IoT Protocol Stack Design Using Sound Communication

1. Abstract:

As humans are growing at a faster rate so they required more and more ways of communication and data transmission. Due to this developing nature, frequencies around us are keep growing which inturn have a huge impact on human health. We have read many surveys and articles regarding effect of high frequencies on human health. Because of this problem we came up with the solution of replacing very high frequencies range to some kind of low frequency range. So we decided to give a digital solution to this problem is to transfer some of the information using nearly ultrasonic sound. After reading and going through many articles regarding data transfer using sound we came up with the unique idea which includes an SDK called 'CHIRP' developed by a team of researchers i.e. James Nesfield,CEO-Chirp, Dr. Adib Mehrabi, Head of Research-Chirp, Dr. Daniel Jones, Chief Technology officer- Chirp

2. Introduction :

Data-over-sound presents a compelling solution for many device-to-device connectivity applications, particularly for use cases that require frictionless,low cost connectivity with nearby devices. This paper introduces the fundamental concepts and benefits of data-over-sound connectivity, and explores key application areas within the Internet of Things, including provisioning smart devices and facilitating secure near-field communication in low cost, low power scenarios. The rise of IoT devices in the home and workplace has created a world where the data and connectivity are becoming increasingly complex. As Iot technology advances and the demand for efficient ways of communicating data between these devices grows, the world has witnessed a rise in emerging new data transmission technology which are looking to provide secure and effective solutions for sharing information. One

solution rising to meet these new demands is ‘’data-over-sound”, which harness devices existing or external speakers and microphones to send and receive data over an acoustic channel.

Why using Sound as transfer media?

(related health hazards due to different frequencies range)

The wireless devices have become an integral part of our everyday life. These devices are being used for many purposes such as for Internet and Telecommunication Being a theoretical approach, the purpose of this study is to investigate the effects of wireless devices on the human body. These devices emit the harmful radiations which cause the diseases like male infertility, brain tumor, hearing impairment, fetus, effect on eyes, etc. Besides, these radiations severely affect other various parts of the human body. This study investigated three major diseases, i.e., brain tumor, male infertility and hearing impairment.The results, collected through interviews and surveys, show the intensity of harm of different wireless devices; Mobile phone is the most effective device with 96%, Bluetooth Device 32%, Laptop 54%, Tablet PC 14% and Wireless router 20%.

Near ultrasonic Sound waves **17-20KHz**	**4G Networks** **2-8GHz**	**Wifi routers frequency** **2.4GHz/5GHz**	**Infrared Light** **300GHz-430THz**
Headache	Blood–brain barrier	Contributes to the Development of Insomnia	Damages human eye lens
Dizziness and Nausea	Cancer	Damaging to Childhood Development	Premature skin ageing
-	Electromagnetic Hypersensitivity	Derails Brain Function	-
-	Effects Glucose metabolism	May Impact Fertility	-

References:

https://www.omicsonline.org/open-access/effects-of-wireless-devices-on-human-body-jcsb-1000229.php?aid=77066

State of the Art:

One of the main advantages of data-over-sound is that the physical infrastructure needed to facilitate sonic data transfer is already largely in place. The voice is gaining momentum as the primary control mechanism for many IoT devices, and as such, microphones are increasingly being incorporated into more and more IoT devices. Beyond mobile devices and home assistants such as Alexa and Google Assistant, we are seeing voice control being added to smart TVs, fridges, door bells, vacuum cleaners, light bulbs, locks, and thermostats. As humans continue to communicate with IoT devices using sound, we are seeing millions of devices of all form factors already equipped with the required processor, speaker, and microphone for data-over-sound functionality - without requiring any physical upgrades to existing hardware. With companies always looking to innovate and future-proof their services, many are now realizing the potential of data-over-sound to provide seamless device-to-device connectivity either to nearby devices or remotely (e.g. down a phone line), using nothing but sound.

	Chirp	QR	NFC	Blue-tooth	Wi-Fi	Li-Fi	ZigBee 802.15.4	LoRa	Sigfox
Two-way communication	•			•	•	•	•	•	•
One-to-many broadcast	•					•	•		
Non-line-of-sight transmission	•			•	•		•	•	•
Works in RF-restricted environments	•	•	•			•			
Zero setup / pairing / configuration	•	•	•						
Available to application by default	•	•							
Low power operation	•	•	•	•			•	•	•
Can transmit with sub-$2 electronics	•	•	•	•			•		
Can receive with sub-$2 electronics	•			•			•		
Wireless broadcasts confined to room boundaries	•	•	•						
Transmit over ranges > 10m	•	•		•	•	•	•	•	•
Can limit the transmission range to < 1m	•	•	•	•					
Supported by dumb media channels and P.A. systems	•	•							
Supported by typical mobile devices	•	•	•	•	•				
Supported by low-end mobile devices	•	•		•	•				
Typical usable max data rate	1 kb/s	3kb	424 kb/s	25 mb/s	70 mb/s	1 gb/s	250 kb/s	50 kb/s	100 b/s
Typical max range	100m	10:1	20cm	100m	50m	10m	100m	10km	40km

There already exists a plethora of connectivity solutions, including extremely short range (NFC and QR); short range, high bandwidth (Bluetooth and Wi-Fi); long range, low power (Sigfox and LoRa); and short range, low power (Zigbee). Each technology has certain advantages that make it more or less

suitable for specific applications. As any technology provider in the IoT arena will appreciate, the application-specific needs are not always as simple as choosing a technology based on cost, data rate, and range. Things can get a lot more complicated when one considers the finer details of device support, backwards compatibility, user experience, and security. Further detail is introduced by the technical specifications of the protocol, such as the nature of the transfer medium (RF, optical, acoustic), number of required channels, security concerns, or whether two way and one-to-many communication is needed. An overview of these considerations is given in Table 1. This is by no means exhaustive, and each use-case will bring its own specific requirements. Nonetheless, data-over-sound has certain advantages that make it particularly attractive for a number of use cases.

References : https://pages.arm.com/rs/312-SAX-488/images/data-over-sound-Chirp_Arm.pdf

https://chirp.io/

LIMITATIONS :

As we are using CHIRP sdk, so there are some limitations regarding this sdk which cannot be altered by some user but it can only be done by developers. Few of them are mentioned below :

1. **RANGE :** As we are transferring the data onto sound using some nearly ultrasonic frequencies range so it has a limitation of range that it can only be used upto 10m or less. So this may arise the long distance communication problem for us.(**solution : multihopping**)
2. **DATA RATE :** Using this chirp sdk we can just transfer data with the speed of 7bytes/seconds(i.e. 7bps). So to transfer large size of data we have to just divide it into several fragments and send them one by one. This situation may also rise a limitation that we cannot send huge amounts of data over sound. So only important data is to be sent through sound.(**solution : our algorithm**)
3. **BROADCAST :** This is the main and crucial problem to deal with. As sound travels all round 360degree in environmental media so it is possible that it can be received by any receiver placed near to it. As all the data transmitted over the sound waves behave like broadcast signal in wireless networks so we have to use some mechanics to control the flow of data in an isolated system.**(solution : our contributed algorithm)**

3. Related works-

(Objective : 15 words, Methodology & Techniques(PL/MAC), Applications : 30-40 words, Limitation: 30-40 words) (100 words)

Chirp Signal-Based Aerial Acoustic Communication for Smart Devices

Aerial acoustic communication delivers information through airborne sound waves. In this paper, they have focused on inaudible aerial acoustic communication for off-the-shelf audio interfaces in long-range indoor environments. The contributions of the paper are three folds, firstly adopting chirp signal that was originally used in radar applications, they drastically extended the communication range up to 25 m at maximum.Secondly They proposed a software digital modem for smartphones that can efficiently demodulate the chirp signal by combining fast Fourier transform (FFT) and Hilbert transform.The proposed envelope detector can also compensate for Doppler effect And lastly also introduced a TV content recognition service as an example application along with a backend query server in order to overcome the low data rate (approximately 16 bps).

I Can Hear More: Pushing the Limit of Ultrasound Sensing on Off-the-Shelf Mobile Device

In this paper, they introduced iChemo, a technology that can enable the ability of COTS (commercial-off-the-shelf) mobile devices to sense high-frequency ultrasounds that is beyond their capacity to fully capture. They introduced how the nonlinearity feature can be used to realize coprime sampling algorithm on mobile devices, with only one available built-in ADC. Also, tackled the two realistic challenges against the implementation of coprime sampling on COTS mobile devices, including the sample sequence alignment and resolution of AAFs. By evaluating iChemo on both Android and

iOS devices, they showed that it can sense the ultrasound at 60 kHz, which is over twice the current upper limit of 24 kHz.

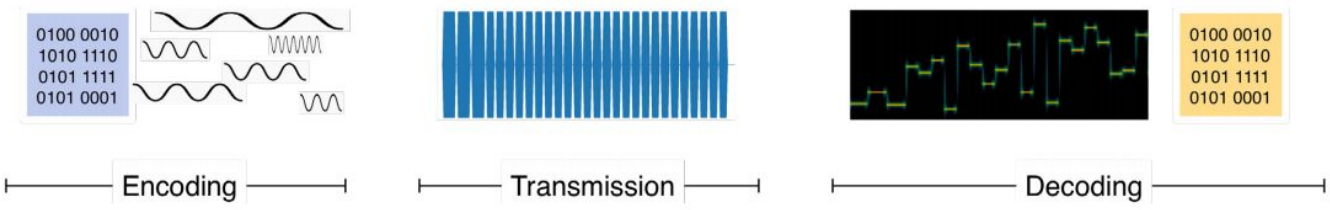
A Practical Guide to Chirp Spread Spectrum for Acoustic Underwater Communication in ShallowWaters

Small and cheap micro AUVs enable diverse underwater monitoring applications in shallow inshore waters; e.g., inspection of underwater assets, observation of water quality, and identification of pollution sources.This paper presents strategies to find proper chirp parameters. First, they studied the benefits and applicability of chirp-based modulation against FSK. Second, they discussed the impact of the chirp parameters in real-world scenarios and identify sweep spots w.r.t. Bandwidth to improve the standard deviation of the position in the time domain up to 86 % above this point. Third, they presented instructions to choose these parameters in an efficient and appropriate way for any shallow water scenario.

4. SYSTEM DESCRIPTION :

CHIRP API Configuration in Raspberry Pi

Chirp is a pre-built software development kit (SDK), which seamlessly transfers data over sound waves. Chirp encodes an array of bytes as an audio signal, which can be transmitted by any device with a Speaker and received by any device with a Microphone and CHIRP SDK. It is designed to roburst over distances of several metres, and in noisy, everyday environments. Chirp is configurable to use audible or inaudible near-ultrasonic frequencies. As the transmission takes place entirely via audio signals, no internet connection or prior pairing is required and device within hearing range can receive the data.



We have developed IOT based framework using CHIRP SDK. Firstly we are collecting environmental data such as concentration of CO2, NO2 etc. in air using various sensors and transmitting it over ultrasonic sound waves.

For this task to get accomplished, we require certain hardwares which are as follows

- RaspberryPI (any version)
- Ultrasonic USB Microphone
- 3.5mm Jack Speaker

CONFIGURATION OF RASPBERRY PI

We implemented CHIRP in raspberryPi using Python programming language.

1. First of all we need to install the SDK's dependencies by this command

```
sudo apt-get install python3-dev python3-setuptools portaudio19-dev libffi-dev libsndfile1
```

2. Then install the CHIRP SDK directly from PyPi

```
pip3 install chirp sdk
```

3. To configure the SDK we need to create .chirprc file in the Home directory with three instruction set i.e. **app_key**, **app_secret**, and **app_config**

PROTOCOLS FOR CHIRP :

Chirp SDKs can be configured to use different communication protocols, which determine various properties of the transmission:

Transmission rate and range :

Protocols can be tailored for short-range, high-speed applications, or long-range applications at a lower bit rate.

In typical real-world environments, Chirp can transmit at approximately the following rates:

- 1000bps at NFC range (30cm)
- 150bps at peer-to-peer range (3m)
- 15bps at long range (100m)

Frequency range :

Chirp can use audible or inaudible near-ultrasonic frequencies.

- Audible frequencies are recommended for channels which have a limited audio sample rate -- VoIP connections, lossy codecs, or lower-spec embedded devices. Specific protocols are available for each of these scenarios.
- Near-ultrasonic frequencies should be used when noise disturbance is not wanted. Frequencies between 17kHz and 20kHz are supported by virtually all mobiles and computers, but are generally inaudible to the human ear.

Available protocols :

- standard: audible transmission, for peer-to-peer ranges
- ultrasonic: inaudible transmission, for peer-to-peer ranges
- 16kHz: intended for embedded devices supporting 16kHz sample rate
- 16kHz-mono: intended for embedded devices supporting 16kHz sample rate
- pstn: for transmission over phone lines and VoIP
- ultrasonic-multichannel: for inaudible multi-channel transmission

- long-range: for large venues, industrial and stadium transmissions
- ultrasonic-poly-192: high-speed, short-range transmission

Selecting a protocol :

First we have selected ultrasonic-poly-192 protocol for transferring the data.

- Transmits up to 192 bytes in 10.32s .
- Uses a frequency range from 17500Hz to 19360Hz with a total of 1 channel.

Advantage - High speed data transmission upto 148.8 bps.

Disadvantage – Short Range transmission only upto 2 metres.

Then we selected ultrasonic-long-range protocol for transferring the data.

- Transmits up to 8 bytes in 4.2s
- Uses a frequency range from 18000Hz to 19800Hz with a total of 1 channel.

Advantage - Data transmission upto 15.2 bps with long range capability

Disadvantage – Simultaneous sending and receiving data is not possible.

Atlast we selected the ultrasonic-multichannel protocol which tackles the problem of simultaneous transmission and receiving of data with a long range and multiple channels.

Ultrasonic-multi channel-protocol :

Ultrasonic multi channel protocol, allowing several devices to transmit data simultaneously.

- Transmits up to 7 bytes in 3.24s (17.3 bps).
- Uses a frequency range from 17500Hz to 20090Hz with a total of 8 channels.

Essentially the frequency spectrum from 17.5kHz to 20kHz is split up into 8 different bands, with each channel operating in its own frequency spectrum.

Frequency Range	Number of channels	Transmission rate	Transmission Distance
17500Hz - 20090Hz	08	7 Bps	10metres(approx)

Setting the channel

The number of channels the current configuration enables by getting the channel count. Single channel protocols will return 1.

```
count=chirp.channel_count
```

Each instance will listen to every channel,but broadcast only on a single specified channel.

If the configuration supports more than one channel,then we can set the transmission channel that we want to send data on.The example below sets the current channel to 3. Every call to send after this will automatically use this transmission channel.

```
chirp.transmission_channel= 3
```

We can get the current transmission channel as

```
channel=chirp.transmission_channel
```

Received channel

When data is received,each callback is passed the channel the data was received on.

```
def on_received(self, payload, channel):
    print("Received data on channel " + channel)
```

Channel State

In multichannel mode,each channel may be in different states. To check the current state for a particular channel is as follows-

```
if chirp.get_state_for_channel(3) == chirpsdk.CHIRP_CONNECT_STATE_STOPPED:
    print("Channel 3 is currently sending data")
```

5. ALGORITHM AND METHODOLOGY USED :

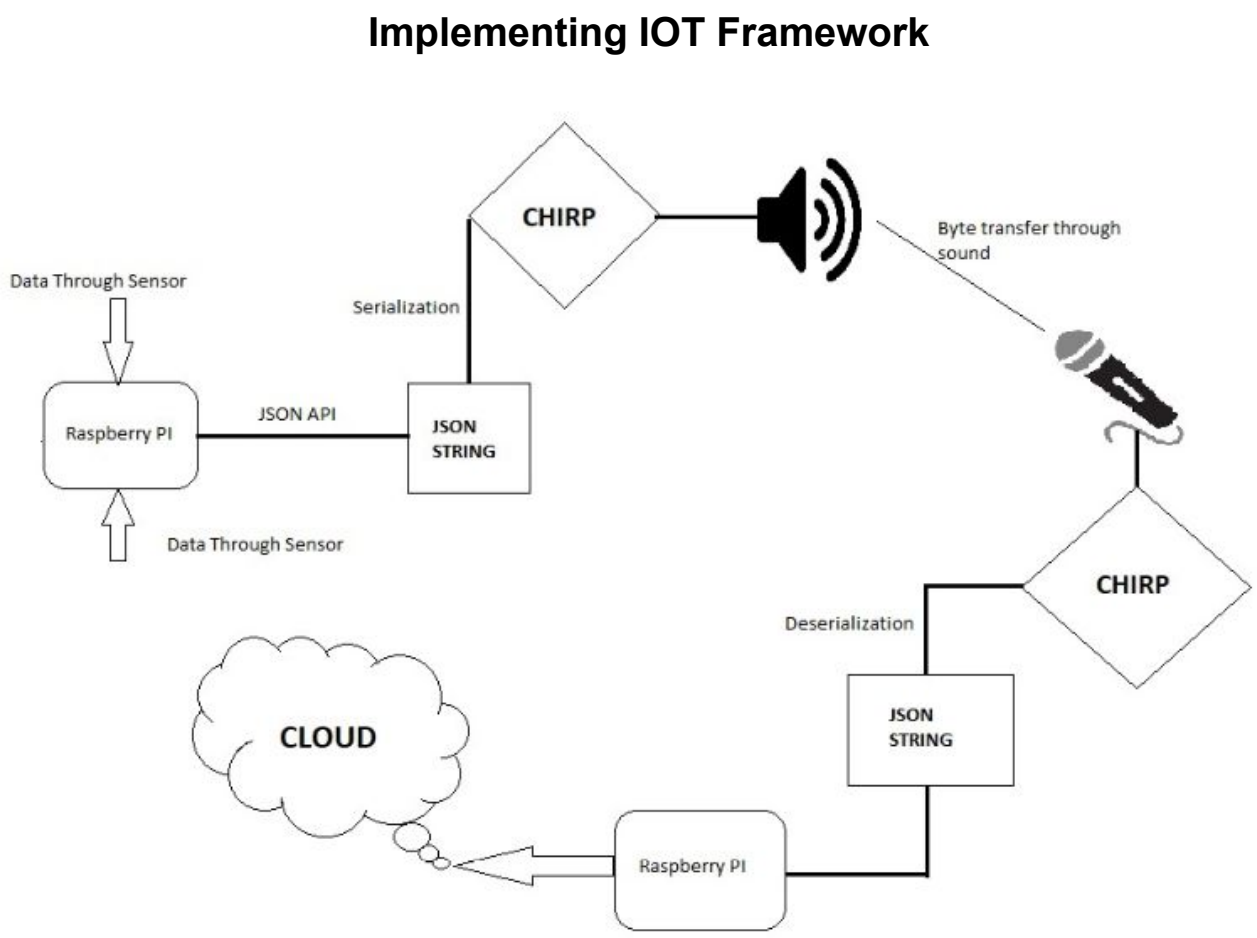


FIG 1: IOT FRAMEWORK

1. The Data is being collected through various sensors along with the timestamp.
2. A devices randomly selects a number between 0-6 which becomes its channel Id.
3. The Data is being received by all other channels and sent by the one generated channel.
4. This data is converted into Python dictionary,afterward converted into json string using JSON API.
5. This JSON string is converted into byte stream also termed as serialization.
6. As we are using Ultrasonic-multi channel protocol which transfers 7 byte data at a time, so we are dividing the byte stream stream into 7 byte fragments for transferring it.
7. These fragments are now transferred using Chirp to all other receiver.
8. After receiving data through various channels, it creates different JSON files which appends the received data in the existing file after converting the byte stream received to json string.
9. At last these JSON files are uploaded to Cloud .

Implementing Stack Protocol

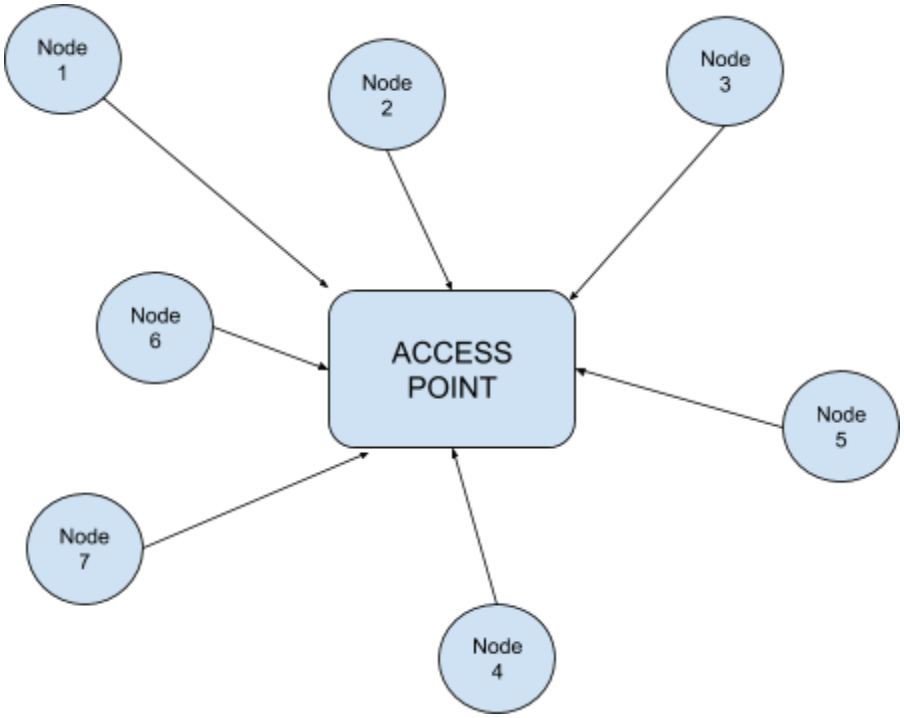


FIG 2 : PROTOCOL DESIGN

This is the scenario in which we are going to implement our protocol where, there are several nodes sending data to access point through ultrasonic multichannel in chirp sdk . Here nodes can only send data to access point and access point will receive their data through various channels at the same time and upload it to the cloud.Access point has been given a fixed channel (ie Channel 7) for sending the data while node are going to get random channel id valued from 0 to 6.

Since there are maximum 8 channel for access point, so the problem rises here that, what if there will be more than 8 nodes at the same time and second problem is, what will happen when two or more nodes will get same channel id.

What happens when two Nodes get the same channel ID?

Since a node is getting a random Channel ID that is from 0-6, It might happen that two nodes gets the same Channel Id, hence there will be data collision while transmitting it through the same channel. We have tackled this problem using well known concept of CTS and RTS as follows.

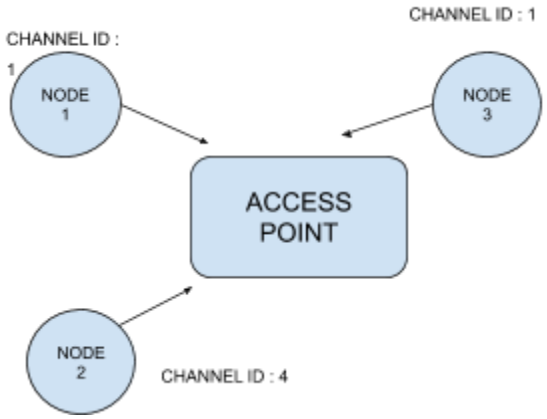
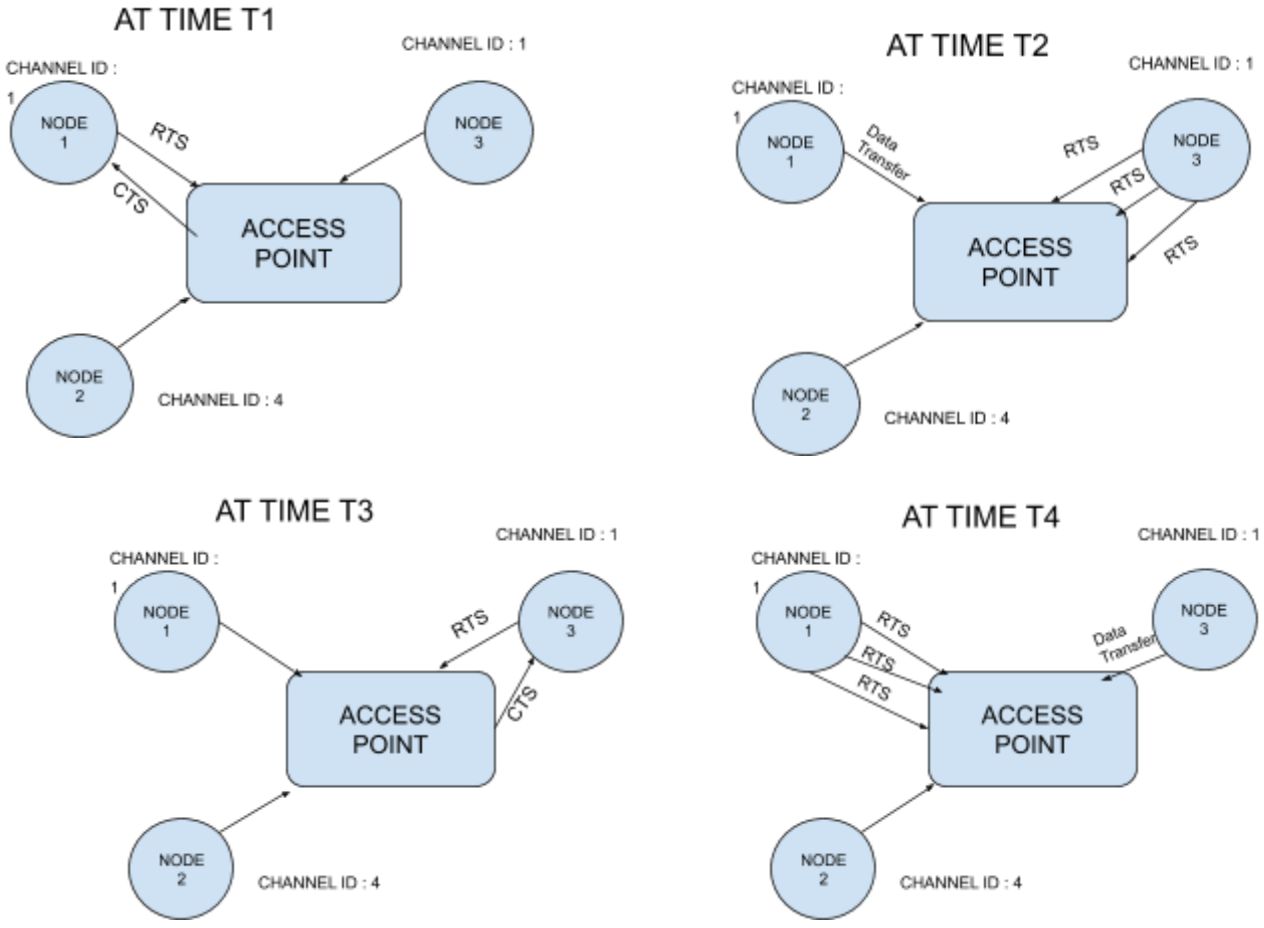


FIG 3: PROBLEM WITH SAME CHANNEL ID

1. A Node sleeps for a random time.
2. If two Nodes gets the same Channel ID (Node 1 and Node 3 gets the same Channel Id) , the one who wakes up early sends an RTS packet to the Access point (let us assume Node 1 wakes up early and sends the RTS)
3. As soon as Access point receives an RTS packet it sends CTS packet back to the node.
4. The data transmission takes place between the Node and Access Point, meanwhile the other node(Node 3) wakes up and sends RTS after every 50 to 60 seconds taken randomly, to Access Point , but Access Point is busy in receiving the data from Node 1 in channel 1,it will not send CTS packet to Node 3.

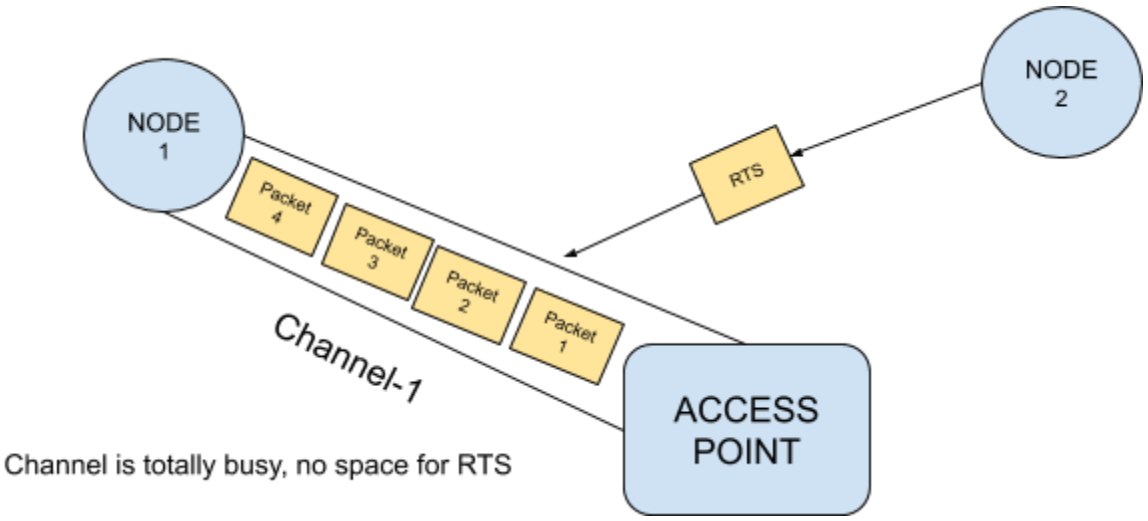
5. After completion of transmission of data from Node 1 ,it again sleeps for random time between 50 to 60 second.
6. The periodically sending Node 3 receives a CTS packet from Access point and start transmitting the data.
7. This process works in synchronisation until any external interrupt occurs.



Why RTS and Data doesn't collide if two nodes are transmitting through same channel?

Consider a case when two nodes 1 and 2 got the same channel ID, and the data transmission is going on between Node 1 and access point, and the Node 2 sends RTS to access point. It is possible that the data and RTS may collide resulting in data loss. But in our implementation it is not occuring.

The reason is simple as when the data is transmitted by a Node, it continuously sends the data in the packet of 7 bytes such that no two sounds of each packet gets collided and also there is no gap between two packets, which keeps the channel busy, hence the RTS packet sent by other Node doesn't received by the Access point. This case is depicted as follows-



The code snippet for this mechanism is as follows-

```
#message here is the json string which is divided into chunks of 7 bytes, and then transmitted.
while start < len(message):
```

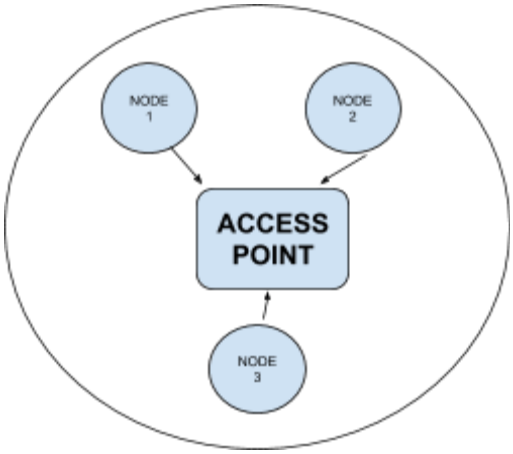
```
if start + 7 < len(message):

        end = start + 7
else :

        end = len(message)

payload = sdk.new_payload(message[start:end])
start = end
sdk.send(payload)
time.sleep(3.5)  #this is the main reason of collision avoidance
```

Here time.sleep(3.5) is basically that the system sleeps for 3.5 second after sending a payload, this time is being experimentally calculated through running tests for multiple times. The case when it was not 3.5 second, either the RTS was being collided with data or the converted sound for each packet is being overlapped, which results in receiving the incorrect data or even null payload by the Access point.

As the bandwidth of the channel is not too large so there is no probability of sending two packet or fragment simultaneously in a certain channel. That's why RTS and data will not collide at any instance of time because there is no space for RTS to go through the channel while transmitting data because of traffic of fragments in the particular channel.

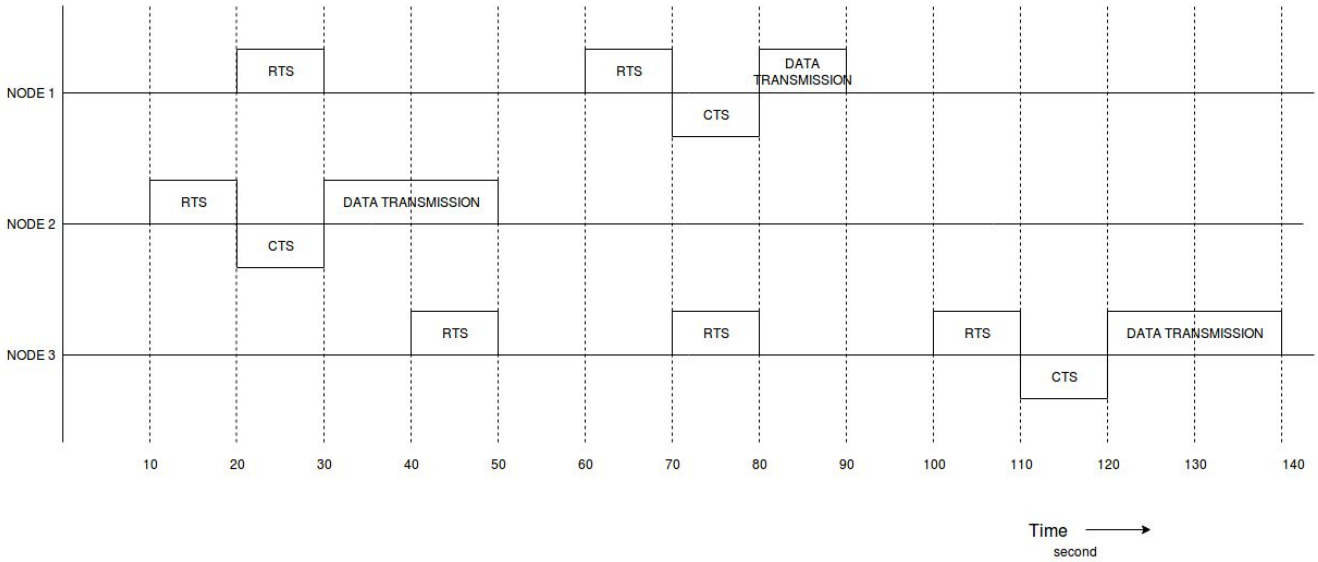
6. PROTOTYPE DESCRIPTION OF SYSTEM

Case study 1

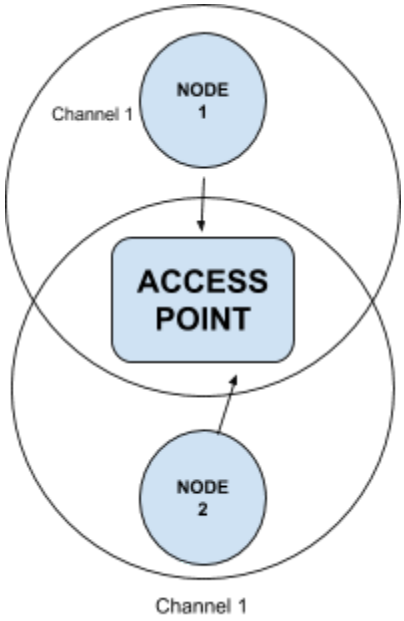


Here the three nodes got the same channel ID ie channel 1. Initially, as mentioned all the three nodes sleeps for the random time. Let's suppose node 1 ,node 2, node 3 sleeps for 50, 55, 60 second respectively. Node 1 wakes up early and transfers the data via protocol mentioned earlier. Then it sleeps for a random time.In meantime the node 2 sends RTS since Channel 1 was busy it doesn't receives CTS. As soon as it gets free CTS is received by the node 2 and data transmission takes place. Same situation occurs for Node 3.

This is how the data collision is prevented.

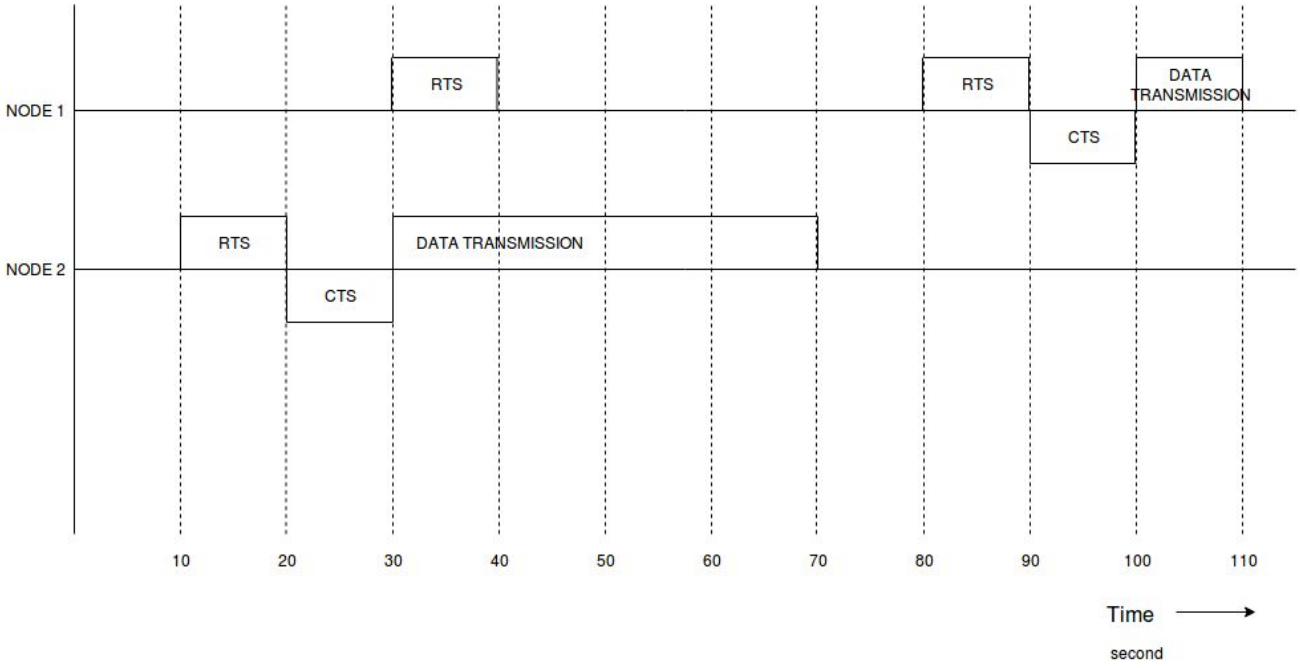


Case study 2:

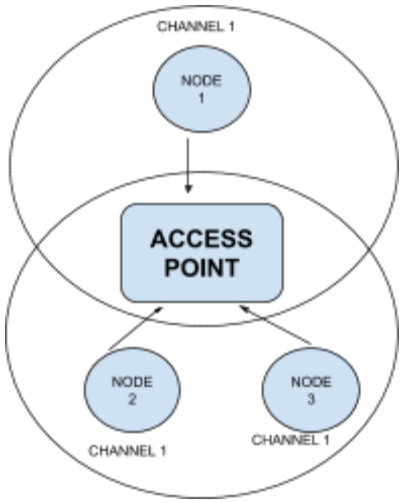


Here Node 1 and 2 are not in the range of each other, and the channel Id allotted to both of them is 1.The one to wake up early starts the data transmission through the protocol described. The other Node sends RTS periodically, as soon as the data transmission gets over with the first Node,it sleeps for a random time, the second one receives CTS and transmits data. In this way we have prevented data

collision.

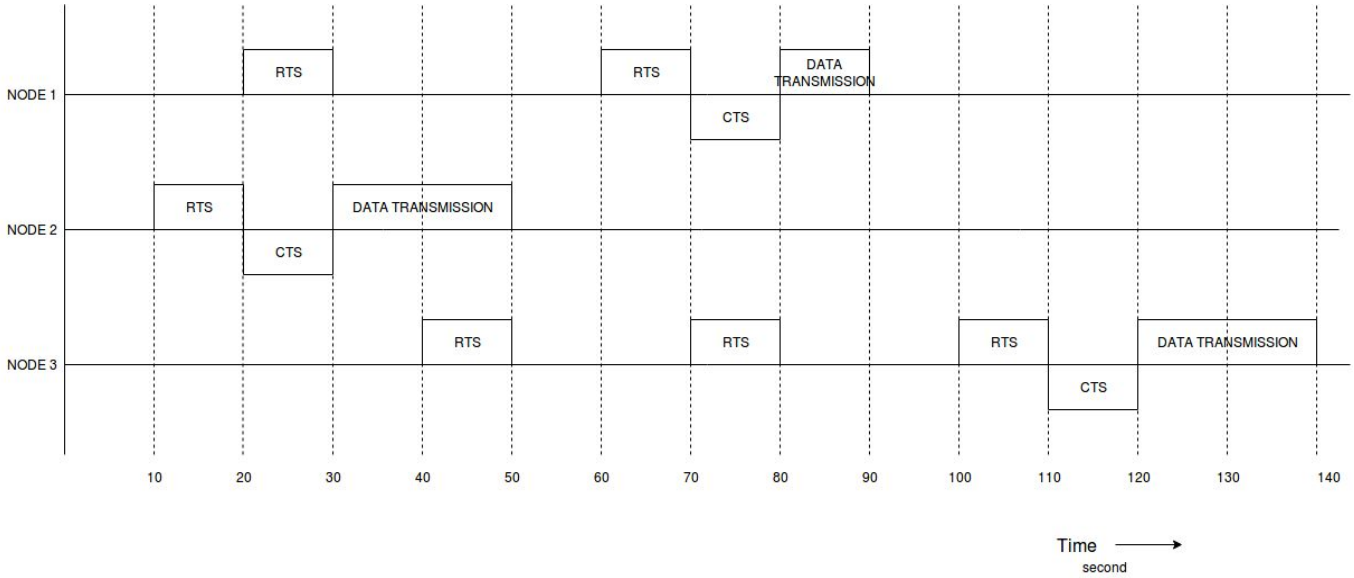


Case Study 3 :

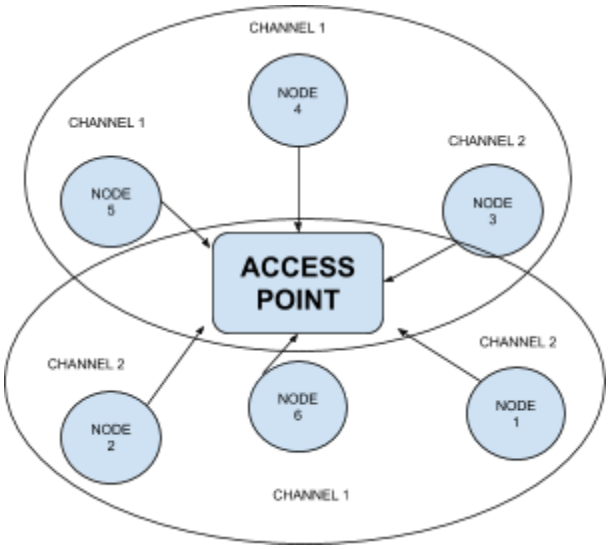


Here in this case study there are 3 node in which node 2 and node 3 are in the same range while node 1 is in different range, so it cannot communicate with node 2 and node 3. Initially all three nodes sleeps for random time suppose 6,9,3 respectively. Hence node 3 wakes up first and sent an RTS to access point and afterward receives CTS and start transmitting data. While in between node 2 and node 3

wakes up and and start sending an RTS to access point after random interval of time. During the transmission time access point will not provide any other node with CTS. but after the completion of transmission the node who sends and RTS first will get CTS and the earlier node will get sleep for some time.Hence this is how we get rid of data collision in this scenario



Case Study 4 :



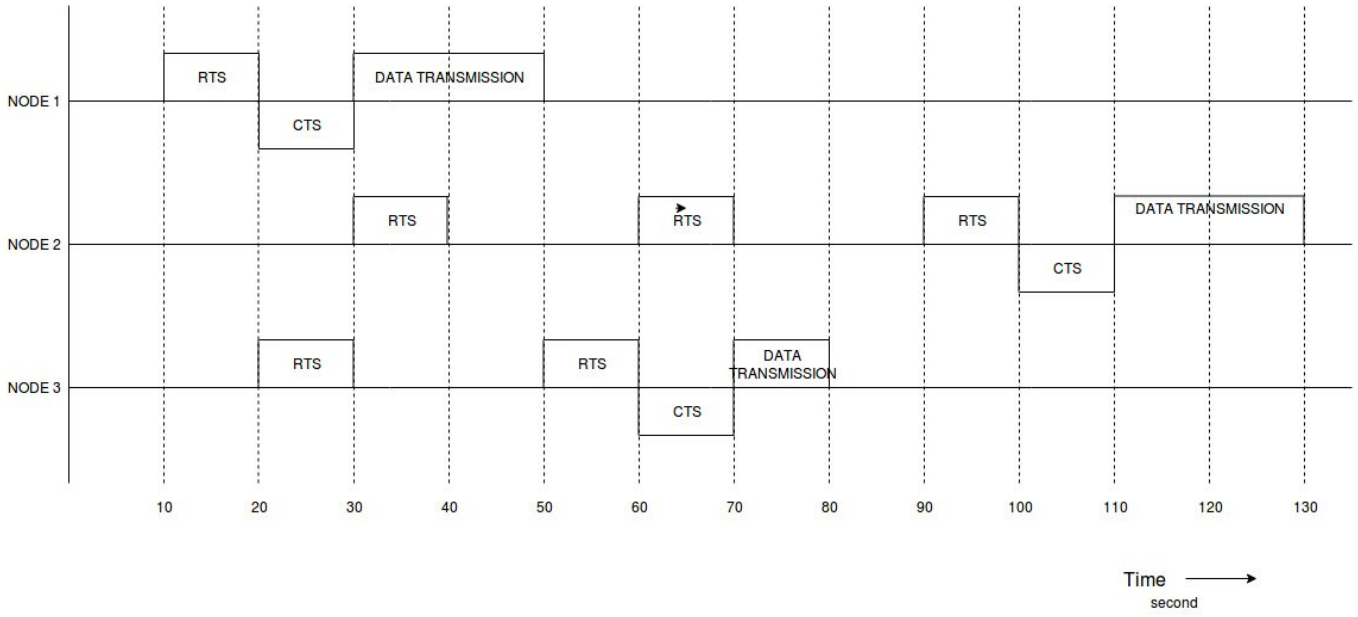
In this scenario there are total six node participating for the transmission of data. As above diagram gives us scenario that node 1, node 2 and node 3 are in same range including the access point, while

node 4, node 5 and node 6 are in different range. As we know node 1 , node 2 and node 6 are sending data through channel 1 while node 3,node 4 and node 5 are ready to send data through channel 2. So there creates a problem of data collision in channel 1 and channel 2 .

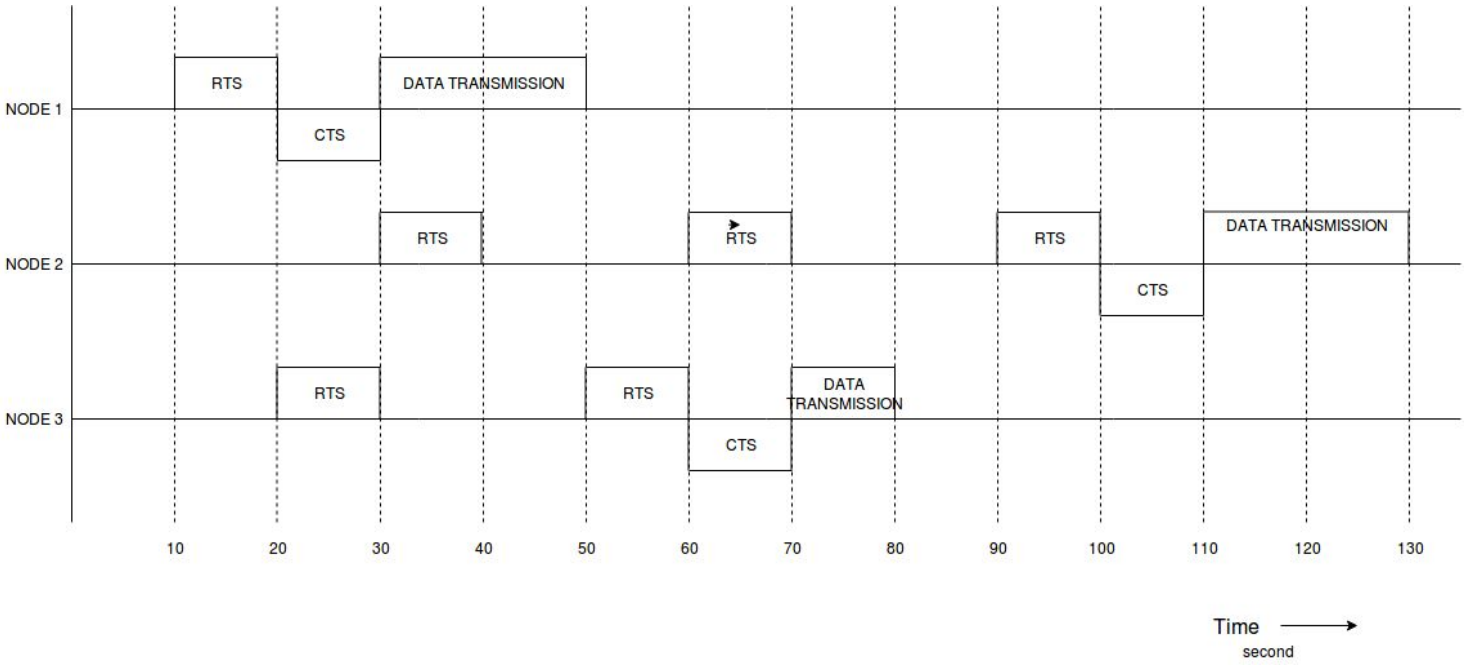
So at first all the node sleeps for random time. The first who wakes up, sends RTS and gets CTS for channel 1 and same condition goes to channel 2 as we are using multi channel protocol so it can receive 7 different channel data at the same time. Hence we can avoid data collision for channel 1 by above solution for case study 1 , 2 and 3. Same as goes for channel 2.

So in any case we can avoid data collision and waste of data and energy

CHANNEL-1



CHANNEL-2



7. Conclusion

The sound waves need a carrier for themselves – air, water, metal conductors so cannot be used in space or vacuum. That's the only drawback and hence they may not be useful for GPS. Other than that, they can be perfect carriers for data without harming anyone with dangerous radiations as the radio waves and do.