Middle East Technical University
Department of Computer Engineering
Wireless Systems, Networks and Cybersecurity (WINS) Laboratory

Implementation of CSMA Plain Protocol using Ad Hoc Computing Framework

CENG 797: Ad Hoc Networks
2021-2022 Spring
Term Project Report

Prepared by
Derin Karadeniz
Student ID: e2598563
derin.karadeniz@metu.edu.tr
Computer Engineering
26 June 2022

Abstract

With the emergence of mobile smart devices, and technologies behind such as the Internet of Things, instant wireless communication has increased its importance. With their adaptation ability, Ad Hoc Networks provide a solution for this subject. In this project, the infrastructure of Ad Hoc Networks are examined and a preexisting protocol, Carrier Sense Multiple Access (CSMA), is implemented using Ad Hoc Computing Framework [1] in Python. Moreover a scenario based application which simulates a GPS module on a wireless device is created. The performance and function tests of the MAC protocol and the application are conducted successfully on B210 USRP devices.

Table of Contents

List of Figures

List of Tables

1 Introduction

Medium Access Control (MAC) is a concept of mechanism to manage data channels which are accessed by multiple communicating devices that send and receive data. If a channel is dedicated to a single device for only one-way communication, then this device will have full access to the channel all the time it requests. In other words, it does not need to contend for usage of the medium. However, in most of the cases, there are multiple users which try to simultaneously access to and use the same channel for sending or receiving data. In such case, it might easily lead to a collision and to data loss in network. To handle these conditions, devices run MAC protocols which regulate the access rules to the medium of communication.

In order to solve this issue, in this project, a MAC protocol working on an Ad Hoc Network, CSMA Plain is implemented using Ad Hoc Computing (AHC) Framework in Python.

Moreover, an application designed to work on military portable radios is implemented inspiring real world scenarios of military personnel might encounter on-site.

Implemented codes were tested on online AHC environment that works on a lab computer which manages four B210 USRP devices. This environment provides us to build an Ad Hoc Network while USRPs provide wireless communicating nodes which are necessary for the experiments. By this experiments, MAC layer and application layers are verified.

2 Background

2.1 Medium Access Control

There are tens of MAC protocols to solve the problem to decide who to use the medium at a specific time. In some of these algorithms and communication protocols, this decision is externally taken where a master node permits and dictates to the requesting node to speak. MIL-STD-1553 is an example of standard where a master as known as bus controller, arranges the bus access. On the other hand, most of the devices in networks, where unpredictable number of communicating nodes exists, prefer distributed algorithms. Distributed architectures of MAC protocols provide high scalability when compared to centralized ones.

In distributed protocols, devices which run the same algorithm competes to capture the channel to transmit its message. Consequently, those algorithms are called "Contention-based Protocols". Carrier Sense Multiple Access (CSMA) is one such protocol.

2.1.1 Carrier Sense Multiple Access

CSMA was introduced in 1975 by Kleinrock and Tobagi. It was developed to offer a solution to collision problems in radio communication channel. It aimed to perform better than ALOHA which puts data on channel whenever it desires. CSMA achieved to be better by sensing the presence of other nodes' transmissions in the channel. [2]

At the beginning of CSMA protocol, the algorithm checks if the channel is idle or not. If it is, then it transmits the packet with a probability p. If it is not idle, then it waits for a random time and checks the channel again. Nomenclature of the protocol slightly varies depending on the value of p. It is called non-persistent CSMA if p is equal to 0, 1-persistent if p is equal to 1 and p-persistent for other values of p. In this project, non-persistent CSMA

is implemented in Python as CSMA Plain using Ad Hoc Computing library [1].

2.2 Ad Hoc Computing Framework

In this project, all software infrastructure is provided by AHC framework for Python [1]. This library has an architecture which exhibits strong abstraction for the user. It allows to build mechanisms, protocols and applications which will run at different layers on USRP devices or on emulator. Library itself has a component-based structure where each component maps the layers of OSI model [3]. Each component is designed event-driven. State flows of the components are managed with event handlers. Thus, this framework allow user to easily experiment on Ad Hoc Networks and examine their manner.

3 Main Contributions

3.1 CSMA Plain

In this project, non-persistent CSMA is implemented in the name of "CSMA Plain" using AHC framework. Non-persistent CSMA is explained in details under 2.1.1. In addition to that, in our implementation, random wait time decision is modified. The program counts the number of unsuccessful trials for accessing to the channel, *CNT*. Wait coefficient, k_{wait} is uniform randomly chosen among a range of numbers from 1 to 2^{CNT}-1. Then, wait time is calculated by multiplying k_{wait} with window size determined by user. The protocol's state flow is depicted in figure 1 .

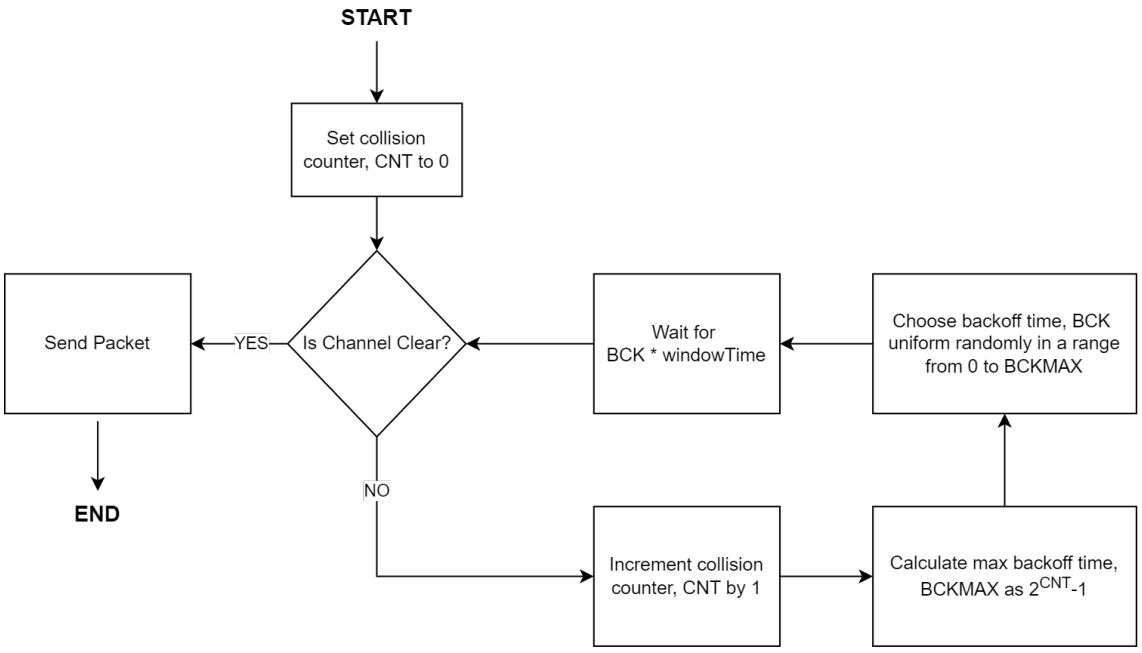

Figure 1 . Flowchart of CSMA Plain Protocol

3.2 A Military Radio Application

Within this project, an application which works using an ad hoc network, is developed. The application is designed considering a possible on-site scenario that an ad hoc network device

such as a military portable radio might encounter.

With the help of the AHC's flexibility, two modules which works within the application layer are implemented. The first module is "GPSHandler" which measures the location and makes necessary calculations. The second one is "Communicator" module which handles sent and received packets and which controls "GPSHandler". OSI model of the layer is indicated in figure 2 . Built-in segmentation and physical layers of AHC are used in the project while CSMA Plain is used instead of built-in MAC layer.

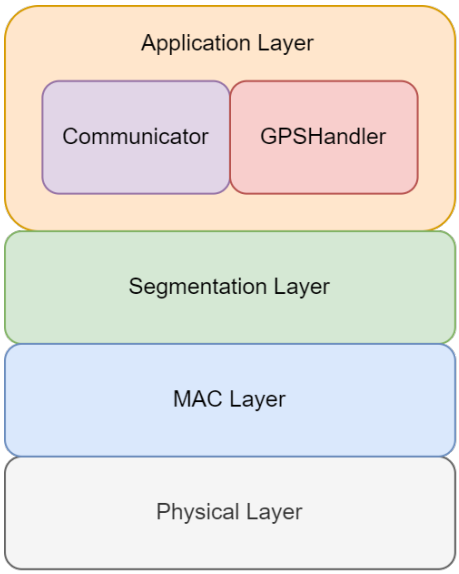

Figure 2 . OSI model of the Node

Pursuant to the scenario, a military personnel is stuck in a difficult situation and needs to send a confidential message to his/her foes who are in a predetermined range, R.

At the beginning, troubled personnel, node X, broadcasts its location for all foes. The device which received the location message, node Y, sends back its own location to node X. When X gets the location message. It calculates its distance to node Y and if this distance is less than or equal to a certain value, it starts to send the confidential message. State flow of the application is depicted in figure 3 .

4 Results and Discussion

4.1 Test Methodology

All tests which are designed to verify the MAC layer are conducted on AHC Framework using 2 of 4 B210 USRPs. The reason of using 2 USRPs is to minimize any delay and uncertainty due to increasing interference.

A new application layer is implemented for testing since the military radio application is not convenient for benchmarking. Normally, all nodes are hearing each other because of their actual topology. However, at the application layer, nodes only process the packets if it is intended to themselves by checking the "messageto" field in the message header.

To create an oncoming message traffic at sender which may lead to collisions, receiver sends an acknowledgment packet whenever it receives a data message. To prevent any

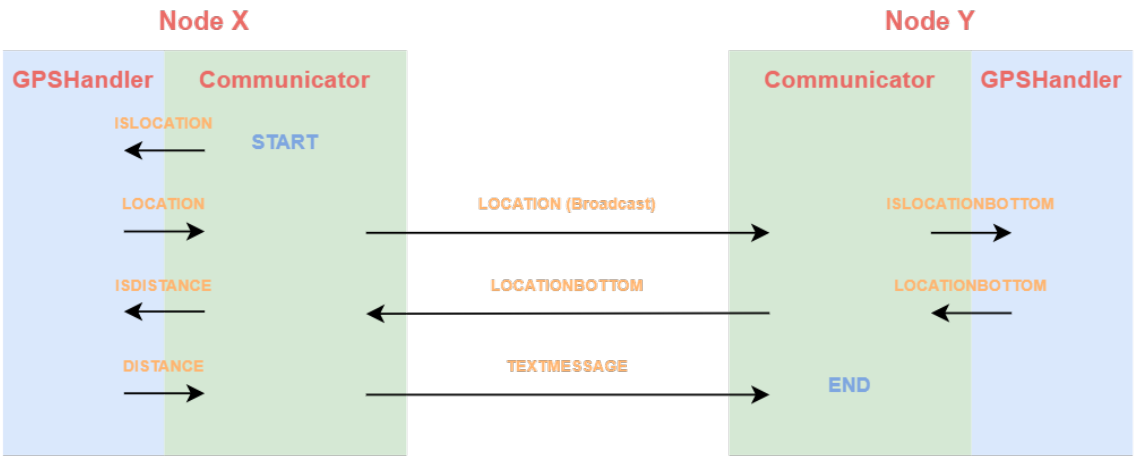

Figure 3 . State Flow of the Application

replay messages a sequence number starting from 0 is incrementally added to each data and acknowledgment packet. Data packets contain a payload of 64-bytes and acknowledgment packets contain a payload of 1-byte.

At every test run, 1000 data packets are sent and 1000 acknowledgment packets are expected.

4.2 Results

For the verification of the test outputs, the results in the original article of CSMA [2] is admitted as reference values.

There are some default parameters which had remained constant during the test. Those are:

- Transmission Frequency = 915 MHz,
- Channel Bandwidth = 2 MHz,
- Channel Clear Threshold = -70 dB,
- Offered Traffic / Throughput = 0.01 [2],

Then the variable parameters are:

- Window Size, t_{window},
- Time between packets' transmission, t_{packet},
- Saturation value of Backoff Coefficient, BCK_{max},

Offered Traffic-Throughput ratio corresponds to t_{window}/t_{packet}. BCK_{max} is the maximum value that BCK can have.

The control parameters are calculated as follows:

- Data Success Ratio = # of Received Data Packets / # of Total Sent Data Packets,
- Ack Success Ratio = # of Received Ack Packets / # of Received Data Packets,

Tests were performed for 4 cases below:

- **Case 1:**
  t_{packet} = 1 second,

$t_{window} = 0.01$ seconds,
$BCK_{max} = 6$,

- **Case 2:**
  $t_{packet} = 0.1$ seconds,
  $t_{window} = 0.001$ seconds,
  $BCK_{max} = 6$,
- **Case 3:**
  $t_{packet} = 0.01$ seconds,
  $t_{window} = 0.0001$ seconds,
  $BCK_{max} = 7$,
- **Case 4:**
  $t_{packet} = 0.01$ seconds,
  $t_{window} = 0.00001$ seconds,
  $BCK_{max} = 7$,

In the first three cases, Offered Traffic-Throughput ratio (= t_{window}/t_{packet}) is equal to 0.01 as it should be. However, it is 0.001 for the last case. BCK_{max} is equal to 6 for first the two case, but, it is equal to 7 at two last cases. The reason behind this is, in this system, 0.0001 and 0.00001 seconds window sizes are small values for the calculation of the wait time between retrials.

The test results of the cases are presented in the table 1.

	Data Success%	Ack Success%	Avg Success%	Goodput[Byte/s]	# of Retrials
Case 1	99	100	99.49	63.36	81
Case 2	64.3	44.8	56.7	411.52	1500
Case 3	32.7	0.0	24.6	209.28	3267
Case 4	24.9	0.0	19.9	159.36	4617

Table 1. Results of the Test Cases.

4.3 Discussion

When the results are compared to the results in the figure 4 which belongs to the original article, it is seen that the results of non-persistent CSMA or CSMA Plain are similar to each other. Even though, the numbers does not exactly match in both data, the maximum throughput point corresponds to the same x-value where the channel traffic is equal to 10 packets per second. In the figure 4 , maximum throughput is 80% at G = 10 and in our results at G = 10 or at $t_{packet} = 0.1$, goodput is at its maximum value with 56.7% success average.

In case 1, we obtained the highest success rate because the data rate through the channel was quite low with 1 packet/second. This value does not creates any collision which would leads to packet loss.

In case 3 and 4, the results that we have got are unsatisfactory since the success rate therefore the packet loss is very high. Notice that the acknowledgment packets are totally missing even in the case 4 where we have provided more precise wait times to the algorithm.

This results may arise from the physical limitations of existing USRP devices which might have high interference. The problems can be originated from the implementations and from AHC framework as well.

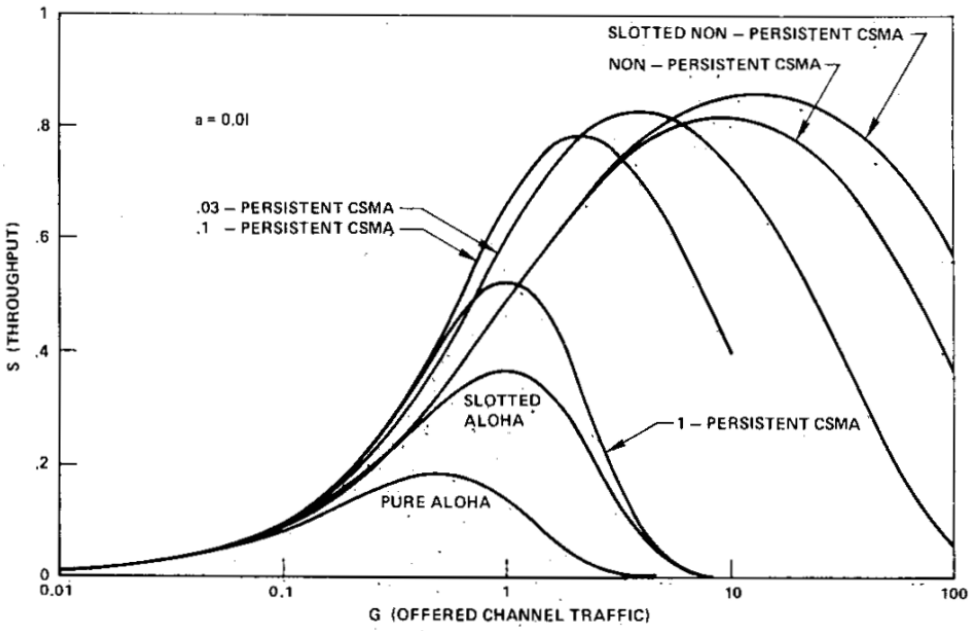

Figure 4 . Throughput for the various access modes [2]

In the future, this experiments can be remade under better conditions.

5 Conclusion

In conclusion, non-persistent CSMA or CSMA Plain and a scenario-based application are successfully implemented in Python for AHC Framework. Even though, there are differences in results obtained from the experiments for CSMA, there exists a correlation between the expected reference values 4 and the test results. The reason of the alteration in the values might be due to some physical phenomenon or to structural errors in the software. However, the source of the differences is yet to be searched.

References

[1] CengWins. (2022) Ad hoc computing framework. [Online]. Available: https://pypi.org/project/adhoccomputing/

[2] L. Kleinrock and F. Tobagi, "Packet switching in radio channels: Part i - carrier sense multiple-access modes and their throughput-delay characteristics," *IEEE Transactions on Communications*, vol. 23, no. 12, pp. 1400--1416, 1975.

[3] N. Briscoe. (2012) Understanding the osi 7-layer model. [Online]. Available: http://sdcc.vn/template/266_osi7layer_t04124.pdf

Appendix A Project Repository Link

Link to the project repository: github.com/derinkaradeniz/CENG797Project/tree/main/Project