

요약

무선 매체를 통한 통신은 상당히 발전하는 기술 세계에서 중요한 위치를 차지합니다. 거의 모든 분야에서 무선 기술은 산업 분야와 같은 다양한 흥미로운 응용 분야를 떠올리는 중요한 기반을 가지고 있습니다. 무선 네트워크는 무게 감소, 전선이없는 자유, 유연성 및 시스템 유지 관리 용이성과 같은 이점을 제공하기 때문입니다. 이러한 이점은 환경 모니터링, 스마트 홈 및 기타 많은 것들과 같은 실제 세계에서 볼 수 있습니다.

사물 인터넷에 사용되는 다양한 유형의 무선 기술이 있습니다. Bluetooth, WI-FI 및 GSM 외에도 MAC 프로토콜 인 IEEE 802.15.4e TSCH (Time Slotted Channel Hopping)는 결정 성있는 저전력 메시 네트워킹을 지원하기 위해 실제 확장 가능한 사물 인터넷 (IoT)의 중요한 표준입니다. WSN (Wireless Sensor Network)은 필수 불가결하고 일단 산업 응용 프로그램에 배포되면 일정에 결정적이고 긴 지연 시간이 필요합니다. TSCH 에서의 스케줄링은 중앙 집중식 또는 분산 될 수 있습니다. 중앙 집중식 스케줄링에서, 중앙 엔티티 또는 조정자는 특정 방식 또는 알고리즘에 기초하여 타임 슬롯 및 채널과 같은 통신 자원을 계산하고 모든 노드에 할당한다. 결국, 해당 스케줄 테이블은 네트워크의 각 노드가 알아야합니다.

우리의 작업은 IPv6 디바이스가 스케줄링 서버로도 작동하는 Raspberry Pi 에 구현 된 6LBR 을 통해 무선 센서 네트워크를 충족하는 글로벌 환경에서 TSCH 중앙 집중식 스케줄링으로 스케줄 테이블을 브로드 캐스트하는 효율적인 방법을 설명합니다. Contiki OS 에서 Path Conflict free Least Laxity First (PC-LLF) 알고리즘 [10]으로 알려진 IEEE 802.15.4e TSCH 기반 산업용 저전력 무선 네트워크를위한 중앙 집중식 링크 스케줄링을 구현하여 스케줄 테이블을 선택했습니다. 무선 센서 용 운영 체제. 패킷 충돌 및 손실을 줄이기 위해 스케줄 테이블 정보 패킷을 모든 전송 생성기 (노드)에 브로드 캐스트하는 방법을 구현하여 모든 데이터 전송이 스케줄에 지정된대로 정확하게 수행되도록합니다. 우리는 방법을 사용할 때와 맞춤형 방법 인 CoAP 및 OCARI 브로드 캐스트 방법과 같은 다른 방법을 사용할 때 모든 노드에 일정 테이블 정보를 전달하는 데 필요한 패킷 메시지 수를 비교했습니다. 구현시 네트워크 토폴로지 구성을위한 RPL 프로토콜 사용을 고려합니다. 사용 된 하드웨어 장치는 Texas Instruments CC 2650 런치 패드입니다.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

Chapter 1: INTRODUCTION

1.1. General Context

In the increasingly evolving world of technology, communication over wireless medium has become a very popular choice. The evolution through the year of the wireless technology give a tremendous number of possible applications of such technology in different field of life such as smart cities, environmental and industrial monitoring, health care and many more. As an example, in automotive domain, cars are becoming smarter or if we look further, they are becoming autonomous. For that a car needs to have some sensors and actuators and according to some survey, a single automobile may contain almost 100 sensors [1] which obviously are increasing in modern cars. These sensors support different applications such as breaking, speeding, etc. and their data communication is done through some in-vehicle communications systems which are usually a bus-based system or protocol such as FlexRay, or Control Area Network (CAN) [2, 3]. All the mentioned use wired communication. The emergency of innovative application and the evolution of market needs leads to the increase of Electronic Control Unit in one vehicle, which implies more wires to connect them and thus a complex and heavy wired communication, which increase vehicle cost as well as fuel consumption.

The complexity in wired communication can be found in any other field such as smart cities, environmental monitoring and especially in the industrial area where the evolution of smart industries implies indisputably wireless communication.

The use of Wireless Sensor Networks (WSNs) in industrial area have been increasing and it is found in different industrial application such as monitoring applications, radiation check, leakage detection, distributed and process control [4]. Those real-time applications are sensitive to the delays and demand high communication reliability as well as a satisfaction of a considerable scalability. The end-to-end delay are constrained by upper bounds which varies according to the field of application; tens of milliseconds for discrete manufacturing, seconds for process control and minutes for asset monitoring [5]. To overcome those challenges, different wireless technology approaches which reduces the cost of wired installation, have been considered.

A wireless solution based on the IEEE802.15.4 [6] standard is used in different applications with devices that require low power consumption. The development of this standard took in consideration the

requirements of Wireless Personal Area Networks (WPAN) either in home or industrial area. It operates in the 2.4GHz ISM frequency band and supports various network topology such as star and mesh.

Designed for constrained devices either in power consumption or in latency, the IEEE 802.15.4 standard may not deliver the desired low-latency and high reliability requirements as seen in its limit in [7] to satisfy real-time applications. An improved version has been defined as the IEEE802.15.4e. This new standard proposes different MAC protocol between others the protocol named Time Slotted Channel Hopping (TSCH). With the time slotted and multi-channel frequency hopping features, TSCH reduces latency and ensures high resistance against interferences as well as multi path fading. The IETF working Group 6TISCH [8] has been working on the standardized mechanisms to run IPv6 on top of the TSCH. In the 6TiSCH architecture, low-power wireless devices build a multi-hop Low power and Lossy Networks (LLN), which will be connected to Internet through one or more LLN Border routers [9].

Although the IEEE802.15.4e standard defines the process to execute communication schedule, it does not define the process of how the schedule is built, maintained, updated as well as the design of the entity in charge of performing such tasks. Scheduling on TSCH networks has been studied widely. Two types can be distinguished: 1. *Distributed scheduling* where there is no central manager (or coordinator) to handle the schedules but nodes in the network agree on the schedule to be used; 2. *Centralized scheduling* where a scheduling is built by a central entity or Coordinator by collecting the network information. Our implementation will be based on a realistic centralized scheduling algorithm.

Path Conflict free Least Laxity First (PC-LLF) [10], the algorithm we considered in our work, is a centralized scheduling algorithm which assigns priority to each packet transmission dynamically based on its laxity (the remaining time to the end-to-end deadline) and the amount of latency imposed by the collisions throughout its path to the final destination node (root node).

With the aim to give a contribution to the study of TSCH networks in real environment, we implemented the centralizing scheduling algorithm PC-LLF in real hardware and also implemented a method to send the schedule information from the scheduling server to each node in the network.

In order to provide open-source implementation of a fully finished protocol stack based on Internet of Things standards, different projects have been initiated such as TinyOS, Contiki OS and OpenWSN. They support different variety of software and hardware platforms. Our focus was brought to Contiki OS and its features of the IEEE802.15.4e (TSCH) protocol.

1.2. Problem statement

In any kind of Wireless Sensor Network (WSN) in which a node has some sensed data, the main purpose is that all the data from each node must be collected by a central entity (or node) to be processed locally or by an external entity such as an application on cloud server, save data into a database, etc. For a better data collection, communication within the network formed by the nodes must be reliable enough to provide the good quality of service (QoS). The response to such requirement can be the defined IEEE 802.15.4e standard with its TSCH MAC protocol.

As in IEEE 802.15.4e standard does not specify the how to create or maintain a TSCH schedule, different research had been focusing on the scheduling issue either it is centralized or distributed scheduling. In centralized algorithm, the schedule is done by a central entity that has to gather all the network information such as for each link, the receiver and sender information. After computation and making a final schedule based on some specified algorithm (PC-LLF for our case), the central entity must send back to all node so that each node can know its assigned timeslot and channel to send or receive data. For that, we must avoid the loss of such essential packets to prevent any wrong data transmission due to schedule information.

In this work, as seen in Figure 1, we worked in a global IoT environment where through its coordinator a WSN entity is connected to a scheduler server (Raspberry Pi) that can also work as router (6LBR) for the external world (IPv6) to reach the WSN entity. In that environment, after network construction (Figure 1 (1)), collection to the Raspberry Pi of all information needed to execute the scheduling algorithm (Figure 1 (2)) and after the computation of the schedule table by the scheduling server (Figure 1 (3)), we aim to propose a method that ensures the reception of the schedule table by all nodes from the Raspberry Pi (Figure 1 (4)) so that the data transmission can follow the specified schedule by the central entity or coordinator.

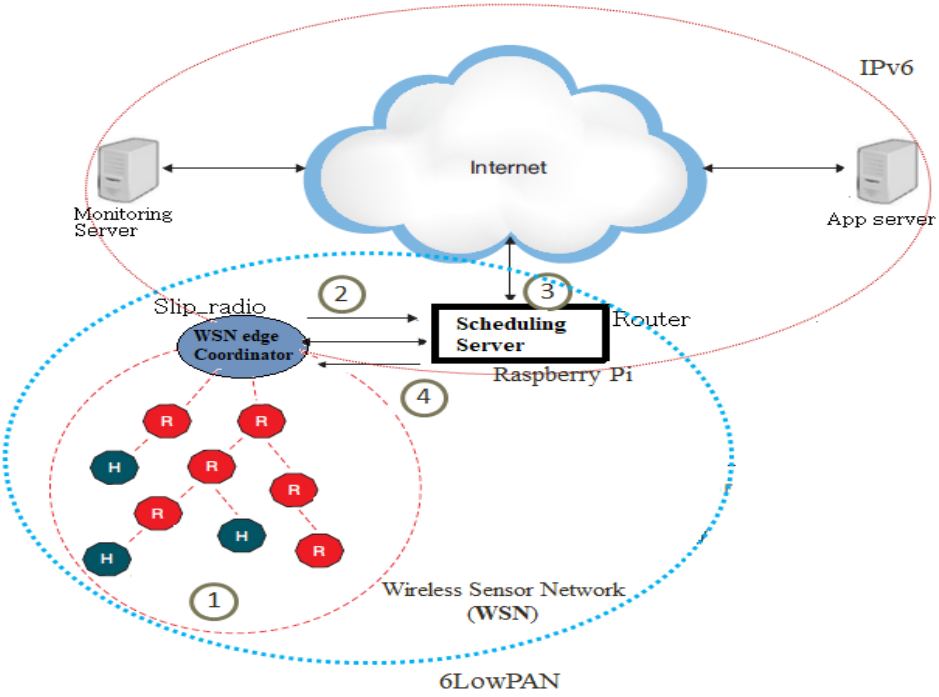


Figure 1 Bigger Picture

1.3. Contribution

Following the discussion above, the thesis work aims to:

1. Based on the network, designing and implementing a method that ensures the network information needed by the coordinator to make a schedule table for all nodes and to make sure all node can receive correctly the schedule table so that the data transmission is done correctly.
2. Evaluating the method by comparing with other existing one to make sure that it is working perfectly, and all nodes can receive the schedule table in real-time.

1.4. Thesis Outline

The remainder of this thesis is organized as follows. Some backgrounds are described in Chapter 2. An overview on certain related works will be given in chapter 3. A brief description of the PC-LLF is given in Chapter 4. In Chapter 5 the implementation process will be described as well as its evaluation. The conclusion will be given in Chapter 6.

Chapter 2: BACKGROUND

In this chapter, we give a background of the protocol or other standards that we used in our thesis. Thus, we provide between others a background on IEEE802.15.4 and its amendment. We discuss about TSCH standard and its mechanism. We provide also a background on RPL which is a routing protocol for WSN and briefly discuss about Contiki OS.

2.1. Wireless Sensor Network

2.1.1. Wireless Sensor

The wireless sensors are devices with resources constrained equipped with a sensor that are place in a desired area with a certain purpose or target to achieve such as monitoring, etc. They are used to collect data based on a specified event or states and those data are forwarded to a central entity such as coordinator or a gateway to be processed. Those devices sensors can be actuator and apply the receive actuation data to a relevant device.

For communication, they use a wireless protocol which can be chosen based on the application requirements as well as the desired standard for wireless communication. Wireless sensors are popular for sensing and data collection which make an improvement in the fulfillment of tasks such as patients monitoring, smart cities, monitoring of utilities such as water or energy. Their low cost for system deployment as well as their flexibility give them a good position in solution providing that's why they can be found in almost every part of our daily life.

2.1.2. Wireless Network

Wireless Network is network formed by deployed wireless sensors. Within the network, the devices can use a variety of wireless communication protocol standard. The appropriate protocol for a certain network is chosen based on applications requirements such as bandwidth, reliability, power consumption, etc. One of the most chosen protocol is IEEE 802.15.4 due to its low communication overhead and some other advantages such as low cost, low energy consumption, etc.

2.1.3. Terminologies

2.1.3.1. Wireless nodes

The nodes or wireless nodes are wireless sensor in the built network. The node can be used to collect data, sensing data or for routing or relaying data between a source node and the destination nodes.

2.1.3.2. Wireless link

Opposite to the wired communication where a wire ensures the communication between two nodes, a wireless link is a free-wire communication between two nodes. Each node has a transceiver equipped with one or more antenna which is used by the node to communicate in either reception or transmission mode separately.

Between two nodes, a wireless link is established when both nodes are in communication range of each other and are the same frequency.

2.1.3.3. Network Topology

Network topology is the disposition of multiples wireless nodes that are deployed in a certain environment to achieve a given purpose. The network topology shows or give an overview of the network communication link, the end devices and the central entity or coordinator (sometimes gateway).

2.2. IEEE802.15.4e

In the increasing domain of IoT, there exist different variety of vendors whom develop different kind of solution based on a standardized protocol. One of the most chosen these days is the IEEE802.15.4e with TSCH as MAC protocol which is an amendment of the previously existing IEEE802.15.4. That protocol was designed for Low-Rate Wireless Personal Area Network (LR-WPAN) where applications use low data throughput and devices are constrained.

With the advent of technology and the increase of various applications demands in terms of low power consumption, latency and as well reliability, the IEEE802.15.4 PHY and MAC layer introduced in 2003 was no longer satisfying the requirements from various entity due to its lack of channel hopping which implies multiple path fading and its energy consumption because the nodes using that standard have to

keep their radio on so they can receive or transmit at their own time. To overcome those issue and meet the market or industrial requirements, a new working group was created in 2008 to suggest innovation in the existing standard. And thus, came out in 2012 the IEEE802.15.4e [11]. The major contribution of this new amendment is the Time Slotted Channel Hopping (TSCH) which brought solutions to lack of the previous one.

2.2.1. Timeslots and Slotframe

The IEEE802.15.4e TSCH proposes a *slotframe* which is sliced into multiple and equal slots of time that are called *timeslot.* Thus, multiple timeslot composes one slotframe. For each timeslot, it is assigned a destination and a source node where the source node wakes up at that specific timeslot to transmit whereas the destination node wakes up to receive.

The timeslot length is long enough to transmit a packet of maximum payload size and to receive its acknowledgement. The timeslot length or duration is by default 10 ms, but it is adjustable for less or more but up to 15 ms. and that depends on the applications requirements as well as the used devices. If a packet is dropped in a certain timeslot, the retransmission will occur in the same timeslot of next slotframe. Based on the standard specification, one network can be with one or more slotframe with a different number of timeslots which is helpful to handle the bandwidth or delay requirements. Based on a schedule, a node is set in transmitting, reception or sleeping mode at a specific timeslot.

A timeslot in a slotframe can be dedicated or shared i.e. multiple links transmission reception can occur in same timeslot and same channel while only one link can occur in dedicated timeslot. A node which is receiving in a timeslot cannot transmit in the same timeslot. Same case for transmitter node, it cannot receive while transmitting. Those cases are considered as *Duplex Conflict Transmission*.

2.2.2. Enhanced Beacon (EB)

Enhanced Beacon or EB are specific beacons which is transmitted by a PAN coordinator at regular intervals with the aim of advertising the network. An EB packet is differentiated with other packets by looking at the frame version field of each packet transmitted in the IEEE802.15.4e TSCH.

The EBs carry crucial information for the network such as the applied schedule to the network, Absolute Slot Number (ASN) information, the slotframe number as well as the slotframe size, etc. Any new node that desires to join the network must follow the information in the received EB i.e. a node can only join the network after reception of an EB packet.

2.2.3. Absolut Slot Number (ASN)

Absolut Slot Number or ASN is a unique identification number marked to each slot. It is used to follow the network and timeslot specially to know the current timeslot which is used. The ASN helps also for the node's synchronization. The ASN is also considered as the elapsed number of timeslots which can be used as the slot counter from the start of the network and it increments globally in the network on the beginning of each time slot.

In the figure 2 we can see the operation of each timeslot in the slotframe. In each timeslot where it is assigned a sender and a receiver, the sender begins the transmission TsTxOffset after the beginning of timeslot, and the receiver listens to the channel guardTime before. Thereby, it implies that the devices cannot be desynchronized more than guardtime μs.

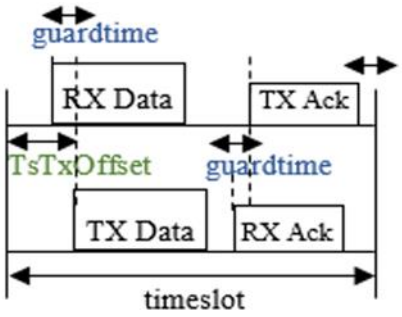


Figure 2 Timeslot operation in slotframe

2.2.4. Channel Hopping

In the IEEE802.15.4e TSCH protocol standard, at 2.4GHz frequency, 16 or less different channels can be used for communication. Each used channel is identified by a channel offset. A timeslot can contain one or multiple channel (<= 16) and each transmission is assigned a channel from the available channel list. The available channel list is called the hopping sequence list.

In the figure 3, we can see an example of assignment timeslot and channel given a certain network. We can see that two transmissions can be assigned at two different channels in the same timeslot as long as they are not in duplex conflict and also one transmission can be assigned two different timeslots within the same slotframe. The assignment depends on the chosen scheduling process.

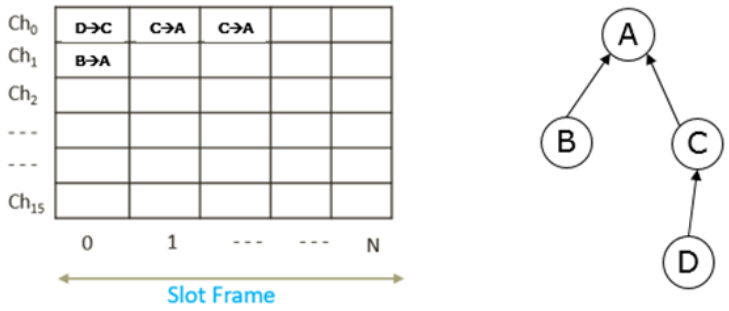


Figure 3 Example of Timeslot and Channel Assignment for a certain Network

2.2.5. Devices

Based on the amended IEEE802.15.4 protocol standard, they define two types of devices which can be used in any network running that protocol. Also, in the new amendment IEEE802.15.4e TSCH the two devices still in consideration.

2.2.5.1. Full Function Devices

A Full Function Devices or FFD is device that implements the whole protocol stack. The FFD can be either a Personal Area Network (PAN) coordinator accomplish all tasks done by the coordinator such as managing and/or initiating the network, etc. It can also act as a simple node or routing node.

2.2.5.2. Reduced Function Devices

As its name indicates, a Reduced Function Devices or RFD, is a device that implements limited sources and reduced protocol stack. It can be considered as a sensing only node device and cannot act as routing node neither a PAN coordinator.

2.2.6. Network formation

With the IEEE802.15.4e TSCH mostly any type of network topology can be formed, and it is supported by based on their configuration, they are placed in some categories such as: Tree, Star and mesh topology. Any kind of network topology must contain at least two nodes in which one is a coordinator.

2.2.6.1. Star Topology

A network with star topology is a network which contain just one PAN coordinator (FFD) and other end devices considered as sensing nodes which can be either an FFD or RFD. Any end device that wishes to communicate with another end device must first pass through the PAN coordinator as shown in Fig 4 (a).

2.2.6.2. Tree Topology

A network with tree topology is a network in which each node is associated to its parent node (i.e. parent-child association) except the PAN Coordinator. In that kind of topology as we can see in Fig 4 (c), the last node without any child node is called *leaf node.* A parent can be a routing node or also a sensing node. The communication is only between parent and child node.

2.2.6.3. Mesh Topology

A network with Mesh topology as shown in Fig 4 (b) is a network where nodes do not follow any specific configuration. The communication can happen between any devices in the network as long as they are in the range of each other. In Mesh topology, there exist one or more route from source node to destination.

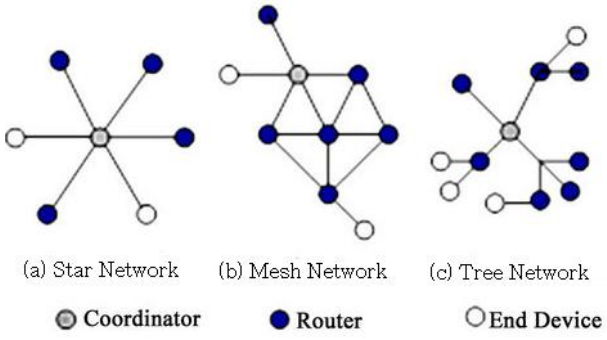


Figure 4 Network topology illustration

2.3. RPL: Routing Protocol for Low power Lossy Network

In IEEE802.15.43 TSCH, the nodes form a network as described above. The network can be of different form and formed in different way. To form a mesh network or any topology in TSCH, there exist different used protocol and the most widely used and the de-facto IPV6 routing for WSN is RPL [12] which is a routing protocol for Low power and Lossy network standardized by the IETF group and which was designed to meet some requirements described in [13], [14] such as latency, reliability, scalability among others. It is a tree-oriented routing protocol in which the routes are organized through the destination oriented directed acyclic graph (DODAG) rooted at the edge router.

It is used to get an accurate vision of the network status and thus to compute the best path from the root to all nodes or from each node to the root node. Those paths are computed based on an Objective Function (OF) and its set of metrics and constraints. In Contiki Operating System that we used for our implementation, the RPL implementation considers the Expected Transmission Count (ETX) [15] as default metrics.

In RPL, a certain number of acyclic DAG form the DODAG and each DAG has a root. Designed for IPv6 networks, RPL uses the features of ICMP6 neighbor discovery messages for its auto-configuration.

The building process of a DODAG is triggered by a root node and four types of control message are used between nodes during that process to exchange different kind of information as shown in Figure 5. Those messages are explained below.

1. ***DODAG Information Object (DIO):*** it is a broadcast message initiate by the DODAG root which trigger the construction on a new DAG. It can also be sent by other router nodes in the DODAG. The DIO message contains network information useful to the neighbor about the current RPL instance, the OF with corresponding metrics and constraints, the sender node rank.
2. ***DODAG Information Solicitation (DIS):*** it is a multicast message sent by a new node desirous to join the network for requesting a DIO from an RPL neighbor.
3. ***Destination Advertisement Object (DAO):*** used to enable source routing by sending reverse route information and propagating the information upward along with DODAG, it is sent by each node except the DODAG root to inform the routing tables with prefixes of their children and to advertise their own addresses as well as prefixes to their parent.
4. ***Destination Advertisement Object Acknowledgment (DAO-ACK):*** it is unicast packet sent by the DAO recipient to acknowledge the reception of the DAO packet

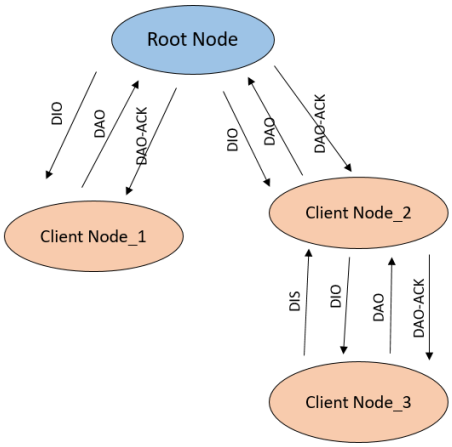


Figure 5 DODAG construction process

2.4. Contiki OS

Written in C language, Contiki operating system [16] was designed for resource constrained devices like Low-Power wireless IoT devices and includes in its latest version [17] the implementation of TSCH based on the IEEE 802.15.4e standards. Its library has a full TCP/IP stack for radio network communication, and it supports different variety of radio enabled platforms. From IETF (Internet Engineering Task Force) known networks and applications standards, Contiki supports in its library TCP, 6LowPAN, UDP, RPL, IPv6 and CoAP. Contiki OS includes an event driven kernel used to load and unload process at run time. Contiki application are based on processes which can be cooperative or preemptive processes [18]. A cooperative process continues execution until the end and cannot be interrupted by another cooperative process whereas the preemptive process can interrupt the currently running process and execute. The interrupted process will resume after the preemptive process finishes execution as we can see that from figure 6 at Time t_2 where process 2 was interrupted by a preemptive process and could resume after the execution of the last.

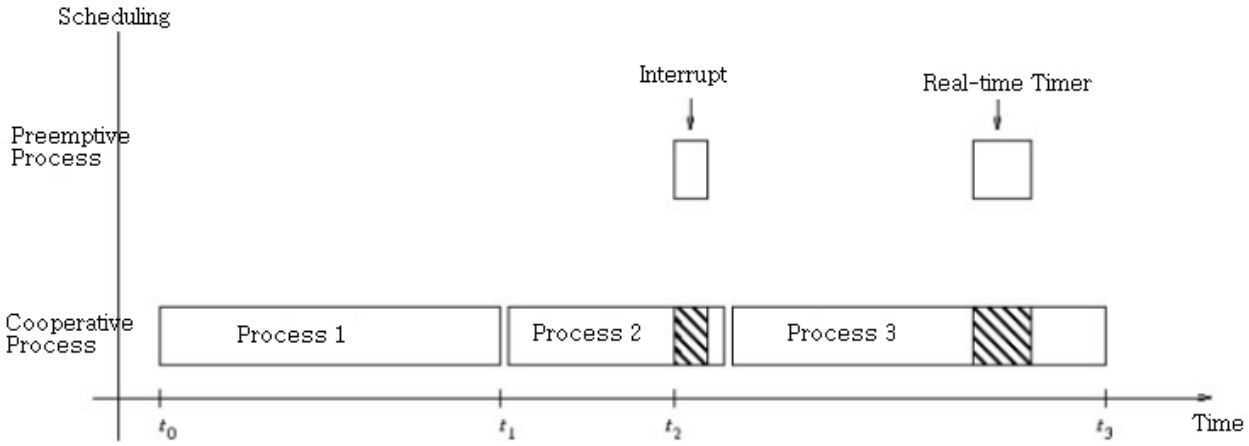


Figure 6 Contiki Process

Contiki has several timer libraries for user program and for the OS itself, such as timer, stimer, ctimer. It includes also etimer for scheduling an event to the processes and rtimer for scheduling real time tasks which was used to handle task from our real time centralized scheduling implementation. For memory allocation, Contiki proposes a "mmem" which is a managed memory allocator and "memb", the one used in our implementation, is a memory blocks with constant size and placed in static memory.

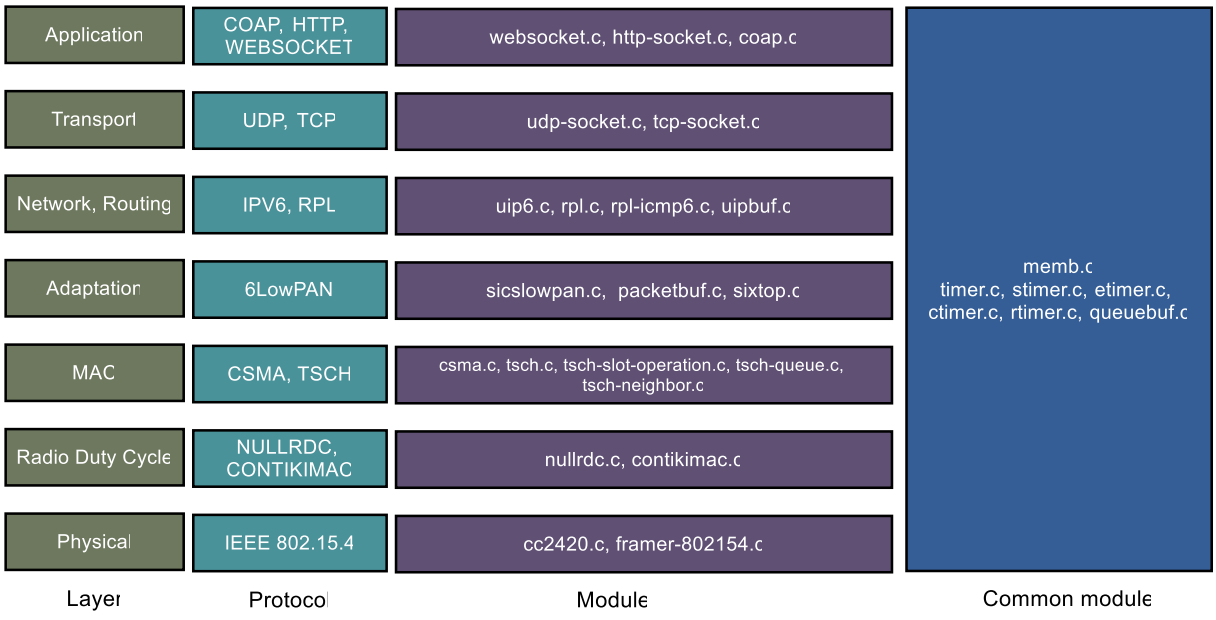


Figure 7 A Contiki OS Network stack

Contiki network stack includes diverse protocol at each layer. From Figure 7, we can see that there are some examples of modules specific to each protocol and some examples of common modules. The queuebuf.c module which is common for example provides a way to manage multiple packets at a time.

Each other module to access the queubuf.c needs to maintain pointers to it. 6LoWPAN at adaptation layer would take advantage of queuebuf.c for fragmenting IPv6 datagrams into multiple packets and TSCH at MAC layer use it for transmission queues. It includes also the 6TiSCH operation sublayer (6top) which provides an abstraction of an IP link over TSCH MAC and eases the cell negotiation dynamically with one-hop neighbors.

Chapter 3: RELATED WORKS

As mentioned before, the IEEE802.15.4e standard is relatively new concept and the standard itself does not specify how the schedule of a network based on TSCH should be computed, executed or maintained. From the publication of the IEEE802.15.4e standard, many researches have been oriented in the making different methods that would generate a reliable and suitable schedule taking in consideration the applications requirements. The schedules proposed are in two categories: *Centralized scheduling* and *Distributed scheduling.*

In centralized scheduling as described above, a central entity such as a PAN coordinator takes charge of the whole network and must get the information about the network topology (links between nodes) and all other information required to make schedule based on a specific schedule algorithm in order to compute the schedule to be applied in the network.

Most of the existing works, proposed only the study of the scheduling but do not provide an explicit overview of how the final scheduling table would be known by all the nodes in the network i.e. how the PAN coordinator broadcast the schedule table after computation so that all nodes can know their assigned cell or timeslot and channel offset.

Some centralized approach such as in [19] where the scheduling is done by dynamically prioritizing each sensor device or node based on its traffic generated, the authors developed the algorithm in Python language but did not specify how the PAN coordinator would broadcast to all nodes the final schedule table. Also, in [20] the scheduling algorithm assigns the timeslot/channel offset based on network topology and traffic load which implies some information to be sent to the PAN coordinator. Even though they developed their own ad-hoc simulator in Python and used it based on their algorithm, they did not specify which method the PAN Coordinator in the simulator would use to broadcast the final schedule to all nodes.

On the other side, in [21] they took in consideration for their main work, the cost to install or update a schedule to all nodes by a central scheduler. They used two approaches to install the schedule *CoAP* and *OCARI* a custom protocol. By using CoAP, they considered three approaches. In the first named “Single”, the central scheduler issues a separate confirmable *CoAP POST* message to write each field of each cell it installs into a node. To do so, they assume that for any network node a route to reach it from the scheduler node is known by all nodes within the route which make all nodes to keep a huge amount of network information while they are constrained devices. The second one called “PATCH”, the central scheduler

only sends the cell that must be updated. This one would not resolve the issue of broadcasting because it does not send the whole schedule table, it only considers update which means that a certain method has to be used in advance for sending the schedule table as a whole. The third is “Broadcast” where the central scheduler broadcast the complete schedule table over CoAP. Even though it ensures that each node receives the schedule by using *CoAP Observe*, it uses many packets to transmit the schedule because of the maximum payload must be respected. This implies a long time in broadcasting i.e all devices keep their battery on to receive the whole schedule table.

With the custom-built OCARI, they intend to provide a lower bound on the number of packets used to broadcast all the schedule table. For that, when a new schedule is ready, it is all broadcasted by piggy-backing into the beacons sent by all parent nodes. The OCARI beacon has a free payload for schedule table equals to 80 bytes. To get that payload size, they implemented some changes to the OCARI standard such as the use of short MAC address, a more efficient cellId representation between others.

Broadcasting by CoAP implies more packets to be used than with OCARI because of the payload size available in each method. But our method uses less packets than all as it will be seen in the evaluation part.

Chapter 4: PC-LLF: PATH CONFLICT AWARE LEAST LAXITY FIRST ALGORITHM

As described above, two different kind of scheduling exist in the IEEE802.15.4e TSCH: *Centralized scheduling* and *Distributed scheduling*. The PC-LLF algorithm [10] is a centralized algorithm designed over IEEE 802.15.4e TSCH with the aim to reduce latency of messages by finding feasible schedule for all transmissions of each flow along their designated path. As described previously, the transmission in TSCH occurs at a certain time slot and channel, the two compose a cell (*t*, *ch*).

Designed for TSCH networks, PC-LLF algorithm obeys to some constraints of that technology described below. Firstly, network forms a topology mesh network (Figure 3) as a Directed Acyclic Graph (DAG) G = (N, L) where N= {N_1, N_2, …, Nn} represents all devices in the considered TSCH networks and arcs in L are communication link. *NpNq* represents a link where Np is the transmitter node and Nq is the receiver node. Links are considered as interfering when the sender or receiver of one link can overhear a sender's transmission from another link and thus the interfering links cannot be scheduled on the same communication cell. Also, the duplex conflicting links cannot share the same time slots. The duplex conflicting links are all the links that have same receiver or sender nodes. To show an example of duplex conflicting link, we considered the Figure 8. Based on that figure, duplex conflicting links of the link N_6N_2 are CNF (N_6N_2) = {N_5N_2, N_4N_2, N_2N_1}.

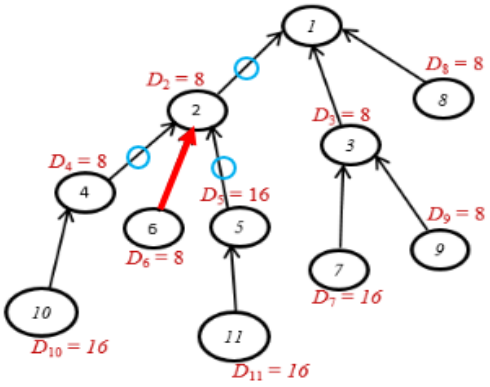


Figure 8 Example of network G = (N, L), N= {N_1… N_{11}*} and D the Period assigned to each transmission from its source node;* N_1 *is the root node*

All transmissions are upward and considered periodic and after each period, the node released a new packet to be transmitted. The period is assigned randomly to each node from a power of two value [2^3, 2^9]. The transmission of each packet must respect an end-to-end delay which is restricted to end-to-end deadlines and equals to the transmission period. A packet's transmission miss deadline when it arrives to the destination later than its release time added its deadline considered in time slot value.

To find a feasible schedule for a given network, the real time PC-LLF algorithm schedules based on calculation of priority and assigns firstly the communications resources to the transmission with the lowest priority value. The algorithm updates dynamically each transmission's priority on each time slot by considering two parameters: the remained time to the end-to-end deadline and the delay posed by the average amount of collisions that the packet may encounter at each remaining intermediate node. From that, a few collisions and longer deadline would imply a larger space for transmission to be scheduled whereas a high amount of collisions would make hard to schedule the transmission within the limited laxity and thus it corresponds to a high priority.

To calculate priority by PC-LLF, some parameters were considered such as a time window denoted *TW [x, y]* for each transmission which is the time phase from the earliest time slot (*x*) and the latest time slot (*y*) that the transmission can occur. The earliest time slot is the actual time slot and thus it is dynamically recomputed at each timeslot. The latest time slot has a fixed value and it depends on the deadline value. With that value we can make sure that each transmission meets the end-to-end deadline. The second considered parameter is the average delay due to collisions that each remaining transmission may deal with at the time slot S. The considered collisions are duplex conflicting collisions. The duplex conflicting collision happens due to a transmission existed in a conflicting link with a time window *TW* that overlaps with the time window of the actual considered transmission. Let denote the amount of conflict transmissions with a certain transmission T_i as $N_{CFL}(T_i)$. The next step is to get the average latency of a packet due to the collisions through the remaining path at a certain time slot. The remaining path is composed by the actual transmission and the future transmission of the packet in order to reach the root nodes. The average latency due to duplex conflicting transmission at a certain time slot S is denoted $D_{CFL}^{avg}(T_i, S)$ and its value change dynamically at each time slot because the amount of duplex conflicting transmissions varies since some transmissions may be scheduled previously, or others may be released later. Let denote *k*, the number of hops count away for a transmission to reach the root node. Thus, the average latency is given by the sum of all conflicting links divided by the number of hops count as in the equation below:

$$D_{CFL}^{avg}(T_i, S) = \sum N_{CFL}(T_i)/k \qquad (1)$$

From the equation (1), the priority of a transmission at a certain time slot S is computed by the difference between the time window of the packet's transmission and its average latency as described in the equation below:

$$Pr(Ti,S) = TW[x, y] - D_{CFL}^{avg}(T_i, S) \quad (2)$$

From equation (2), the lowest value corresponds to a higher priority. Based on the computed priorities for all transmissions, PC-LLF algorithm assign the time slot and channel to the transmission with the lowest value. When several transmissions have same priority, the transmission with the smallest length of time window will be assigned firstly. Furthermore, if the transmissions are executed in an environment with more than one channel, among the transmissions which are not in duplex conflict with the firstly assigned transmission, the transmission with highest priority is assigned to a different channel in the same time slot.

Chapter 5: IMPLEMENTATION AND EVALUATION

With the aim to run a network in real environment, the PC-LLF algorithm was implemented using Contiki OS into the low-power microcontroller of a Texas instruments TI CC2650 LaunchPad kit. Its microcontroller is based on ARM Cortex-M3 architecture which operates at 48MHz, with 128KB of flash memory and 20KB of SRAM.

5.1. Implementation process

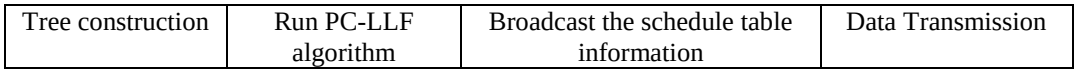


Figure 9 Implementation Process

Figure 9 gives a big picture of the implementation process divided into four parts. Firstly, is the tree construction where nodes make a network topology using the RPL routing protocol and the root node gathers all information related to the network and to each node. Secondly is the execution by root node of the implemented PC-LLF algorithm based on the previously built network topology and gathered information. At third, root node assigns to each node a timeslot to broadcast the schedule table information packet, broadcasts the resultant schedule table to all nodes and that part includes also the broadcast of the schedule table information packets. Fourthly all nodes will start data transmission in the exact cell i.e. timeslot and channel assigned by the PC-LLF scheduling algorithm.

5.1.1. Tree Construction

Once the nodes are deployed in a certain place, they must form a network in order to send their sensed data or receive any other kind of data. As said in the chapter 2, there exist different way to make network topology in WSN. We have chosen the RPL between others. RPL is an already implemented protocol in Contiki OS which is an operating system widely used in Wireless Sensor Network.

Contiki OS includes the RPL routing protocol in two modes:

1. ***Storing*** mode where each node in the network stores the information of the network especially those related to its children and parent node.

2. ***Non-storing*** mode where each node forwards the information to the PAN coordinator. All the information is stored and known by the PAN coordinator only.

We chose to use RPL non-storing mode as it allows the root node to gather all network information from each node in the network which is indispensable for execution of PC-LLF algorithm.

The nodes form a network by RPL non-storing mode by exchanging control messages. The nodes run Minimum Scheduling Function [22] in the tree construction period. The root nodes broadcast first the DIO packet to inform of the existence of a network and each receiver node try to join by following the configurations parameters in DIO message packet such as the used objective function (OF), rank of the sender, DAGid, mode of operation. After computation by considering OF, the joiner node chooses its preferred parent and join the network. The joiner node sends a DAO packet to its preferred parent node for joining and to inform that it has joined the network. Once joined, the node may also act as a router and broadcast the DIO at its turn.

Type: 0x0A	Option length	Period
1 byte	2 bytes	2 bytes

Figure 10 Structure of new Period Option in RPL DAO packet

In the DAO messages, there exist different DAO option used for carrying routing information such as *target option* which carry the information of DAGid of the new joiner node and the joiner node itself. The other is *transit option* which carry different information such as the joiner node's parent information and thus the links information are forwarded to the root node. To send from each node, more information needed by root node to run PC-LLF, a new option named *Period Option* as shown in Figure 10 was added in the DAO packet with the aim to forward to the root node the period value of each node's transmission. To apply it in Contiki, we modified *the rpl-icmp6.c* module which implement the handling of input/output for RPL control messages. The period option has three field: *Type* to identify the option type with value 0x0A, *Option length* to specify the length and *Period* which contains the period for the transmission.

As we use the RPL non-storing mode, all the networks information and topology details are known by the root node from the forwarded packets by each node. We assume that the number of nodes in network is fixed and known by root node in the beginning and thus, it can check through defined function if it had received all the information from all nodes in order to start running the PC-LLF algorithm.

5.1.2. Implementing the PC_LLF

For the PAN coordinator to run the PC-LLF algorithm, it compulsory considers the network information such as link, period, etc. From the tree construction period, all the packets with network link information and transmission's period are forwarded to the root node from each node via RPL control messages. The root node forwards all that information to the scheduler server which is Raspberry Pi in our case. To forward the information from root node to Raspberry Pi, we implemented the 6LBR protocol [24] which is a Routing protocol for Low Power Lossy Networks. The 6LBR implemented, allows the communication between the scheduler server and the WSN entity with one device acting as Slip-radio to ensure that communication as the two entity (i.e. the scheduler sever and the WSN have different radio standard).

The forwarded information from the Wireless Sensor Networks (WSN) to the Raspberry Pi contains the node's address, the node's parent which give the link and each transmission's period which is equal to its deadline. Based on all that information, the Raspberry Pi executes the PC-LLF algorithm to schedule all node's transmissions. In case of a non-schedulable network topology, the Raspberry Pi must stop the execution and the change in the topology is required.

The input parameter for the execution of algorithm are G the network topology with links information built from RPL previously, R which is the list of released transmission and in the beginning each node is considered to have a transmission to be forwarded. The next parameter is P, each node's transmission period which is equal to its deadline and N_{ch} the number of channels used in data transmission. We used two channels in our implementation (N_{ch} = 2) and for that in Contiki we considered the Channel offset with value 15 and 25 at index 0, 1 from the given 4 channels sequence length 4 option in module *tsch-conf.h*. The output is the schedule table of all transmissions (Fig.12.) which is forwarded to all nodes from root node.

The PC-LLF algorithm was written in C as it runs under Contiki OS and was added as new created library in Contiki. The library accesses the routing layer handled by RPL to get network topology with routing links and period of each transmission as input. For running the algorithm, the transmission structure was defined with node sender id, node receiver id, integer x and y to define the time window (TW) value, the transmission's priority, the transmission's deadline and the conflicting transmissions with the current one which can be defined from the routing links in the built network topology by RPL.

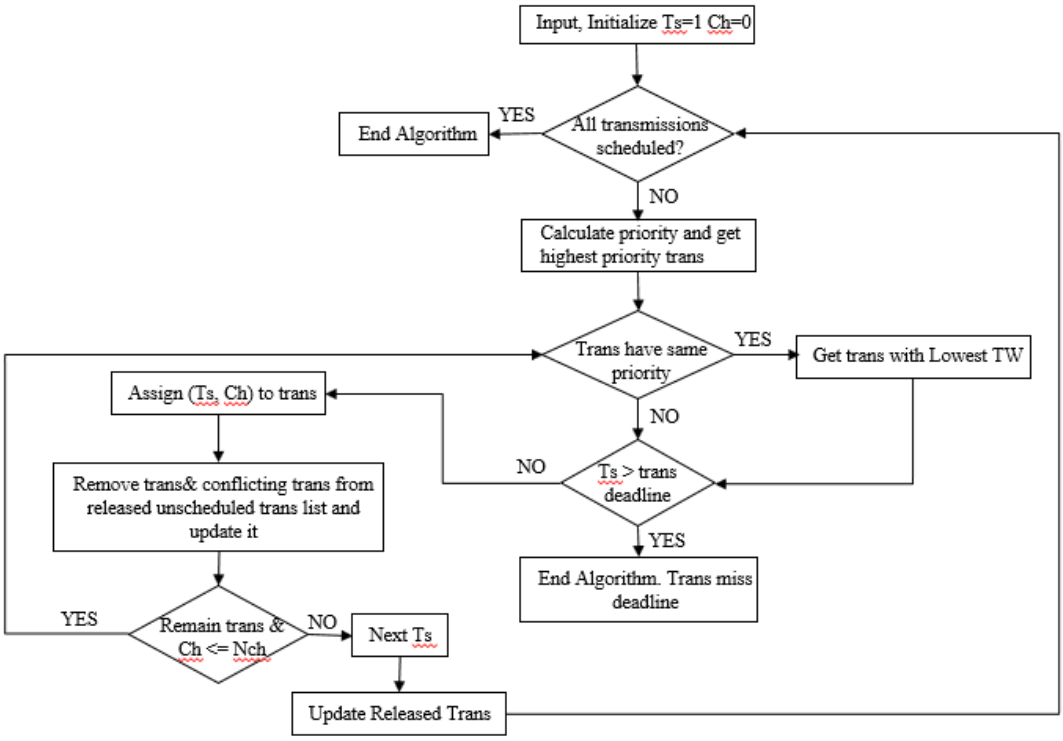


Figure 11 PC-LLF algorithm computation flow

Figure 11 gives the flow chart of computation of PC-LLF algorithm as described in chapter 4. In the flow chart, the timeslot is noted *Ts* and *ch* is the channel offset and N_{ch} is the total number of channels considered. The transmission is noted *trans* and time window as *TW*. The algorithm must make sure that all the transmissions are scheduled within the time limit defined by their deadline.

Timeslot	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Ch0	T(2-1)-1	T(4-2)-1	T(4-1)-1	T(6-2)-1	T(6-1)-1	T(11-2)-1	T(11-1)-1	T(10-2)-1	T(10-1)-1	T(2-1)-2	T(4-2)-2	T(4-1)-2	T(5-2)-1	T(5-1)-1	T(6-2)-2	T(6-1)-2
Ch1	T(9-3)-1	T(9-1)-1	T(11-5)-1	T(3-1)-1	T(10-4)-1	T(8-1)-1	T(7-2)-1	T(7-1)-1	T(9-3)-2		T(9-1)-2		T(3-1)-2		T(8-1)-2	

Figure 12 Example of scheduling table information from a network topology in Fig. 8

Figure 12 shows an example of cell assignment schedule table computed by PC-LLF based on the topology in figure 8.

The algorithm must finish to schedule all transmissions no longer than the Least Common Multiplier (LCM) of all periods value in the network. In the example above, the algorithm assigned cell to all transmissions within the limit timeslot defined by LCM (LCM=16 in the example). For example, at timeslot one (Ts=1, ch=0), the transmission from node 2 to root node (1) was assigned. The transmissions

are noted as $T\ (N_{id} - M_{id}) - I$ where N_{id} is the source node id of the packet, M_{id} is the receiver node id and I is the instance of the transmission which specifies the current period of the transmission, as all transmission are periodic. In the example (Fig.8) nodes N_2 and N_3 for example have two period of transmission within the LCM. The number of periods of transmission is defined by the LCM divided by the period value of the sender node's transmission.

5.1.3. Sending the schedule table information

This process is divided into multiple parts as the sending mechanism is considered by Broadcasting and Unicasting. Firstly, in Broadcasting, is the assignment of timeslot to each node to broadcast the schedule table information and then is to send the assignment information to all nodes. Last one is the broadcast of schedule table information packets by all nodes as assigned. Secondly, in Unicasting, we considered the Orchestra receiver-based Slot frame [25] and the second option is to send immediately to each node as through 6LBR the scheduler server knows each node's address and each path to the destination.

5.1.3.1. Assignment of timeslot to broadcast the schedule table information

After execution of PC-LLF algorithm, each node in the network must receive the schedule information. For that, the packets have to flow from the root node towards all the nodes in the network. To ensure the minimum loss of schedule table information packets, we assigned a timeslot to each node for broadcasting that packet. All the assigned timeslots are dedicated and thus only one node can transmit.

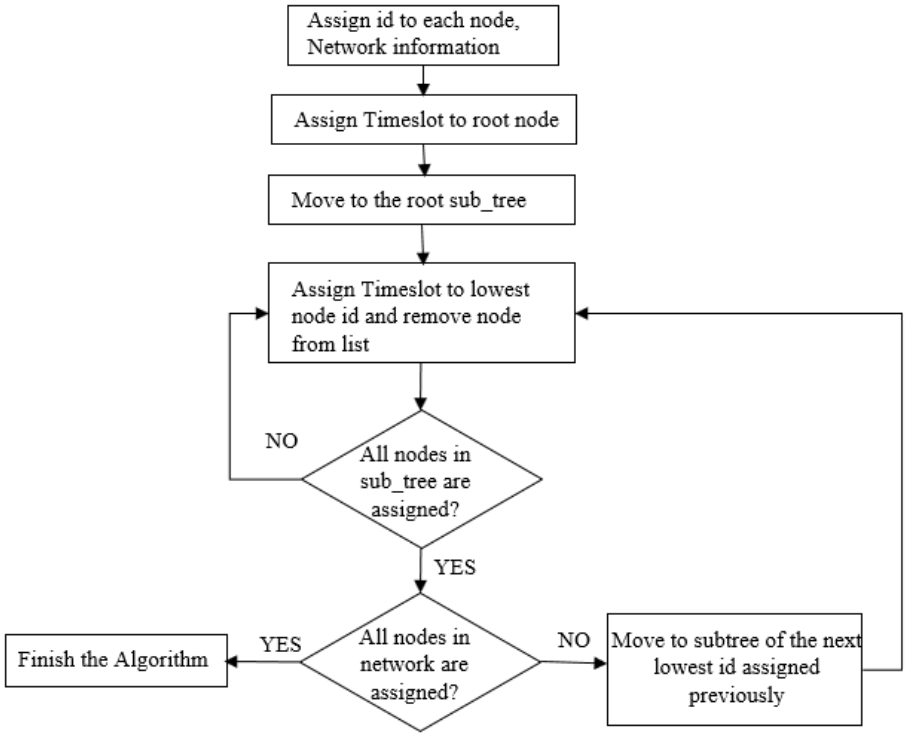


Figure 13 Flowchart of timeslot assignment for sending schedule table packet

Figure 13 shows the flowchart of timeslots assignment to each node for send the schedule table packet based on the network topology constructed by RPL in non-storing mode and the gathered network information by root node. The assignment was done by considering the Orchestra sender-based Slot frame as described in [25] in which the assignment was modified with Breadth-First Search Algorithm (BFS) [23] by considering each node_id per depth. Each node in the network is assigned a unique identifier (*node_id*) used to assign timeslot to the nodes to broadcast the schedule table information packet. By following the BFS standard, we assign firstly the master node (root node) in the first timeslot and then the algorithm traverses the tree from the adjacent node of the root node and assigns the next timeslot to the node with the smallest *node_id* value until all the adjacent nodes to the root nodes are assigned. At next hop-count, the algorithm firstly traverses the adjacent nodes to the firstly assign nodes from the adjacent node to the root and assign timeslot based on the value of *node_id.* The process continues until all nodes are assigned.

This assignment reduces the number of collisions firstly when transmitting because only one node is assigned for transmitting and there is no competition between nodes, and secondly when receiving the packet because one node (or more) can receive only from one transmitter, the assigned node.

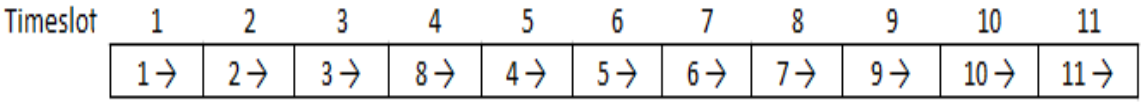


Figure 14 Result of assignment of Timeslot to each node in Fig.3 to send schedule table information

We can see from figure 14 the assignment results of network topology in Fig 8. The root node with node id equal 1 transmits in the first timeslot and then the node with *node_id* equal 2 is assigned the next timeslot (Ts=2) as it is the smallest *node_id* from the nodes adjacent to the root node and so on until each node is assigned its own timeslot to broadcast.

5.1.3.2. Sending the assignment information to all nodes

The assigned timeslots must be known by all node in the network and hence the root node will, after creating the assignment information packet, broadcast it to each node. To realize that, we used the Contiki OS implemented module named *tsch-queue* which defines a queue of outgoing packets for each neighbor and we added other queues neighbor beside the existing one, for the purpose of our implementation. The *n_assign* is used for queuing the assignment information packet when it is broadcasted and *n_data* for handling the outgoing data packet used specifically for data transmission period.

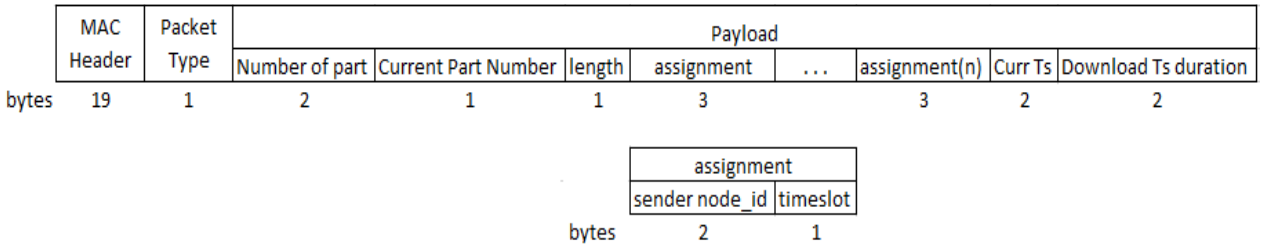


Figure 15 Frame format of Sch. Table Broadcast assignment information packet

The root node triggers the download of assignment information packet by broadcasting the packet. When a node receives the assignment packet, it activates its assignment information download period. Each node is assigned a number of timeslot delay between two transmissions. Each assignment information packet in node's queue is transmitted after the delay. A larger number of nodes in the network implies a large packet and hence it must be divided into parts to be transmitted. The use of *MSF* i.e. one timeslot and one channel offset as specified in the standard during that period may cause collisions in the packet traffic. To reduce the collision, we implemented a competition scheme within a timeslot which allow each node to access the communications resources by reducing contention time

based on the sequence of the assignment information packets received and by checking if the channel is idle or not.

When broadcasting the assignment information packet, we took in consideration the maximum packet size originally supported in TSCH which is 127kbytes. Fig15 shows the packet frame format and the size we used for the assignment information packet. Based on the frame format, one packet can include up to 30 assignment information which make a total size of 118kbytes. This allow to send a large number of assignment information packet while the allowed maximum size of packet is not exceeded.

Taking into consideration that the assignment information sent in one packet are considerable and that one assignment information contains only one node information, we assumed that the needed number of timeslot to send all packets in a given network and to be received by all nodes is equal to the number of nodes in the network multiply by two to avoid loss.

5.1.3.3. Sending the schedule table information

In order to transmit data packets, each node needs to know in which timeslot it must wake up for transmitting or receiving the data packet. That means the broadcast of the schedule table which is the result of the PC-LLF algorithm scheduling. To broadcast the schedule information packet, each node broadcasts in the assigned timeslot during the previous assignment process.

Based on the assignment in Fig14. and considering the Orchestra sender-based Slot Frame, we can see from Table 1 how the broadcast process will take effect during one Slot Frame in the network given in Fig8. The Slot Frame duration equals to eleven Timeslot (Timeslot =11).

Table 1: Broadcast process (network in Fig.8) with Orchestra sender-based Slot Frame

	Slot Frame										
Ts	1	2	3	4	5	6	7	8	9	10	11
N1	Tx										
N2	Rx	Tx									
N3	Rx		Tx								
N4		Rx			Tx						
N5		Rx				Tx					
N6		Rx									
N7			Rx								
N8	Rx										
N9			Rx								
N10					Rx						
N11						Rx					

From table 1, Ts represents the timeslot, Tx represents the transmission, Rx represents the receptions and N is the node and its id for example N1 is node 1 and N7 is node 7. Given the network in Fig8. and the assignment result in Fig14. we can see in Table 1 that node 1 (N1) will broadcast the schedule packet the first after receiving it from the scheduler server through 6LBR. When N1 broadcasts, N2, N3 and N8 go in reception mode to listen for a packet as they are direct children of N1. Same as when N2 broadcasts the N4, N5, N6 listen for the reception of the packet. In short, when it's a timeslot for a node N to transmit, its direct children wake up into reception mode during that timeslot. A node without any child node does not broadcast.

	MAC Header	Packet Type	Payload								
			LCM	Number of part	Current Part Number	length	Sched Info (1)	...	Sched Info (n)	Curr Ts	Download Ts duration
bytes	19	1	2	2	1	1	7		7	2	2

1<=n<=13

	Schedule information			
	sender address	receiver address	Timeslot	Channel
bytes	2	2	2	1

Figure 16 Schedule information packet frame format

When a node receives the schedule information packet, it adds the packet in the queue via *n_schedule* queue neighbor used for queuing the schedule table information packet. It is created through the *tsch_queue* module described above. After adding packet to the queue, the node has to process the received packet. For that, each node maintains two list RxList and TxList which are updated when processing the received packet i.e. it stores in RxList the information of its timeslot and channel in which it will wake up for receiving data and timeslot and channel for transmitting are stored in TxList. After processing the packet, the node gets the packet from the queue and broadcast at its turn in its assigned timeslot for broadcasting. Each node follows the same procedure.

Figure 16 shows the schedule information packet constitutes by *MAC header, packet type* to identify the packet. As a network can be made by many nodes, the full schedule information of such network cannot fit in one packet. Thus, the payload is divided into different fields to handle that. The packet frame format includes the *LCM* which is the least common multiplier for all period value of all transmission, *Number of part* which contains the number of packets needed to transmit all the schedule information packet, the *current part number* for identifying the currently being transmitted packet, *length* with the value of total transmission information in the current packet, *sched info* containing the assigned cell i.e. timeslot and channel offset for a transmission as well as the sender and receiver node address. It contains also the *curr Ts* which informs on the current used timeslot and the *Download Ts duration* which is the number of timeslots to be used for broadcasting the schedule information packet. Assuming a normal broadcasting and since each node has its own timeslot to broadcast the packet, we considered the maximum duration to broadcast the packet to be equal to the number of nodes multiply the number of packets used to transmit all the full packet (*Number of Part*). Based on the packet size and the maximum allowed size packet to be transmitted in TSCH (127kbytes), one packet can contain at most 13 scheduled transmission information which make the size equal to 121kbytes.

Beside the broadcast by Orchestra based Slot Frame, the second broadcasting method we considered is the broadcast using Minimum Scheduling Function (MSF) [22]. In this method as described in [22], all the nodes in the network share one timeslot and one channel which means that all nodes with a packet will try to transmit at same timeslot which trigger a channel clearing by all nodes to check if the channel is free to transmit. The winner i.e. the node that started transmitting will make its transmission and all other nodes go into reception mode.

Table 2: Broadcast process (network in Fig.8) with Minimum Scheduling Function (MSF)

	Slot Frame 1				Slot Frame 2				Slot Frame 3				Slot Frame 4			
Ts	1	-	-	11	1	-	-	11	1	-	-	11	1	-	-	11
N1	Tx				Rx				Rx				Rx			
N2	Rx				Tx				Rx				Rx			
N3	Rx				Rx				Tx				Rx			
N4	Rx				Rx				Rx				Rx			
N5	Rx				Rx				Rx				Rx			
N6	Rx				Rx				Rx				Rx			
N7	Rx				Rx				Rx				Rx			
N8	Rx				Rx				Rx				Tx			
N9	Rx				Rx				Rx				Rx			
N10	Rx				Rx				Rx				Rx			
N11	Rx				Rx				Rx				Rx			

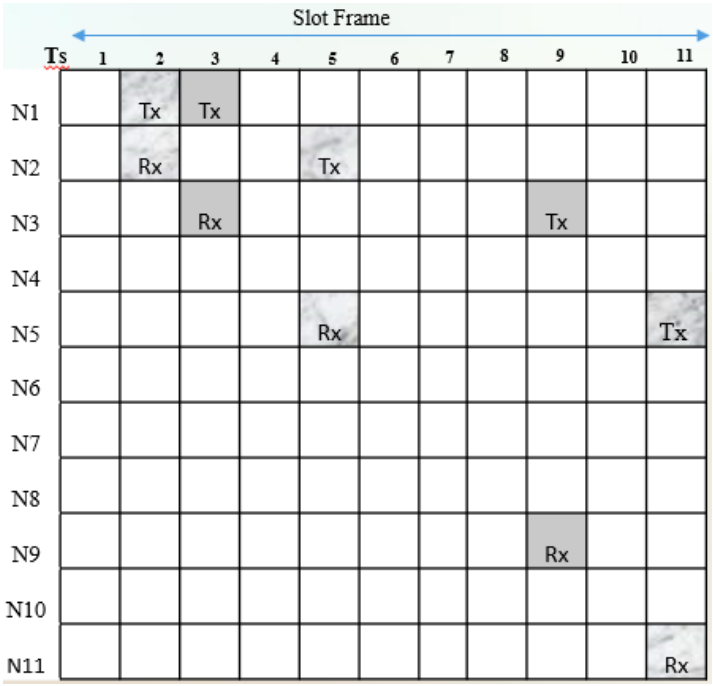
In table 2, Ts represents the timeslot, Tx represents the transmission, Rx represents the receptions and N is the node and its id for example N1 is node 1 and N7 is node 7. Table 2 shows the broadcast process by MSF during four Slot Frame (the process can have more) and each Slot Frame is eleven timeslot long. As all nodes use only one timeslot during one SlotFrame, the remaining timeslot in the Slot Frame will be idle.

In our case by considering the network in Fig8. Node 1 broadcasts first as it's the one to receive the packet the first from the scheduler server (Raspberry Pi). Thus, in the first Slot Frame at timeslot 1 Node 1 broadcasts the schedule table packet and all other nodes go into reception mode. In the second Slot Frame at time slot 1, Node 2 can broadcast the packet and all other nodes are in reception mode. In the next Slot Frame, Node 3 broadcasts and other nodes listen for reception. The process goes on and the transmitter in can be different in each Slot Frame because the winner can change. Using that method, the collision between nodes that desire to transmit are very high which can cause many packets dropping. As nodes must compete to transmit, it is not certain that given a certain time all nodes would have transmitted their packets and hence, some node's transmission can be over delayed.

As we are using the 6LBR for communication between WSN entity and the scheduler server, 6LBR allows the scheduler server to receive each node's address, parent and thus, link and other information which allow the scheduler server to know exactly each node location and path to each destination. With that benefit, we can send the schedule table packet by unicast by using the Orchestra receiver-based Slot Frame.

In the Orchestra receiver-based Slot Frame [25], the receiver node is assigned a timeslot to receiver (based also on node's id) a packet and its parent or the sender must use that same timeslot to transmit.

Table 3. Sending packet (network in Fig.8) by Unicast with Orchestra receiver-based Slot Frame



Based on the network in Fig8. Table 3 shows the process of sending the schedule packet by Unicast based on the Orchestra receiver-based Slot Frame. In Table 3 we just show an example of Flow from Node 1 to Node 11. Thus, Node 2 is assigned timeslot 2 for receiving and Node 1 will use that timeslot to send the packet. As Node 5 has been assigned to timeslot 5 for reception, its parent node, Node 2 use the same timeslot 5 for transmission. The process goes one until all the nodes are assigned and their parent nodes.

Sending the schedule packets by unicast may implies a long process and the use of plenty packets.

5.1.4. Data Transmission period

The data transmission is the period where all nodes transmit their data towards to the root node based on the schedule assign by the PC-LLF algorithm. Based on the schedule table information downloaded to all nodes during the download process, each node knows from the RxList and TxList exactly at which timeslot and channel it must wake up to transmit or to receive data packet. The Absolute slot Number (ASN) described above, gives the possibility to know exactly the current used timeslot and hence, each node can trace and know its timeslot to wake up.

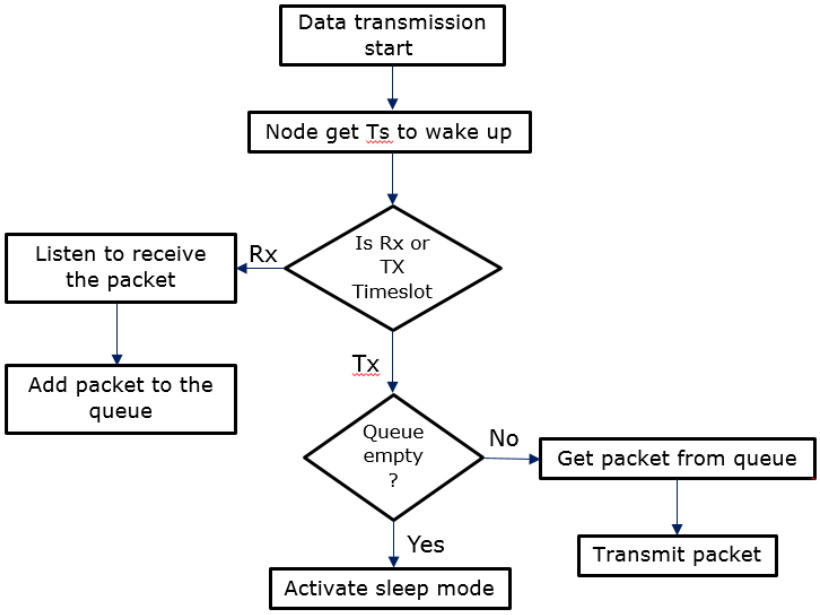


Figure 17 Data transmission period flow (Rx: Reception, Tx: Transmission)

The data transmission period duration i.e. number of timeslots for data transmission is equal to the LCM value of all node's periods in the network. The flow of data transmission is shown in Fig17. If the current timeslot is for reception mode, the node listen for the packet and after reception, it adds the packet in its queue and if is for transmission mode, the node firstly checks if any packet is available in the queue buffer, get the packet from the queue and send it; after transmission the packet is removed from the queue. In case the queue buffer is empty, it switches into sleep mode. Once the packet is ready, the 6LoWPAN of Adaptation Layer, passes it to the MAC layer for being transmitted. Likewise, when receiving packet, the MAC layer passes it to adaptation Layer using the *packetbuf.c* module. As each node wake up at the specific timeslot and channel for transmitting or receiving, the data packet transmission is done by broadcasting from its source node and only the node which woke up at that timeslot for reception can receive the packet. The data packet format contains the *MAC Header* field, the *Packet type* field for identification which differentiate with others packets and the *Payload* field which contains the data to be transmitted.

5.2. Implementation evaluation

The PC-LLF algorithm was implemented using Contiki OS in nodes of Texas Instruments TI CC2650 LaunchPad kit in which one acts as root node. The experiment environment and the devices used are details in Table 4 and Table 5.

Table 4. Specifications of experimental devices

Devices		
Raspberry Pi	CC2650 Launchpad	Computer
Quad Core 1.2GHz Broadcom BCM2837 64bit CPU	SimpleLink multi-standard 2.4 GHz ultra-low power wireless MCU	For Contiki OS as it ran in VMWare Ubuntu 16.04
1 GB RAM	Up to 48-MHz Clock Speed	Vmware in a PC with RAM 8Gb 64-bit OS 1T HDD
BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board	128KB of In-System Programmable Flash	Intel(R) Xeon (R) CPUE5-1620 0 @ 3.60GHz
100 Base Ethernet	8KB of SRAM for Cache	
40-pin extended GPIO	20KB of Ultralow-Leakage SRAM	
4 USB 2 ports	2-Pin CJTAG and JTAG Debugging	
Full size HDMI	Normal Operation: 1.8 to 3.8V	
	External Regulator Mode: 1.7 to 1.95V	

Table 5. Experimental environment information

Number of nodes	Location	
	Inside Building alley	**Outside Building**
5	20 m distance between nodes	100 m distance between nodes
10	20 m distance between nodes	50 m distance between nodes
15	-	50 m distance between nodes
20	-	50 m distance between nodes

Table 6. Table of each Flow and their period and delay time (in term of Timeslot).

Flow	Period	End-to-end Delay	Flow	Period	End-to-End Delay
2 → 1	*8*	1	7 → 1	*16*	8
3 → 1	*8*	4	8 → 1	*8*	6
4 → 1	*8*	3	9 → 1	*8*	2
5 → 1	*16*	14	10 → 1	*16*	9
6 → 1	*8*	5	11 → 1	*16*	7

Table 6 describes the period of each transmission which is equal to its deadline and that value is expressed in timeslot number. The timeslot is 15ms long and thus by multiplying each value in timeslot assigned by PC-LLF (from Fig12.) in column "*End-to-End Delay"* by 15ms, we get the end-to-end delay value in *ms*. for each flow (represented in first column with sender node id → destination node id), the value is less than its period value and thus all the flow didn't exceed the deadline for transmission (as period equals deadline) as it is seen in Table 6.

We compared the numbers of packets used by our broadcasting method comparing to those that can be used with CoAP or OCARI as specified in [21].

In Table 7, the *number of cells* column describes the existing cells in one schedule table and obviously increase when the number of nodes increases. Each cell is 7 bytes as shown in Fig.5.7. Our method allows at most 91 bytes over the 127 bytes max packet as the remaining is used for other system and configuration information.

Table 7. Number of packets required to broadcast the schedule packets.

Number of nodes	Number of cells	Our method (packets)	CoAP (packets)	OCARI (packets)
5	*12x7(84 bytes)*	1x5	3x5	2x5
10	*28x7(196 bytes)*	3x5	7x5	3x5
15	*60x7(420 bytes)*	5x5	14x5	6x5
20	*78x7(546 bytes)*	6x5	18x5	7x5

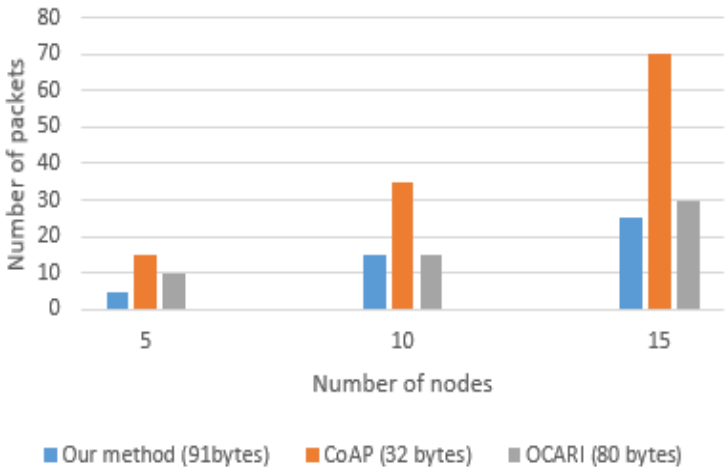


Figure 18 Number of packets used to broadcast the schedule table packet as a function of number of nodes

As described in [21], CoAP and OCARI allow respectively 32 bytes and 80 bytes. Considering the topology in Fig.8, the nodes that can broadcast the schedule packets i.e. with at least one children node, are 5 and thus we multiply by 5 to get the total number of packets to broadcast. As result, we can see from

Fig18. that our method uses less packets for broadcasting the schedule table, which has a positive impact on node's duty cycle as well as on the duration of broadcasting the schedule.

Chapter 6: CONCLUSION

In this thesis, an efficient broadcasting method of a schedule table for a centralized scheduling algorithm in the IEEE802.15.4E TSCH standard was presented and it successfully send to all nodes the schedule table information. We implemented also a heuristic scheduling approach named PC-LLF for TSCH networks using Contiki OS which is well known operating system in the IoT area. The PC-LLF component was implemented as a plus on the existing stack structure of Contiki OS. For constructing a network, we used RPL which is the most widely used routing protocol for Low power and Lossy networks. The communication between the scheduler server (Raspberry Pi) and the WSN entity was done by 6LBR with its slip-radio features

By comparing with some other method to broadcast schedule table such as CoAP or OCARI, we saw that our method not only successfully makes all nodes able to receive the schedule table computed by the PAN coordinator, but it also outperforms other methods either considering the delay or the number of packet used which of course both affect the performance of the network or the nodes in the network.

Based on the result, we believe that our method can be successfully applied to a network running a centralized scheduling algorithm for IEEE802.15.4e TSCH in real world which can be a non-negligible contribution in the IoT domain.

REFERENCES

1. W.J. Flemming. New automotive sensors-a review. IEEE sensors journal, pages 1900-1921, November 2008.
2. D. Paret. Multiplexed Networks for Embedded Systems: CAN, LIN, FlexRay, Safe-by-Wire. Wiley, May 2007.
3. S. Tuohy, M. Glavin, C. Hughes, E. Jones, M. Trivedi, and L. Kilmartin. Intra-vehicle networks: A review. IEEE Transactions on Intelligent Transportation Systems, pages 534{545, May 2014.
4. B. Sinopoli, C. Sharp, L. Schenato, S. Schaffert, and S. S. Sastry, "Distributed control applications within sensor networks," Proc. IEEE, vol. 91, no. 8, pp. 1235–1246, Aug. 2003
5. R. Zurawski, "Networked embedded systems: An overview," in Networked Embedded Systems, R. Zurawski, Ed. Boca Raton, FL: CRC Press, 2009, ch. 1, pp. 1.11–1.16.
6. IIEEE Standard for Local and Metropolitan Area Networks-Parts 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANS), IEEE Standard 802.15.4-2011, 2011, pp.1-314
7. G. Anastasi, M. Conti and M.Di Francesco. 2011. A comprehensive analysis of the MAC unreliability problem in IEEE 802.15.4 wireless sensor networks. IEEE Transactions on Industrial Informatics 7, 1 (2011), 52-65
8. P. Thubert (Ed.). An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4 - draft-ietf-6tisch-architecture10, June 2016. IETF Draftalwa, Editor, Magnetic Nanostructures, American Scientific Publishers, Los Angeles (2003)
9. P. Thubert, T. Watteyne, M. Palattella, X. Vilajosana, and Q. Wang "Ietf 6tisch: Combining ipv6 connectivity with industrial performance," in 7th Int. Conf. on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), July 2013, pp.541-546
10. A. Darbandi and M. K. Kim, "Path Collision-aware Real-time Link Scheduling for TSCH Wireless Networks", KSII TRANSACTIONS ON INTERNET AND INFORMATION SYSTEMS, September 2019.
11. IEEE standard for local and metropolitan area networks part 15.4: Low-rate wireless. IEEE 802.15.4e, page 1225, April 2012.
12. T. Winter, P.Thubert, A. Brandt, J.Hui, R. Kelsey, RPL: IPV6 "Routing protocol for Low Power and Lossy Networks", IETF request for comments 6550, March 2012
13. E.J. Martocci, P.D. Mil, N. Riou, W. Vermeylen, « Building automation routing requirements in Low-Power and Lossy Networks", IETF request for comments: 5867, March 2010
14. E.K. Pister, E.P. Thubert, S. Dwars, T. Phinney, "Industrial routing requirements in Low-Power and Lossy Networks", IETF request for comments:5673, March 2009

15. D. S. J. De Couto, D.Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In Proceedings of the 9th ACM International Conference on Mobile Computing and Networking (MobiCom), San Diego, California, September 2003
16. A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a lightweight and exible operating system for tiny networked sensors. 29th Annual IEEE international conference on Local Computer Networks, pages 455{462, November 2004. 17
17. Contiki 3.x. https://github.com/contiki-os/contiki.Online; accessed 2-June-2016. 17
18. Contiki processes. https://github.com/contiki-os/contiki/wiki/Processes. Online; accessed 2-June-2016. 17
19. M.R. Palattella, N. Accettura, M. Dohler, L.A. Grieco and G. Boggia. 2012. Traffic Aware Scheduling Algorithm for reliable low-power multi-hop IEEE 802.15.4e networks. In Proceedings of IEEE 23rd International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC), 12 Sept. 2012, doi: 10.1109/PIMRC.2012.6362805.
20. M.R. Palattella, N. Accettura, L.A. Grieco, G. Boggia, M. Dohler and T. Engel. 2013. On Optimal Scheduling in Duty-Cycled Industrial IoT Applications Using IEEE802.15.4e TSCH. IEEE Sensors Journal, 13 (10), 3655-3666, Oct. 2013, doi: 10.1109/JSEN.2013.2266417.
21. Erwan Livolant, Pascale Minet, Thomas Watteyne, “The Cost of Installing a 6TiSCH Schedule” AdHoc-Now 2016 - International Conference on Ad Hoc Networks and Wireless, July 2016, Lille, France.
22. Minimum Scheduling Function (MSF) https://tools.ietf.org/html/draft-ietf-6tisch-minimal-21
23. Breadth-First Searching (BFS) Algorithm https://en.wikipedia.org/wiki/Breadth-first_search
24. 6LBR Routing protocol for 6LoWPAN https://github.com/cetic/6lbr/
25. Duquennoy, S.; Nahas, B.A.; Landsiedel, O.; Watteyne, T. Orchestra: Robust Mesh Networks through Autonomously Scheduled TSCH. In Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems, Seoul, South Korea, 1–4 November 2015; ACM: New York, NY, USA, 2015; pp. 337–350.

ABSTRACT

Communication over wireless medium has an important place in the considerably evolving world of technology. In almost every field of life, wireless technologies have a significant ground which rises a variety of interesting applications such as in the industrial area. That because the wireless network offer benefits such as weight reduction, freedom from wires, flexibility and ease in maintaining the system. Those benefits can be seen in real world like environmental monitoring, smart home and many others. It exists different type of wireless technology used in the Internet of Things. Apart the Bluetooth, WI-FI and GSM, the IEEE 802.15.4e Time Slotted Channel Hopping (TSCH) as MAC protocol, is an important standard in the actual scalable Internet of Things (IoT) to habilitate deterministic low-power mesh networking. Wireless Sensor Network (WSN) are indispensable and once deployed in industrial applications, they require deterministic and tight latencies in scheduling. Scheduling in TSCH can be centralized or distributed. In Centralized scheduling, the central entity or coordinator, computes and assigns the communications resources like timeslot and channel to all nodes based on a certain scheme or algorithm. At the end, that schedule table needs to be known by each node in the network. Our work describes an efficient method to broadcast a schedule table in TSCH centralized scheduling in a global environment where IPv6 devices meet the Wireless Sensor Network through the 6LBR implemented in Raspberry Pi which works also as a scheduling server. We chose to make a schedule table by implementing a centralized link scheduling for IEEE 802.15.4e TSCH based industrial Low Power Wireless Networks, known as Path Conflict free Least Laxity First (PC-LLF) algorithm [10] in Contiki OS, a real time operating system for wireless sensor. We ensure that all data transmissions are done exactly as assigned in schedules by implementing a method to broadcast the schedule table information packets to all transmissions generator (nodes) with the aim to reduce packets collision and loss. We compared the number of packets messages needed to deliver to all nodes the schedule table information when using our method and when using other method such as CoAP and OCARI broadcasting method which is a custom-made method. The implementation takes in consideration the use of RPL protocol for network topology construction. The used hardware devices are Texas instruments CC 2650 Launchpad.