Lessons Learnt From the Implementation of the IEEE 802.15.4e-TSCH MAC

Ipsita Sanyal, Deeksha P Rao, Rakshana Gunasekaran, Sachin SM, T V Prabhakar
Department of Electronic Systems Engineering
Indian Institute of Science
Bengaluru, India
ipsita.san1@gmail.com, deekshaprao24@gmail.com, raks139@seas.upenn.edu, smsachin1995@gmail.com, tvprabs@iisc.ac.in

2023 15th International Conference on COMmunication Systems & NETworkS (COMSNETS) | 978-1-6654-7706-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/COMSNETS56262.2023.10041267

***Abstract*—In this work, we implement the IEEE 802.15.4e-Time Slotted Channel Hopping (TSCH) Medium Access Control (MAC) protocol software stack on standard hardware. An Application Programming Interfaces (API) support and ready to deploy star network is built and tested. The interfacing between micro-controller and radio uses the Serial Peripheral Interface (SPI) bus. Our implementation reveals that over 50% of the time overhead is SPI communication. To maintain synchronisation over the entire network, we implemented algorithms to successfully communicate between fast and slow nodes. Resilience to interference and packet error ratio is evaluated. We consolidate our implementation in terms of 6 lessons that were learnt in building the MAC protocol.**

***Index Terms*—Frequency hopping, IEEE 802.15.4e, IIoT, Time synchronization, TSCH, SPI, Information Element (IE)**

I. Introduction

The IEEE 802.15.4 wireless standard [1] based radio systems are widely applied in commercial and industrial environments owing to their noise resistant features. For instance, the radio supports Offset Quadrature Phase Shift Keying (OQPSK) modulation with Direct Sequence Spread Spectrum (DSSS). A few commercial applications include medical fields, environment monitoring, bio-diversity mapping, etc. The industrial applications include monitoring production lines, smart grids, real time condition monitoring and several other critical applications. The IEEE 802.15.4 piqued for its noise resistant capacity, lacks deterministic behavior which is an important requirement in time critical applications. To circumvent these shortcomings, the standard in the year 2012, introduced IEEE 802.15.4e [2] amendments to the base standard. The IEEE 802.15.4e is an evolving protocol with various Medium Access Control (MAC) modes and provides noise resistance, power efficiency and deterministic behavior making it the future protocol, for the IoT eco-system.

The IEEE 802.15.4e standard provides different MAC modes suitable for various applications. Time Slotted Channel hopping (TSCH) mode facilitates multi-hop multi-channel network, supporting various topology. The provision for channel hopping allows the motes to communicate over a range of frequencies imparting resistance towards channel interference and fading. The Time Division Multiple Access (TDMA) fortifies the unrestrained delay and provides communication reliability. Hence, the integration of TDMA and channel hopping imparts the deterministic characteristic to the TSCH mode.

This work proposes the implementation of the MAC layer for the IEEE 802.15.4e TSCH mode. The MAC layer is developed on the NRF52840 [3] micro-controller platform and the PHY layer functionality of the ADF7242 [4] radio transceiver chipset is used. The entire TSCH MAC stack as stated in the standards which includes the timeslot structure, the data frames, TDMA, channel hopping, time synchronization and the network formation algorithm are implemented and tested on hardware platforms as discussed in the following sections.

Funded by - Ministry of Electronics and Information Technology - MITO-003.

II. Related works

The authors of [5] have surveyed and outlined the major enhancement made to the MAC layer in 802.15.4e from 802.15.4 standard. Given that the IEEE 802.15.4e-TSCH communication protocol provides deterministic delay guarantees, it is ideally suited for Industrial IoT systems. There have been several simulation studies and a few hardware implementations as well. From the work reported in this paper, one can characterize the latency from application layer to the physical layer. Data transferred to the radio chip via the Serial Peripheral Interface (SPI) protocol and the associated SPI clock frequency is available as a handle for users. Several works are as discussed further in this section.

The works in [6] provide the performance evaluation of the IEEE 802.15.4e-TSCH standard based on the number of dedicated links and number of nodes in the network. The authors evaluate the network using Matlab simulations. With the increase in number of dedicated channels the reliability, throughput and energy consumption of TSCH network increases. Whereas, in our implementation we have set aside a few shared links within the slotframe structure. The Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) protocol implementation for these links is pending an implementation. [7] demonstrates the existence and reliability of the TSCH network in a noisy environment such as an aircraft cabin. Here three Wi-Fi access points operating in three different channels, each provide services to about 19 to 20 clients. We conduct Wi-Fi interference measurements in a laboratory setting and enumerate our results from practice.

Simulations of the standard have been performed on many open-source platforms like GNU radio companion [8], Open WSN [9], Contiki [10], etc. They provide detailed view on the development and performance of the IEEE 802.15.4e-TSCH MAC stack.

Since the standard provides flexibility of implementing the scheduling algorithm and node joining process, large number of works have been proposed in these areas. A scheduling algorithm called Traffic Awareness Scheduling Algorithm (TASA) [11], is a centralized scheduling method based on the graph theory. The Decentralized version of TASA is called DeTAS [12]. Paper [13] provides a distributed scheduling algorithm, aiming to minimize the difference between the number of packets to be sent and number of scheduled cell. Since scheduling is a requirement for the PAN coordinator or some node equivalent of this functionality, it is strictly not in the purview of the standard. In our work, we have hand crafted the slotframe structure, time slot, time schedule for shared channels, Enhanced Beacon (EB) broadcast slot. A node that wishes to join the network can pick the information available in the EB and automatically join the network. The works in [14] provide a node joining algorithm where the EBs are advertised in different channels to ensure faster joining. In order to increase the success rate of EB reception even by the nodes which are far from the PAN Coordinator, the author in [15] proposes "Sparse Beacon Advertisement". In this approach the nodes that has joined the network will re-transmit the Beacon, which helps in faster node association. Our implementation uses factory programmed single channel for EB reception and node joining.

III. Implementation

This section explains the algorithms and techniques followed to develop the TSCH MAC from scratch, as defined in the IEEE 802.15.4e-TSCH standard. The MAC stack was coded on ARM cortex M4 (NRF52840) micro-controller platform and interfaced to ADF7242 chipset (ADF), which supports base-band and PHY layer. The communication between the two hardware were established through the SPI protocol. The frequency of the SPI clock was set to 8MHz. The various functionalities implemented are available as Application Programming Interface (API) and explained in the following subsections.

A. TDMA, Channel Hopping, CCA

The IEEE 802.15.4e-TSCH mode shown in Figure1. is characterized by time slotting and channel hopping. The horizontal axis represents the division in time and the vertical axis depicts the division in frequency. The time slotting mechanism also known as TDMA has the time slots as its integral part which when combined together forms a slotframe. This slotframe structure repeats periodically. Each time slot is uniquely identified by an Absolute Slot Number (ASN). The ASN is a 5 byte number that is initialised by the PAN coordinator during the network initiation. For every timeslot the ASN increments by 1, which helps in maintaining a track of number of time slots since the start of network. As depicted in Figure 1., each timeslot can have a dedicated link (or dedicated channel) between two nodes to communicate or a shared link (or shared channel) between nodes. To enable nodes to join the network, a few links (or channels) are used to send out a special packet called the Enhanced Beacon (EB). The EB contains the Information Elements (IEs) related to network parameters.

The Clear Channel Assessment (CCA) is optional in TSCH mode, but we have implemented the algorithm as per the standard, mostly for improving the communication reliability in the presence of interference. In our implementation we have set the threshold for CCA to +10dBm above the noise sensitivity of the receiver.

The time slotting is implemented by utilising the services of NRF52840's application timer. All communication between Transmitter (Tx) and Receiver (Rx) nodes is expected to be completed in one timeslot. While Figure 3(a). shows the Tx's activities, Figure 3(b). shows the Rx's activities.

To determine the channel hopping frequency, we have implemented the hopping equation 1, which provides the frequency of operation based on the ASN. This frequency is set in the ADF before the node enters either Tx or Rx state. As per the standard there are 16 channels available for channel hopping.

$$f = F[(ASN + channel\_offset)\%number\_of\_channels] \quad (1)$$

F is implemented as a lookup table that contains a list of 16 frequencies. This channel hopping sequence is shared with the node as Channel Hopping IE through EB during node joining.

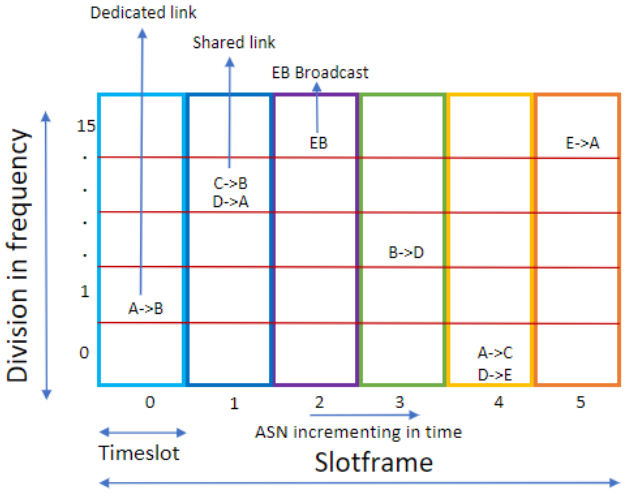
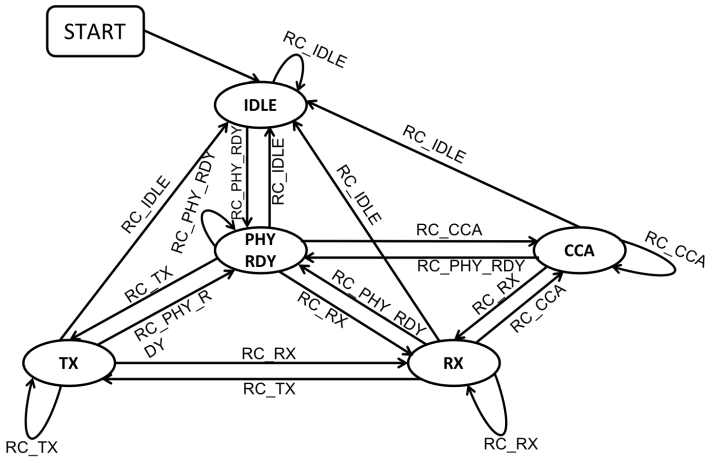


Fig. 1. Illustration of time and frequency division

B. Choice of Radio Chip-set

The requirement for a radio chip includes support for accepting MAC Protocol Data Units (PDUs), timeslot structure as shown in Figure 3., as well as modulating the data suitable for IEEE 802.15.4 communication. Our survey included CC2480 [16], CC2420 [17], MCR20AVHM [18] and DW1000-1-TR-13 [19]. The Texas Instruments CC2480 chipset's current consumption was found to be higher than the rest of the surveyed chipsets. The DW1000-1-TR-13 has support for Binary Phase Shift Keying (BPSK), which is not the suitable modulation scheme for the IEEE 802.15.4 standard. One drawback among CC2480, CC2420, MCR20AVHM and DW1000-1-TR-13 is that they don't provide us the flexibility to implement the timeslot structure proposed in the IEEE 802.15.4e-TSCH standard. In contrast to the above hardware, ADF7242 overcomes these limitations, and emerges as a suitable candidate to demonstrate our implementation. Some of its features include low power, flexible register settings for configuring modulation schemes, packet size, and mouldable state machine. We describe these functions in the ensuing sections.

C. State Machine



3

Fig. 2. Finite state machine

The Figure 2. illustrates the state machine of the ADF7242 depicting the various states of the radio chipset namely, Idle state, Physical ready (Phy_Rdy) state, Transmitter (Tx) state, Receiver (Rx) state and CCA state, along with the commands required by the ADF to transit to these states. The commands for state transition, channel hopping, setting modulation schemes and other packet configurations are issued by the controller and communicated to the ADF through SPI communication. The **first lesson** is that a minimum of $380us$ is required for any state transition. This includes prior clearing of interrupts and enabling the 'rc_rdy' interrupt and providing the state transition command. Each register configuration requires 3 bytes of data to be sent over SPI. This requires 125us. Along with these Phy layer settings, the data packets, EBs, and the enhanced acknowledgement (ACK) packets are created in the MAC stack i.e., the software running on the embedded platform and communicated to the ADF. This communication requires a minimum of $1000us$.

We measured the current consumption in the Idle, Phy_Rdy, and Rx states. We found that the low power state is indeed the Idle state as it consumes $1.8mA$. This state consumes $8.2mA$ lower than Phy_Rdy state and $17.2mA$ lower than Rx state. To configure this low power Idle state, the ADF supports the RC_IDLE command. In this state, the radio transmitter and receiver blocks are in sleep mode. Furthermore, we complete other configuration parameters such as modulation scheme, packet configuration and buffer configuration in this state. In our implementation, the communication frequency is set while in Phy_Rdy state because it supports the RF frequency synthesizer block and system calibration. The ADF can be made to transit to this state using RC_PHY_RDY command. The transition to Rx state and Tx state can be achieved using RC_ RX and RC_TX command respectively. Upon entering these states, the receiver and transmitter block of the radio are enabled. In Rx state the received data frame is stored in the RX_BUFFER of the radio, which can be read by the micro-controller and in the Tx state the data frame in TX_BUFFER written by the controller is transmitted. In CCA state, the ADF will calculate the Received Signal Strength Indicator (RSSI) value and stores the result in "read_rssi" register, which can be read by the controller and provides information about the channel busy state or otherwise.

D. Time Slot Structure Implementation

To suit our hardware environment of micro-controller and radio, we defined a timeslot template ID. The MAC PAN Information Base (PIB) attributes supported follow section 6.5.4.2 of the standard, also shown in Figure 3. Figure 3. depicts ADF's state machine mapped to the standard's Tx and Rx timeslot structure. The terms used in the standard time slot diagram are as explained in Table I. The attributes shown in the table are transmitted as part of the TSCH timeslot IE. The values of the MAC PIB attributes for the defined timeslot template ID are listed in the section IV (C). Before a state transition, "the state transition ready interrupt" is enabled to ensure that the ADF is available to accept a state transition command from the MAC stack (i.e., controller).

In Rx timeslot, the ADF will transit from Idle state to Phy_Rdy state during macTsRxOffset. The transition to Rx state and data packet reception is completed during macTsRxWait. Over macTsTxAckDelay, the ADF transits to Tx state from Rx state. Transmission of the ACK, transition to Phy_Rdy state and finally to Idle state is completed in macTsMaxAck.

In Tx timeslot, the ADF is initially in the Idle state, which will transit to Phy_Rdy state accounting for the macTsCcaOffset. In macTsCca the radio transits to CCA state and measures the channel energy and transits back to Phy_Rdy state. During macTsRxTx the ADF will transit to Tx state and within the period of macTsMaxTx the ADF will transmit the data frame. During macTsRxAckDelay the turnover from Tx state to Rx

TABLE I
STANDARD TIMESLOT TERMINOLOGIES ASSOCIATED WITH MAC PIB TIMESLOT TEMPLATE ID

Term	Description as per standard
macTsCcaOffset	Waiting time till node can enter CCA state.
macTsCca	Duration for CCA.
macTsRxTx	Time to transit to Tx after CCA.
macTsMaxTx	Time required to send a maximum length Packet.
macTsRxAckDelay	Waiting time to start Rx state for ACK reception.
macTsAckWait	Time between end of packet transmission and start of ACK reception.
macTsRxOffset	Minimum waiting time for a node before entering Rx state.
macTsRxWait	Minimum waiting time for Packet.
macTsTxAckDelay	Time between end of packet reception and start of ACK transmission.
macTsMaxAck	Time to transmit a Maximum length ACK.

state is performed and the ADF waits for the ACK for the duration of macTsAckWait. Soon after receiving the Ack, the ADF will transit back to Phy_Rdy state followed by Idle state within the macTsMaxAck time period. **The second lesson** we learnt is that the SPI protocol used for state transitions, register configuration, and MAC PDU transfer requires 5080us. This shows that, for a 10ms time slot, over 50% is utilized for communication between controller and radio.

E. Time Synchronization

For any micro-controller to operate, it ought to have a clock source. Generally a crystal clock oscillator is the preferred choice. The crystal's accuracy is measured in parts per million (ppm). The NRF52840 works on a crystal that has an accuracy of $+/-40$ppm, implying that given two NRFs with $64MHz$ clock frequency the frequency deviation between them will be $+/-256Hz$ i.e., their frequency will vary in the range of $63.99744MHz$ to $64.00256MHz$. Since the oscillator circuit is implemented within the controller, due to temperature variations, clock drifts result in frequency variation.

To maintain synchronization among the nodes in a TSCH network, the standard suggests two synchronization methods viz. Frame based synchronization and Ack based synchronization. In Frame based synchronization the receiving node will correct its clock based on the packet arrival time, whereas in Ack based synchronization the time correction information is sent to the transmitter node to correct its clock. Although it suggests the two-time synchronization algorithm it does not specify the application scenario for the two methods. The **third lesson** we learnt is that, the nodes in a TSCH network requires deployment of both Ack based and Frame based synchronization because in the IEEE 802.15.4e TSCH network, there can be multiple transceivers with faster and slower clocks. Each node should be equipped with both the algorithms and should be able to choose one among the two synchronization algorithm.

Our first step is to evaluate the number of communicating node pairs and determine which node in a given pair is faster. Since it is impossible to fasten up a slow node, we adopted the mechanism to slow down a faster node by introducing a delay to achieve time synchronization. Subsequent to this step, either Frame based or Ack based synchronization is applied. For instance, the Ack based synchronization is executed when the Tx clock is faster and the frame-based synchronization is implemented when the Rx node has a faster clock.

In our deployment, for example, in a scenario where node A is transmitting to node B in timeslot 0 and transmitting to node C in timeslot 4, (captured in Figure 1.) and node A is faster than node B, but slower than node C. In such a case, node A performs Ack based synchronization with node B and frame-based synchronization with node C and the set of delays and calculated offsets are unique to each of these links.

1) Determination of the faster node: Algorithm 1 explains the steps followed in determination of the faster node. The determination of the faster node is processed at the receiver node. Initially an offset is computed by taking the difference between the previous arrival time of the data frame and the current arrival time of the data frame received. The acquisition of the current arrival time is done by starting an application timer concurrently at the beginning of the Rx time slot and stopping it at the arrival of the Start Frame Delimiter (SFD) of the data frame. This algorithm is made to loop for 10 times, and hence the current arrival time value of the $(n-1)^{th}$ loop is assigned to the previous arrival time variable of the n^{th} loop. The reason we deployed this algorithm to loop 10 times is for the fact that there is a significant change in the SFD arrival time from one loop to the next and this difference becomes stable after the first 5 loops. Table II depicts the first 10 packet's SFD arrival time which is the average value calculated over 10 such transitions. In this case the Tx clock is faster than the Rx clock. Initially the previous arrival time variable is assigned with a value 0. Depending on the sign of the offset computed the faster and the slower node is determined and the Tx_fast or the Rx_fast counter is incremented. At the end of the loop, the counter with the highest value is considered for determining the node with the faster clock. The value of the current arrival time of the last loop is assigned to the expected arrival time variable used in the time synchronization algorithm.

TABLE II
SFD TIMINGS FOR FIRST 10 TRANSITIONS

Transition No.	SFD arrival time (ms)
1	936
2	819
3	536
4	439
5	183
6	2538
7	2291
8	1086
9	1095
10	1090

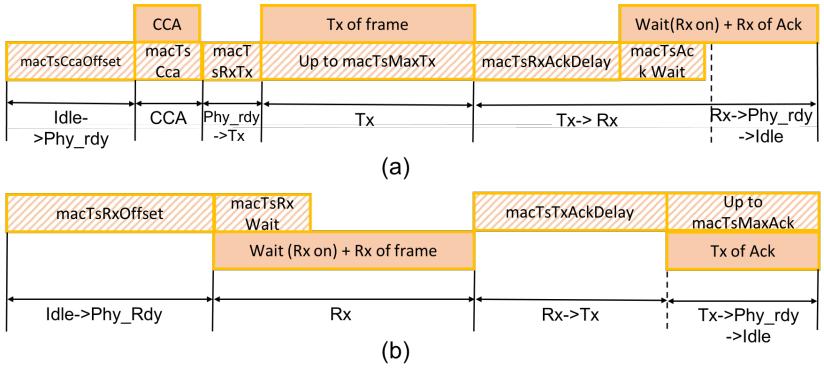


Fig. 3. Timeslot structure of acknowledged transmission defined in section 6.5.4.2 of the standard. The implemented finite state machine is mapped to this structure.

Algorithm 1 Determination of faster node

$N \leftarrow 1$
$Tx\_fast \leftarrow 0$
$Rx\_fast \leftarrow 0$
while $N \leq 10$ **do**
  $diff \leftarrow previous\_SFD\_Arrivaltime$ - $current\_SFD\_Arrivaltime$
  if $diff > 0$ **then**
    Tx_fast++; /* Tx node is faster */
  else
    Rx_fast++; /* Rx node is faster */
  end
  $N++$
end

2) Ack Based Synchronization: Algorithm 2 outlines the deployment of Ack based synchronization. The time offset is calculated at the receiver end by evaluating the difference between the expected arrival time and the actual arrival time as stated in the standard. The actual arrival time is acquired in similar lines to the current arrival time as discussed in the previous section. If the computed offset is a positive value, then this value is sent to the Transmitter node in the ACK. This value is given as delay in the next timeslot of the Tx node, before transiting to the Tx state. If the offset is a negative value, then it is given as a delay at the Rx and no offset value is communicated to the Tx through the ACK. However the latest non zero delay received through the ACK is maintained.

3) Frame Based Synchronization: Algorithm 3 presents the procedure to deploy the frame-based synchronization. If the computed offset is a positive value, then this value is given as a delay in the next timeslot of the Rx node before transiting to the Rx state. If the value is negative then the offset value is communicated to the Tx node through ACK and is given as delay in the consecutive Tx time slot and the previous non zero delay is maintained at the Rx node.

F. Node (Re)Joining

In our IEEE 802.15.4e-TSCH implementation, we broadcast one EB per slotframe in a single channel. This EB includes:

Algorithm 2 Ack based synchronization when Tx is a faster node

$Offset \leftarrow expected\_Arrivaltime$ - $actual\_Arrivaltime$
if $diff > 0$ **then**
  Send Offset in ACK and Delay the Tx node according to the Offset received
  Give 0 delay in Rx
else
  Send 0 as Offset in ACK and retain the Previous delay in Tx node
  The negative Offset delay is given to the Rx node
end

Algorithm 3 Frame based synchronization when Rx is a faster node

$Offset \leftarrow expected\_Arrivaltime$ - $actual\_Arrivaltime$
if $diff > 0$ **then**
  Send Offset in ACK and Delay the Tx node by the according to the Offset Received
  Give previous delay in Rx
else
  Send 0 as Offset in ACK and no delay is given in Tx node
  The negative Offset delay is given to the Rx node
end

(a) TSCH Synchronization IE. (b) TSCH Slotframe and Link IE. (c) TSCH Timeslot IE. (d) TSCH Channel hopping IE. These IEs contain information about the network structure and network parameters. A minimum of two slotframes are required for the node to join the network. The network parameter data, extracted from the first EB is stored in the Non-Volatile Memory (NVM). Soon after, a flag is set in the memory. The ASN data extracted from the TSCH Synchronization IE from the second EB is used to synchronize with the network, which is globally same for all nodes in the network. This is also the required EB for the node rejoining process. With the help of the flag, the two step joining process is reduced to one step rejoining process. We use this two beacon process because extracting and storing the network parameters consume some time. We use the ASN from second beacon for accurate

synchronization Algorithm 4 outlines the steps followed by a node to join or rejoin the network.

Algorithm 4 Node Joining algorithm

if *Network Parameters stored in NVM* **then**
  Retrieve the following from NVM
  - State
  - Timeslot period
  - Channel Offset
  - Channel hopping sequence
  - Number of channels

else
  Receive EB
  Store the network parameters in NVM

end
Receive EB by following the timeslot period and update ASN
Follow the schedule as mentioned in the link IEs

G. API Description

Our implementation of the IEEE 802.15.4e-TSCH standard comprises of the following main functions:

1) **channel_write():** This function is called in Phy_Rdy state in order to set the channel frequency in ADF.
2) **timeslot_tx():** This holds the entire Tx structure. This function encloses two sub-functions called packet_write() and ack_read() to write the data packet to the ADF's TX_BUFFER and to read acknowledgement from ADF's RX_BUFFER respectively.
3) **timeslot_rx():** This holds the Rx Structure and encapsulates packet_read() and ack_send() to read data packet from ADF's RXBUFFER and write ACK to the ADF's TX_BUFFER respectively.
4) **Enhanced_Beacon():** This function will broadcast the EB carrying all IEs. This function is specific to PAN coordinator.
5) **Receive_Beacon():** It is function specific to new nodes trying to join network, with this function being called the nodes will receive the EB and joins the network by following node joining process as explained in section III-F.

The **fourth lesson** we learnt is that the timeslot_tx and timeslot_rx execution time is $1420us$ for a controller running at $64MHz$. If we replace with a $100MHz$ controller, our calculations show that the execution times reduces to $910us$. Clearly, SPI communication delay overwhelms the execution delay.

IV. Experimental Settings and Results

Our current implementation employs a star topology network with 4 sensor nodes including the PAN coordinator. Figure 4. captures the network with the transceiver nodes placed about 2 meters away from the Coordinator. The transmit power was set to $+3dBm$. The implemented link schedule for the experimentation setup is shown in Figure 5. We have implemented a slotframe with four timeslots across sixteen frequencies. Among four timeslots, three timeslots are utilized by the PAN coordinator to communicate with the nodes and one timeslot is used for EB broadcast. The nodes will transmit the temperature sensed by them during their respective timeslots. This experimental setup was used to validate the implementation by computing parameters such as:

1) Analysis of TSCH MAC PIB attributes for macTimeslotTemplate - Here we define each state transition to the standard's timeslot template.
2) Packet Error Ratio (PER) - To check the accuracy of our implementation which includes frequency hopping, slotframe structure, timesloting and time synchronization.
3) The Node joining time - Basically to evaluate our node joining algorithm, which estimates the time required to join a productive network.
4) Effect of Interference

These parameters are as discussed in the following sections.

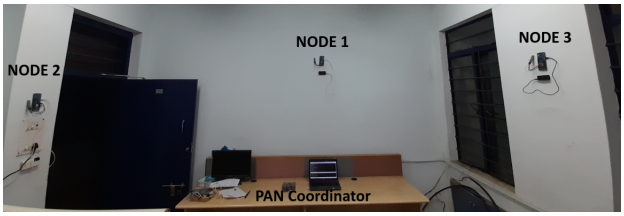


Fig. 4. Experimental setup depicting Star network topology

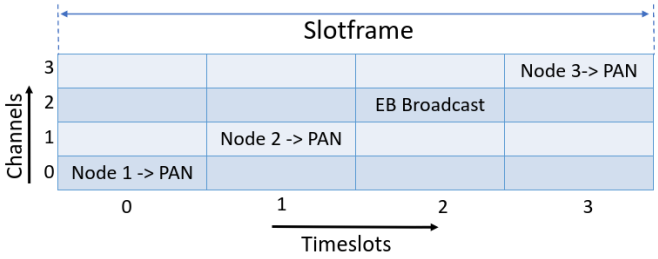


Fig. 5. Link schedule of the network

A. Analysis of TSCH MAC PIB attributes for macTimeslotTemplate

In our implementation, the communication between the micro-controller and the radio chip is through the SPI bus. Our experiments revealed that a minimum of $100us$ delay is required for reliable communication. Thus the state machine transitions are paced with this overhead. Implementing all the transitioning states of the radio required for the timeslot structure sums up to a $13ms$ timeslot period, Although the Clear Channel Assessment for the TSCH mode is optional, it can come handy when there is a significant interference in the operating frequency. The overall time consumed by CCA was found to be $2600us$. This time is added to the Idle state of the Rx node. As the Idle state is the least current consuming state for ADF, this serves us as a power efficient technique.

In order to achieve the standard's default timeslot period of $10ms$, our supported packet size is 72 bytes with CCA disabled. The **fifth lesson** we learnt is that the CCA state takes two additional transitions, and requires several register configuration such as threshold for CCA, CCA type, RSSI average time and reading the RSSI value. Thus by skipping this state, we were able to achieve 10ms timeslot. The total timeslot duration achieved with CCA is $13ms$ and without CCA is $10ms$. Table III shows $13ms$ timeslot measurements for Rx. The TSCH MAC PIB attributes explained in section III-D is mapped to state transition of ADF. The average time taken in the different states is encouraging from a power consumption perspective. For instance, the Rx and Tx states are about 60% of the timeslot period. We achieved this by tight synchronization. Table IV shows the Rx timeslot measurements for $10ms$. The size of the ACK packet is also set to 72 bytes. Table VI and Table V depicts the Tx timeslot measurements for $13ms$ and $10ms$ respectively. The respective tables show entries for macTsCca with transitions Phy_Rdy to CCA and macTsCCA to Idle to Phy_Rdy transitions. The macTsAckWait time makes sure the timeslot delineation is maintained.

TABLE III
RX TIMESLOT MEASUREMENTS FOR $13ms$

TSCH MAC PIB attributes for macTimeslotTemplate	Transitions	Average Time Taken(us)
macTsRxOffset	Inactive to Idle Idle to Phy Rdy	3525.56 1099.47
macTsRxWait	Phy Rdy to Rx + Rx of pckt	5834.07
macTsTxAckDelay	Rx to Tx & ACK Send	1720.99
macTsMaxAck	Tx to Phy Rdy Phy Rdy to Idle Wait time to complete 13ms	22.87 571.18 250.74
Timeslot Period	-	13024.88

TABLE IV
RX TIMESLOT MEASUREMENTS FOR $10ms$

TSCH MAC PIB attributes for macTimeslotTemplate	Transitions	Average Time Taken(us)
macTsRxOffset	Inactive to Idle Idle to Phy Rdy	845.75 1058.17
macTsRxWait	Phy Rdy to Rx + Rx of pckt	5204.95
macTsTxAckDelay	Rx to Tx & ACK Send	1837.07
macTsMaxAck	Tx to Phy Rdy Phy Rdy to Idle Wait time to complete 10ms timer	23.02 521.29 552.82
Timeslot Period	-	10043.07

TABLE V
TX TIMESLOT MEASUREMENTS FOR $13ms$

TSCH MAC PIB attributes for macTimeslotTemplate	Transitions	Average Time Taken(us)
macTsCcaOffset	Inactive to Idle Idle to Phy Rdy	849.77 1100.28
macTsCca	Phy Rdy to CCA	2146.55
macTsRxTx	CCA to Phy Rdy	523.75
macTsMaxTx	Phy rdy to Tx + Tx of pckt	1652.36
macTsRxAckDelay	Tx to Rx & ACK Received	5511.01
macTsAckWait	Rx to Phy Rdy Phy Rdy to Idle Wait time to complete 13ms	23.23 520.71 695.92
Timeslot Period	-	13023.58

TABLE VI
TX TIMESLOT MEASUREMENTS FOR $10ms$

TSCH MAC PIB attributes for macTimeslotTemplate	Transitions	Average Time Taken(us)
macTsCcaOffset	Inactive to Idle	849.45
macTsCca	Idle to Phy Rdy	1100.32
macTsRxTx	Phy Rdy to Tx	497.02
macTsMaxTx	Tx of pckt	1182.40
macTsRxAckDelay	Tx to Rx & ACK received	5528.51
macTsAckWait	Rx to Phy Rdy Phy Rdy to Idle Wait time to complete 10ms	22.78 520.46 337.76
Timeslot Period	-	$10,038.7$

B. Time synchronization and Packet Error Ratio

Figure. 6 illustrates the packet arrival time at the receiver with and without synchronization. The x-axis of the graph indicates ASN (timeslot number) and y-axis indicates the Rx node timeline. The Radio's Rx state begins soon after macTsRxOffset. The 'Green' horizontal line is the ideal expected arrival time. There are two possible cases for the IoT node, where the Rx might be faster than the Tx node, or Rx might be slower than the Tx. The lower arrival time is indicative of Tx being faster, and is corrected in real-time by the Ack based synchronisation. This ensures that the RX is in sync with the Tx. Similarly, the increasing delay in arrival time depicts the case of Rx node being faster than Tx node, which is corrected by Frame based synchronization. Even the slightest deviation from the expected arrival time is corrected using these synchronization algorithms. The packet arrival time is computed by taking SFDArrival time (beginning of packet arrival). The SFDArrival time was taken for 6660 timeslots each of $10ms$ spread over a total of 26648 timeslots with a inter timeslot of $30ms$. Table VII indicates a communication pair of nodes have lost 102 packets out of 6660 packets. This evaluates to a packet

error ratio of 1.53% without enabling Ack based or Frame based synchronization. With our synchronization algorithms, the packet loss ratio is significantly reduced to 0.1%.

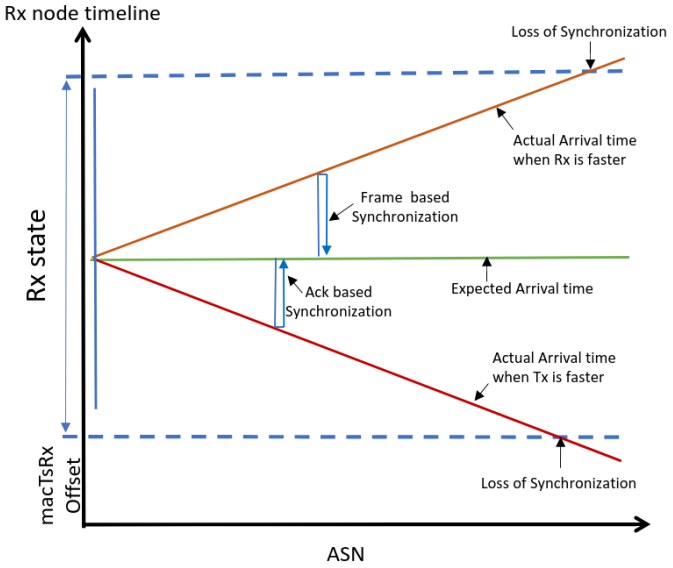


Fig. 6. Frame and Ack Based Synchronization to handle variation in packet arrival time

TABLE VII
EFFECT OF SYNCHRONIZATION ON PACKET ERROR RATIO

Number of Packets transmitted	Number of Packets lost	PER(%)
6660(Without Synchronization)	102	1.531531
6660(With Synchronization)	7	0.105105

C. Node joining time

Our implementation assumes the existence of the PAN coordinator. Sensor nodes join the network sequentially. The node joining time majorly depends on the frequency and successful reception of the EB. In our implementation the EBs are broadcast once in a slotframe. The nodes perform passive scanning in a known static channel awaiting a beacon. The last byte of the received beacon is checked to ensure that all IEs required by the node is received successfully. If the node fails to receive and analyse all IEs from the EB broadcast, then it waits for another EB. Our implementation results show that an average of $250ms$ is required for a node to join the network. As explained in section III-F, the rejoining of an existing node is a single step process and it requires significantly less time compared to the new node joining time. Table VIII shows the average joining and average rejoining time with and without CCA implementation.

The **sixth lesson** we learnt is the need to handle catastrophic events such as unexpected shutdowns, reboots and battery issues. A node rejoin procedure is simplified if the time slot period, channel offset, number of channels hopping sequence and communication state are readily available.

TABLE VIII
JOINING TIME OF THE NODE WITH AND WITHOUT CCA IMPLEMENTATION

Condition	New node joining time (ms)	Existing node joining time (ms)
With CCA	250	89
Without CCA	273	72

D. Effect of Interference

To test the effect of interference on the TSCH network, our source consists of a Universal Software Radio Peripheral (USRP) B210 transmitting a continuous $2.4GHz$ sinusoidal signal. The nodes were made to hop across 4 channels. The idea is make sure that the interference is experienced as frequently as possible. Table IX shows the RSSI value read by the ADF for different interference settings. We first established the baseline without packet receptions as well as no interference. The remaining results, as expected, is observed that the average PER increases when interference spans the full channel range. When USRP generates noise of $-30dbm$ on a particular channel, the node suffers a PER of about 10.23%.

TABLE IX
EFFECT OF INTERFERENCE ON PACKET ERROR RATIO

Condition	RSSI	PER
Channel is free (No reception & no interference)	$-85dBm$	-
Inteference in Channel	$-66dBm$	0.06%
Inteference in Channel	$-30dBm$	10.23%

V. CONCLUSION AND FUTURE WORK

This work constructs the IEEE 802.15.e-TSCH MAC using commercially available ARM controllers and radio hardware. The implementation involved constructing and building several algorithms, code optimizations, testing the deployment etc. Our implementation exhaustively covers the IEEE standard document as well as adheres to several recommendations. Several lessons were learnt from problems faced during the MAC implementation. The first two lessons deals with the SPI overhead experiences. The lesson 3 and 6 explains on the implementation methodology, which is not mentioned in the standard. From lesson 4, it is evident that the impact of using the higher accurate clock source on the timing values is unexciting. Lesson 5 shows the usage of optional states to meet time constraint. Our future goal is to develop a more generic MAC stack which abstracts the state machine of the radio chip. This will ensure that a large number of platform hardware can be supported.

REFERENCES

[1] "IEEE Standard for Low-Rate Wireless Networks," in IEEE Std 802.15.4-2020 (Revision of IEEE Std 802.15.4-2015) , vol., no., pp.1-800, 23 July 2020, doi: 10.1109/IEEESTD.2020.9144691.

[2] 802.15.4e-2012 - IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer

[3] https://infocenter.nordicsemi.com/pdf/nRF52840_PS_v1.1.pdf

[4] https://www.analog.com/media/en/technical-documentation/data-sheets/ADF7242.pdf _datasheet.pdf

[5] Guglielmo, Domenico& Anastasi, Giuseppe & Seghetti, Alessio. (2014). From IEEE 802.15.4 to IEEE 802.15.4e: A step towards the Internet of Things. Advances in Intelligent Systems and Computing. 260. 135-152. 10.1007/978-3-319-03992-310.

[6] Touloum, S., Bouallouche-Medjkoune, L., Yazid, M. et al. Performance modeling of the IEEE 802.15.4e TSCH enabling both shared and dedicated links in industrial WSNs. Computing 104, 859–891 (2022). https://doi.org/10.1007/s00607-021-00990-2

[7] M. Gürsu, M. Vilgelm, S. Zoppi and W. Kellerer, "Reliable co-existence of 802.15.4e TSCH-based WSN and Wi-Fi in an aircraft cabin," 2016 IEEE International Conference on Communications Workshops (ICC), 2016, pp. 663-668, doi: 10.1109/ICCW.2016.7503863.

[8] I. Sanyal and T. Prabhakar, "IEEE 802.15.4e-TSCH MAC Stack Extensions to GNU Radio," 2022 14th International Conference on COMmunication Systems NETworkS (COMSNETS), 2022, pp. 199-203, doi: 10.1109/COMSNETS53615.2022.9668390.

[9] https://aio.inria.fr/berkeleys-openwsn/

[10] https://github.com/contiki-ng/contiki-ng/wiki

[11] Palattella, Maria Accettura, Nicola Dohler, Mischa Grieco, Luigi Boggia, Gennaro. (2012). Traffic Aware Scheduling Algorithm for Reliable Low-Power Multi-Hop IEEE 802.15.4e Networks. IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC. 10.1109/PIMRC.2012.6362805.

[12] N. Accettura, E. Vogli, M. R. Palattella, L. A. Grieco, G. Boggia and M. Dohler, "Decentralized Traffic Aware Scheduling in 6TiSCH Networks: Design and Experimental Evaluation," in IEEE Internet of Things Journal, vol. 2, no. 6, pp. 455-470, Dec. 2015, doi: 10.1109/JIOT.2015.2476915.

[13] Y. Zhang, C. Chen and S. Zhu, "An Adaptive Distributed Scheduling Algorithm for IEEE 802.15.4e TSCH Protocol," 2019 3rd International Symposium on Autonomous Systems (ISAS), 2019, pp. 193-198, doi: 10.1109/ISASS.2019.8757770.

[14] I. Khoufi, P. Minet, E. Livolant and B. Rmili, "Building an IEEE 802.15.4e TSCH network," 2016 IEEE 35th International Performance Computing and Communications Conference (IPCCC), 2016, pp. 1-2, doi: 10.1109/PCCC.2016.7820597.

[15] https://etd.iisc.ac.in/handle/2005/5229

[16] https://www.ti.com/lit/ds/swrs074a/swrs074a.pdf?ts=1662525748719ref_url=https%253A%252F%252Fwww.google.com%252F

[17] https://www.ti.com/lit/ds/symlink/cc2420.pdf?ts=1662525816626ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FCC2420

[18] https://www.nxp.com/docs/en/data-sheet/MCR20AVHM.pdf

[19] https://download.mikroe.com/documents/datasheets/DWM1000_datasheet.pdf