Wireless Ad-hoc and Sensor Networks Project Report

Abstract

The purpose of this report is to present all the preparations and work done on the WSN project, this report consist of 6 sections; Introduction, Preparation, Design, Implementation, Measurement Results and finally Results Discussion.

Introduction

There are two main goals for this project; one is to modify the existing CTP protocol to store the addresses of sensors where all data packets hop through until it reaches the sink node and also to store cost to sink.

Second, to carry out an experiment to measure maximum achievable throughput where packet loss is less than 5%.

Linear topology was used to carry out the measurements experiment.

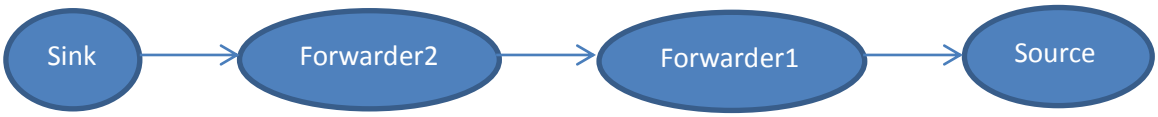

Figure 1 Linear Topology

In the preparations section we will show what we did to prepare and understand the CTP protocol and will also show some of the things that we found out which helped us with all the stages of the project.

In the design section we will discuss the design options that we had and will also cover the chosen approach in more details.

The implementation section, we will go over the code modification of the CTP protocol and all the extra work/code that we had to write to achieve the first goal. We will also show code snippets and explain the logic behind it.

The measurements results section we will cover the experiments we did, the reasons for choosing these approaches and also the results we got.

Finally in the last section we will discuss the results.

Preparations

We started with taking small steps to first understand the CTP protocol and how the whole MultihopOscilloscope program works. First we configured the sensors and ran the java listener tool to look at the packets. Next step was to generate a constant 16bit number for every packet. From there we found out the below information:

The above steps helped us decide the best simplest approach which will be discussed in the below section.

Design

Once we understood the CTP protocol and how it's packets are ordered and what is available in it, we had two design options, one is to add a new field in the CTP packet which would tell us how big the message is so we can append addresses and ETX to the end of it and the second option is to reserve the bits needed (as we know how many hops will be used) to record the addresses and ETX of the forwarders and source.

To keep the implementation as simple as possible we have decided to go with the second option which is to reserve the bits needed.

We added three new uint8_t fields in the Oscilloscope packet/header, two for the forwarders and one for the source where each will hold an address for the Forwarder/Source sensor.

We also added another uint16_t field for holding the ETX value (ETX is a 16bit to match the ETX in the CTP header).

As a result of adding these fields to the MultihopOscilloscope we had to decrease the number of reading from 5 to 3 readings to make sure that we don`t go over the size of the CTP packet (which contains the MultihopOscilloscope packet as the message).

To make it easier and clearer to trace the packets being generated from the Source and not having too many packets printed out by the java listener, we decided to stopped all the forwarders and the sink from generating packets (see implementation section for more details) and only the source can generate data. We have also decided to disabled any debugging packets from being generated as we didn`t want to cause confusion and they were not needed at this stage.

We also decided to modify the transmission power of the sensor to be as low as possible to be able to carry out our experiments in a space small.

Implementation

Before starting the implementation process, we had to first change the power transmission and stop the debugging packets as mentioned above in the design section.

As mentioned in the design section we wanted to only have the source node/sensor generating data. This was achieved by a function so that only the source node (where id of the node is 3) to generate and send data.

The next step was to modify the generated/received packets every time before it gets forwarded to the next hop/sensor; this modification is done to add the sensor's id to user's data and also calculate the new ETX.

The Engine is responsible for receiving the data (being generated from the same sensor or being forwarded from another sensor). We did our modifications in function.

What this code does is it accesses the payload of the packet using oscilloscope. It then checks the node id and assigns the appropriate address id to the appropriate address field (which was added to the oscilloscope to hold the addresses of hops/sensors).

The last line is for setting the ETX, all what this does is adding the ETX from the CTp packet to the ETX (the one we added to the message/Oscilloscope packet) of the message.

After implementing all the above it was time to do some testing and start on the second goal of the project. The listener was only displaying information on the screen and this wasn`t going to help us with our experiment especially when we wanted to test with 1000 packets (screen will not show us everything we need).

Below are 3 examples of the outputs we got after we finished the implementation phase:

00 FF FF 00 00 13 00 93 00 00 00 04 00 03 00 C7 03 00 00 00 0A 01 C2 01 C2 01 C2

00 FF FF 00 00 13 00 93 00 00 00 05 00 03 00 3B 03 02 00 00 1E 01 A3 01 A3 01 A2

00 FF FF 00 00 13 00 93 00 00 00 05 00 03 00 2A 03 02 01 00 3C 01 B3 01 B3 01 B2

Purple is the id of the source generated the packet

Blue is the sequence number of the packets

Green is the 3 new fields created to hold the addresses of the hops/sensors

Grey is the new ETX field in the Oscilloscope struct

Red is the reading being generated by the source sensor

Line 1, this is the output of using one hop. Line 2 is the output from using 2 hops and Line 3 is the output from using 3 hops

Measurement Results

The approach followed for the all measurement was: -

1) Check the initial interval (1024).
2) Divide the interval by 2.
3) Test the performance with the new internal.
4) Follow the same procedure until the last stable interval is found.
   Stable referring to maintain the performance, receiving 95% of the total packets.

5) 1000 packets were used for testing.

Setup for 1 Hop

We had to check the output with different intervals and find the least interval with a throughput giving us at least 95%.

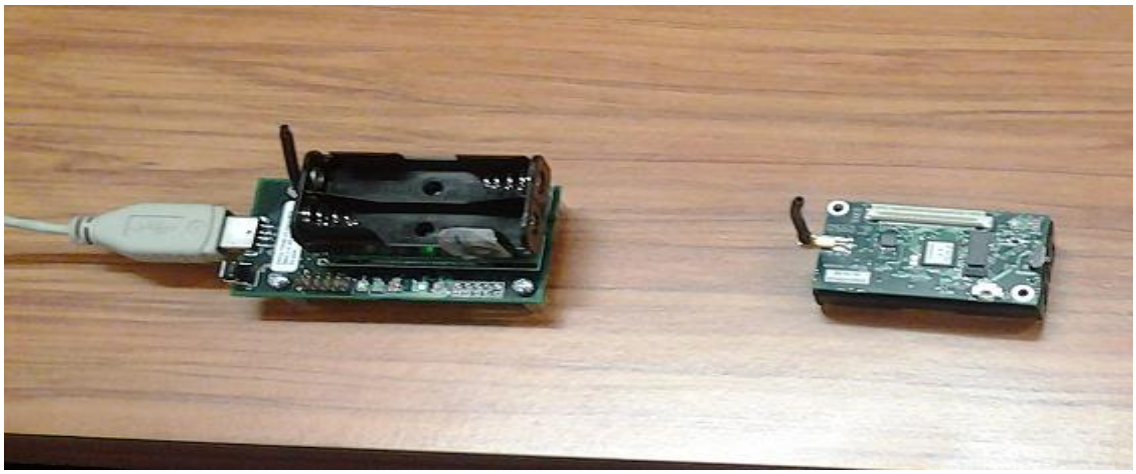

Figure 2 - One Hop Setup

Example output with interval 1024(100%)

00 FF FF 00 00 13 00 93 00 00 04 00 00 03 00 00 **03 00 00 00 0A** 01 C1 01 C1 01 C1
..
00 FF FF 00 00 13 00 93 00 00 04 00 00 03 03 E8 **03 00 00 00 0A** 01 C1 01 C1 01 C1

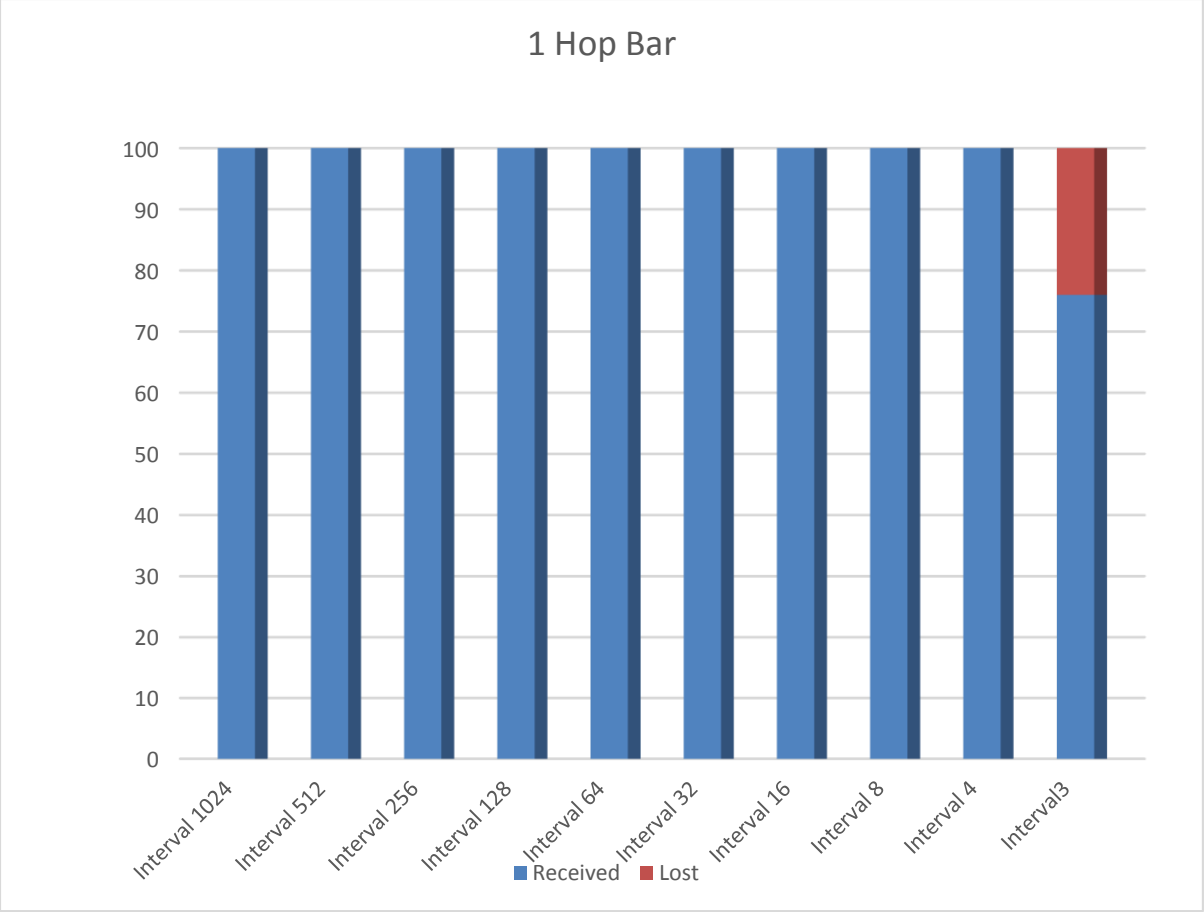

Figure 3 - Graph shows the Statistics at different intervals for 1 hop.

Setup for 2 Hops

Figure 4 - Two Hops Setup

Example output with Interval 1024

00 FF FF 00 00 13 00 93 00 00 00 20 00 03 00 01 **03 02 00 00 1E** 01 B0 01 B0 01 B1
..
00 FF FF 00 00 13 00 93 00 00 00 20 00 03 03 E8 **03 02 00 00 1E** 01 B1 01 B1 01 B1

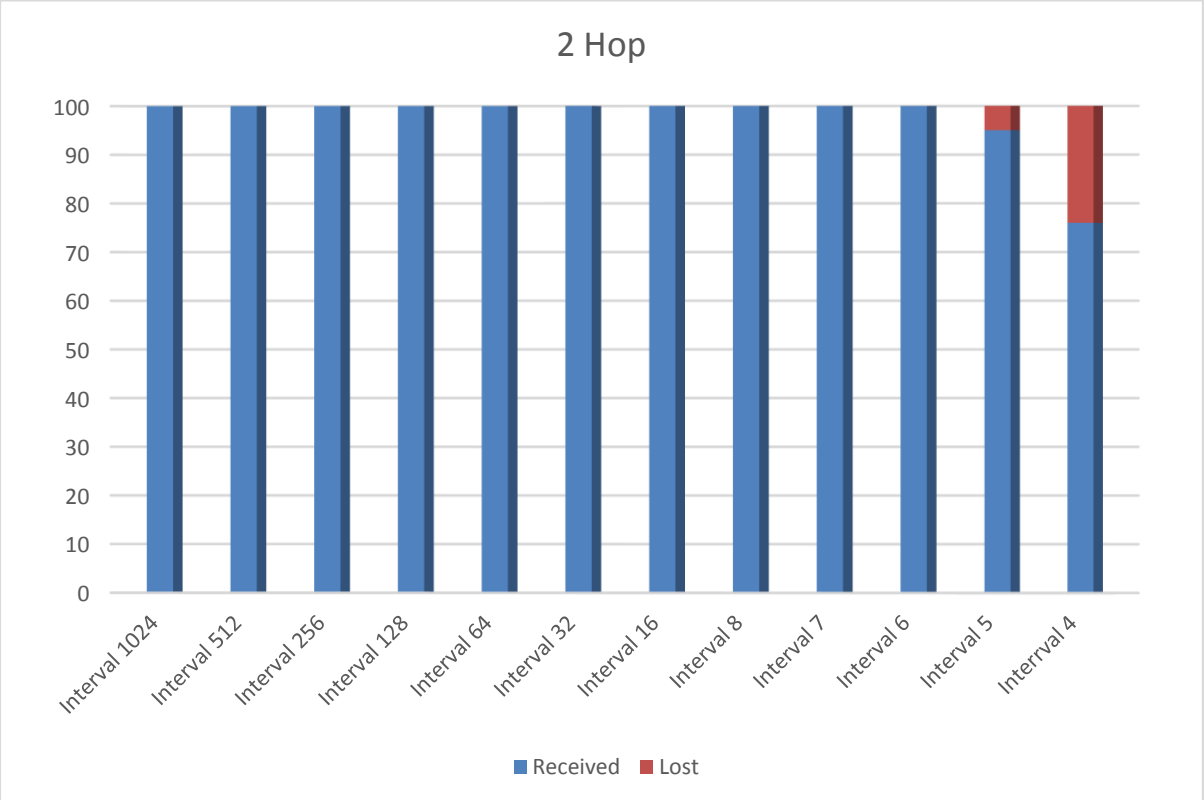

Figure 5 - Graph shows the Statistics at different intervals for 2 hops.

When the interval was 5 the throughput measured was 95% which we considered as an acceptable rate.

Setup for 3 Hops

This is the main testing stage as the number of hops increased and if the distance was right then lose would occur.

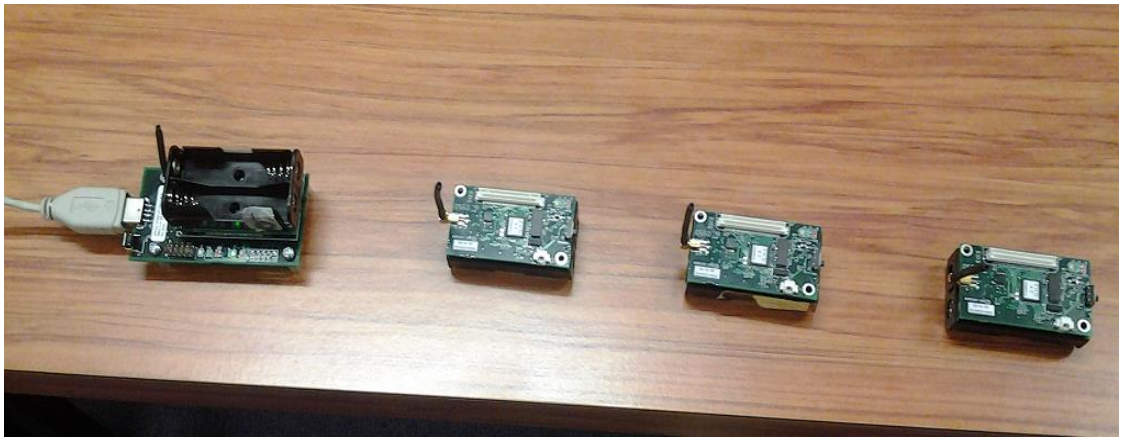

Figure 6 - Three Hops Setup

Example output with Interval 1024

```
00 FF FF 00 00 13 00 93 00 00 00 80 00 03 00 01 03 02 01 00 3C 01 D7 01 D8 01 D7
..
00 FF FF 00 00 13 00 93 00 00 00 80 00 03 03 E8 03 02 01 00 3C 01 D9 01 D9 01 D9
```

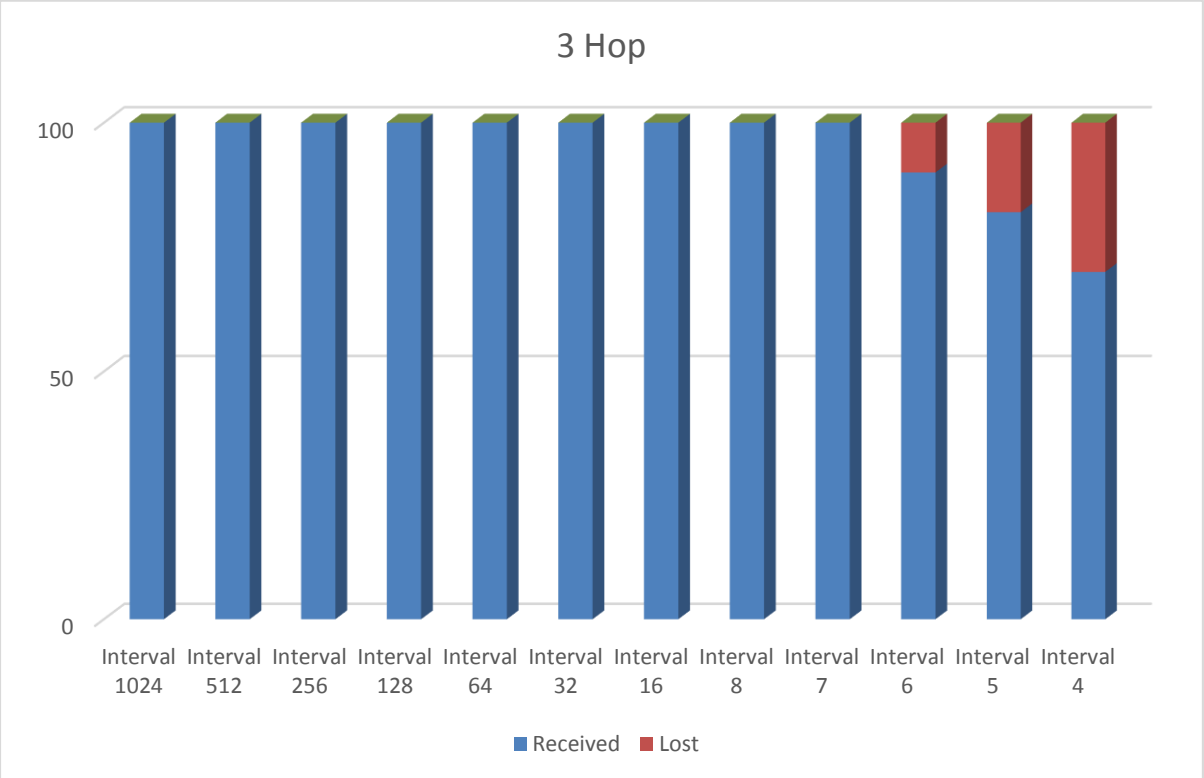

Figure 7 - Graph shows the Statistics at different intervals for 3 hops.

Results Discussion

The tested result hence made it clear that the performance varies on a number of factors: -

The number of hops involved.

- As the number of forwarders increase, the interval also needs to increase to stay in the limit of not losing packets more than 5%. Therefore this was noticed in the performance for output of 1, 2 and 3 hops.

The distance between sensors.

- The distance at which packets are accepted is different for all cases. As initially considering the maximum distance that accept packets is not true when reducing the intervals as packets were lost than. Hence to maintain the consistency of the packets the distance was set in a manner that the rule of forwarding and no shortcuts can be maintained.

The interval at which packets are generated. I.e. how many packets per second can be generated?

- The issue of losing the packets because of forwarders dropping the packet because of their own transmissions.

In other case, the sink not receiving the packet it is meant to failing the network flow. Hence the buffer space is vacant for the packets generated by the source only.

The throughput could be blocked by a source or the forwarders and sink which could have their own on-going transmission and to avoid collisions of packets we accept packets only from the source.

This also prevents dropping of a packet by the forwarder though there could be other reasons apart from these which occur in the loss of packets affecting the performance.

General obstacles like our hand or an object to block the node from transmitting packets as it causes distortion, the same was noticed while testing. Hence to avoid this we set the nodes first and then start the listener to display output.