Wireless Sensor Network
White Paper

Performance Evaluation of IEEE 802.15.4 based Wireless Sensor Networks

<table>
<thead>
<tr>
<th>Work</th>
<th>Author</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created</td>
<td>TK, SS</td>
<td>10th Jun 2012</td>
</tr>
<tr>
<td>Review</td>
<td>SS, PV</td>
<td>20th June 2012</td>
</tr>
<tr>
<td>Update</td>
<td>TK</td>
<td>28th June 2012</td>
</tr>
<tr>
<td>Review</td>
<td>SS, PV</td>
<td>10th July 2012</td>
</tr>
<tr>
<td>Update</td>
<td>TK</td>
<td>11th July 2012</td>
</tr>
</tbody>
</table>

Authors:

(TK) T. Tarun Krishna IIT Madras
(PV) Pranav Viswanathan Tetcos
(SS) Shashikanth Suman Tetcos

©2012. Tetcos. This document is protected by copyright, all rights reserved.
Contact: sales@tetcos.com
Table of Contents:

TABLE OF CONTENTS:.................................................................................................................. 2
1. ABSTRACT................................................................................................................................. 3
2. INTRODUCTION TO WSN......................................................................................................... 3
  2.1 PAN COORDINATOR............................................................................................................... 3
  2.2 SUPER FRAME STRUCTURE.................................................................................................. 3
  2.3 CSMA/CA ALGORITHM......................................................................................................... 4
      KEY PACKET PARAMETERS: ................................................................................................. 4
      ALGORITHM FLOW: ............................................................................................................... 5
3. INTRODUCTION TO NETSIM...................................................................................................... 6
4. MODEL ........................................................................................................................................ 6
5. WSN SIMULATION IN NETSIM.................................................................................................. 6
      CONFIGURATION FILE: .......................................................................................................... 7
      STATIC ROUTING: .................................................................................................................. 7
6. CALCULATION.............................................................................................................................. 8
      THROUGHPUT: ....................................................................................................................... 8
      ATTEMPT RATE: .................................................................................................................... 8
      DISCARD PROBABILITY: ......................................................................................................... 8
      PAYLOAD AND PLOTS: ........................................................................................................ 8
7. THEORETICAL ANALYSIS FOR 1 NODE .................................................................................... 9
8. RESULTS ...................................................................................................................................... 10
9. INFERENCE.................................................................................................................................. 13
BIBLIOGRAPHY............................................................................................................................. 13
APPENDIX 1: PYTHON PLOTS .................................................................................................... 13
1. Abstract

In this paper, we compare NetSim’s performance metrics with a theoretical analysis of the IEEE 802.15.4 standard. It’s well known that the 802.15.4 protocol suffers from sharp throughput drops at higher loads. To enhance the same we implemented modifications to the backoff parameters per the IEEE reference paper and find a significant improvement in the saturation throughput at higher loads.

2. Introduction to WSN

IEEE 802.15.4 based wireless sensor networks has witnessed explosive growth in the recent past, driven by a) its position independent sensing capabilities even in toxic and inaccessible regions to humans & b) the low cost of sensors and c) a very long field lifetime given their low power consumption.

WSN has typically two types of devices: FFD and RFD. Full Functional Devices (FFD) can sense, act as PAN coordinator, route packets and can communicate with other FFD’s and RFD’s. Reduced Functional Device (RFD) can sense and communicate only with FFD’s.

2.1 PAN Coordinator

PAN Coordinator is the principal controller in a WPAN and there is only one PAN Coordinator in a WSN. If the PAN Coordinator uses Beacon enabled mode then it manages communication between devices using superframes and nodes use Slotted CSMA/CA algorithm for transmitting packets else if the PAN Coordinator uses non-Beacon enabled mode then nodes use Unslotted CSMA/CA and there are neither superframes nor beacons.

2.2 Super Frame Structure

Superframe is contained in a beacon Interval bounded by two beacon frames, and has an active period and an inactive period. The coordinator interacts with its PAN during the active period, and enters in a low power mode (sleep) during the inactive period. The structure of a superframe is defined by two parameters

1) $macBeaconOrder(BO) , 0 \leq BO \leq 14$;
2) $macSuperframeOrder(SO), 0 \leq SO \leq BO \leq 14$;

BO describes the beacon interval (BI) at which the coordinator must transmit beacon frames and is given by:

$$BI = aBaseSuperframeDuration \times 2^{BO}$$ symbols.

SO describes the Active Period and is discussed next.
Beacon Frames:
Beacons occupy the first slot in Super Frame and are used to synchronize the attached devices, to identify the PAN, and to describe the structure of the superframes. Beacon frames are transmitted periodically to announce the presence of a network.

Active Period:
Active period consists of two periods: Contention Access Period (CAP) and Contention Free Period (CFP). Length of active period is determined by the $macSuperframeOrder(SO)$.

Active Period Length = $aBaseSuperframeDuration \times 2^{SO}$ symbols

Contention Access Period (CAP):
In CAP, all nodes with packets to transmit compete for the channel and follow Slotted CSMA/CA algorithm. A sufficient portion of the CAP remains for contention-based access of other networked devices or new devices wishing to join the network.

Contention Free Period (CFP):
CFP is used by the PAN coordinator to allocate guaranteed time slots (GTS) for nodes requiring low latency. A node issues GTS allocation request to the PAN coordinator, which then allocates available GTS in the next superframe to such nodes and indicates this in the beacons broadcasted before the next superframe. In CFP of next superframe, the node requested for GTS can transmit during its GTS, if it is allocated, without contention from other devices. CFP can be disabled by disabling GTS.

Inactive Period:
In the inactive period, the coordinator may enter a low-power mode and doesn’t interact with its PAN which helps in reduced energy consumption and so extends the network lifetime. Inactive period can be remove by setting SuperFrame Order same as Beacon Order i.e., $BO = SO$

2.3 CSMA/CA Algorithm

Key Packet parameters:
Each packet is characterized by 3 variables NB, CW & BE

<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB</td>
<td>Number of backoffs the node has underwent while attempting the current transmission, initialized to 0 before every new transmission</td>
</tr>
<tr>
<td>CW</td>
<td>Contention Window length, defines the number of backoff periods that needs to be clear of channel activity before the transmission can start. Initialized to 2 or 1 before each transmission attempt and reset to 2 or 1 each time the channel is sensed to be busy and based on Slotted or Unslotted CSMA/CA</td>
</tr>
<tr>
<td>BE</td>
<td>Backoff exponent is related to how many backoff periods $[0,2^{BE} – 1]$ a device has to wait before attempting to assess the channel.</td>
</tr>
</tbody>
</table>

©2012. Tetcos. This document is protected by copyright, all rights reserved. Contact: sales@tetcos.com
<table>
<thead>
<tr>
<th>Step</th>
<th>Slotted CSMA/CA</th>
<th>Unslotted CSMA/CA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>NB(=0), CW(=2), BE are initialized.</td>
<td>NB(=0), CW(=1), BE are initialized.</td>
</tr>
<tr>
<td></td>
<td>Locate the backoff boundary and attempt for transmission or channel assessments (CCA) at the start of the slot.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>MAC layer delays for a random number of backoff periods in the range ([0, 2^{BE} - 1]) slots</td>
<td>Same</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>If the frame transmission and acknowledgment can be completed before the end of the current CAP then MAC sub layer will request PHY layer to perform CCA, else it will wait until the start of the CAP in the next SuperFrame and repeat the evaluation.</td>
<td>MAC will request PHY layer to perform CCA.</td>
</tr>
<tr>
<td><strong>Step 4</strong> (CCA failed i.e., Channel is busy)</td>
<td>MAC sub layer increments both NB and BE by one and sets CW to its default value (2), ensuring that BE shall be no more than (a_{MaxBE}). If NB is greater than (max_{MaxCSMABackoffs}) then the packet is discarded else return to step 2.</td>
<td>Same and CW is reset to 1.</td>
</tr>
<tr>
<td><strong>Step 5</strong> (CCA success i.e., Channel is Idle)</td>
<td>Decrement CW by one and if CW is zero then start transmission else return to step 3</td>
<td>Same.</td>
</tr>
</tbody>
</table>

A HOL (Head Of Line) packet begins with a random backoff sampled uniformly from \([0, 2^{BE} - 1]\) slots (1 slot = 20 \(T_s\)), followed by a CCA with initial backoff exponent (BE) as \(mac_{MinBE}\). Each CCA failure starts a new backoff process with the backoff exponent raised by one without exceeding \(a_{MaxBE}\). The maximum number of CCA failures for a packet is governed by \(mac_{MaxCSMABackoffs}\), exceeding which the packet is discarded at the MAC layer.

A successful CCA is followed by the radio turnaround time and then packet transmission. In case of Slotted CSMA/CA two consecutive CCA have to be successful for a packet transmission to begin.

After successfully receiving the packet, without any collisions or corruption due to PHY layer noise, the receiver sends an ACK packet to transmitter after waiting for the radio turnaround time. A failed packet reception causes no ACK generation.

After transmission, transmitter waits for \(mac_{AckWaitDuration}\) to receive ACK. If it does then it starts transmission of next packet else it infers that the packet has failed and retransmits the packet for a maximum of \(a_{MaxFrameRetries}\) times before discarding it at the MAC layer.
3. Introduction to NetSim

NetSim is a popular discrete event network simulator software used for academic research. NetSim’s development environment platform allows users to develop custom codes, simulate their models and statistically analyze performance metrics.

NetSim’s WSN library contains C source code for the primitives, and the configuration files are available as xml files. In this paper we have modified the source code and created custom configurations which were linked & de-bugged using NetSim’s development environment.

4. Model

Using [1] as reference we have modified NetSim WSN to meet the following assumptions. The results are then compared with ns2 [1].

<table>
<thead>
<tr>
<th>Assumptions (ref [1])</th>
<th>IEEE 802.15.4 Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Star Network with sensors at the tips.</td>
<td>Nodes can be anywhere.</td>
</tr>
<tr>
<td>Infinite backlog of packets at each node.</td>
<td>Sensor gets a packet only when it senses agent.</td>
</tr>
<tr>
<td>Static Routing.</td>
<td></td>
</tr>
<tr>
<td>Packets flow only from the sensors to the PAN Coordinator.</td>
<td>Packets can flow from one sensor (RFD or FFD) to another(FFD) or PAN Coordinator.</td>
</tr>
<tr>
<td>Data requests from nodes to the PAN Coordinator are not considered.</td>
<td>Data transfer can be to/from PAN coordinator.</td>
</tr>
<tr>
<td>No Inactive period and CFP (i.e., BO equal to SO and GTSSs are disabled).</td>
<td>SuperFrame has both active (CAP + CFP) and inactive periods.</td>
</tr>
<tr>
<td>All of the channel time is used for CSMA/CA based access (i.e., ignore beacon transmission time by using large SuperFrame duration).</td>
<td></td>
</tr>
<tr>
<td>None of the devices disassociates during the whole traffic flow and communication failures never cause a device to conclude that node has been orphaned.</td>
<td>Disassociation process can be initiated by a PAN coordinator or the device itself. A device may conclude that it is an orphan device if a predetermined number of transmission attempts have failed.</td>
</tr>
</tbody>
</table>

5. WSN Simulation in NetSim

In NetSim 6.1, WSN has been developed based on the IEEE 802.15.4 standard and is an Agent based model i.e., sensors are placed on the simulation environment (either using Uniform placement or via drag and drop) and they sense the agents based on the sensor’s range and agent’s position.

We can simulate several complex scenarios using the Configuration.xml file located in windows temporary (%temp%\NetSim) directory and using the primitives of WSN library of NetSim 6.1.
Configuration file:

Star network is constructed by placing the sensors uniformly on a circle with PAN coordinator at the centre as shown in the figure.

Configuration.xml has to be modified to simulate the scenario as shown below.

Number of devices (Sensors and PAN coordinator) is defined in the tag:

```xml
<Tetcos_NetSim -> Network_Configuration -> Device_Count
<DeviceCount="1"Type="SinkNode"/>
<DeviceCount="5"Type="Sensor"/>
```

SinkNode and Sensors main properties are to be set in the following tags:

- **X and Y Position:** (SinkNode and Sensors)
  ```xml
  <Tetcos_NetSim -> Network_Configuration -> Device_Configuration -> Device-> Pos_2D
  </xml`

- **Physical layer properties:** (SinkNode)
  ```xml
  <DeviceId="1">Port_Count="1"Type="SinkNode">PortId="1">
  <LayerType="Physical_Layer">IEEE802_15_4_Phy_Properties
  </xml`

- **Physical layer properties:** (Sensor)
  ```xml
  <DeviceId="1">Port_Count="1"Type="Sensor">PortId="1">
  <LayerType="DataLink_Layer">IEEE802_15_4_Mac_Properties
  </xml`

- **MAC layer properties:** (SinkNode)
  ```xml
  <DeviceId="1">Port_Count="1"Type="SinkNode">PortId="1">
  <LayerType="Network_Layer">
  </xml`

- **IP Address:** (Sensors)
  ```xml
  <DeviceId="1">Port_Count="1"Type="Sensor">PortId="1">
  <LayerType="Network_Layer">
  </xml`

**Static Routing:**

Static routing can be implemented using the **Check Route Found** primitive. We add the static routes for the current scenario as shown below, before checking if TempTable is not NULL.

```c
if(!TempTable)
{
    TempTable = fnpAllocateMemory(1, sizeof *TempTable);
    TempTable->n_M_SourceNode = n NP_DeviceId;
    TempTable->n_M_DestinationNode = 1;TempTable->n_M_HopCount = 1;
    TempTable->pstru_ROUTEPath = fnpAllocateMemory (1, sizeof ROUTETABLEPath));
    TempTable->pstru_ROUTEPath->stru_Path[0] = n NP_DeviceId;
    TempTable->pstru_ROUTEPath->stru_Path[1] = 1;
    NETWORK->pstruNodeList[n NP_DeviceId-1]->pstru_NC_Device_NetworkLayer->pstruDSRNetworkLayer->pstruRouteTable = TempTable;
}
```
6. Calculation

Throughput:

Definition: Average rate of successful packet delivery. In terms of NetSim’s metrics it is:

\[
\text{ThroughPut} = \frac{\text{Frames Transmitted} - \text{Frames Errored}}{\text{Simulation Time}}
\]

Attempt Rate:

Definition: Number of backoffs over the time spent in backoffs (per node basis):

\[
\beta = \frac{\text{Number of Backoffs or attempts}}{\text{Time (in backoff slots) spent in backoff + CCA}}
\]

Attempt rate gives the number of attempts made by sensor in the backoff times i.e., the probability that a sensor attempts a CCA in a slot given that it has a packet.

Backoff is always followed by CCA, which can be success or failure. A successful attempt (successful CCA) involves a frame transmission and failed attempt means CCA failure and so number of backoffs or attempts is the sum of failed CCA and frames transmitted.

\[
\text{Number of Backoffs} = \text{Frames Transmitted} + \text{Failed CCA count}
\]

Each packet transmission is followed by radio turnaround time, ACK packet and IFS, so time spent in backoff is the simulation time without the packet transmission time, radio turnaround time, ACK, IFS and the beacon transmission time.

\[
\text{Time spent in backoff + CCA} = \text{Simulation Time} - \text{Beacon Transmission Time} - \{\text{Frames Transmitted} * (\text{Frame Transmission Time} + \text{Turnaround Time} + \text{ACK Time} + \text{IFS})\}
\]

Average of \(\beta_i\{i \in \text{set of sensors}\}\) gives the probability that a sensor attempts in a backoff slot.

Discard Probability:

Definition: Probability of packet being discarded.

\[
P_{\text{discard}} = \frac{\text{Frames Discarded}}{\text{Frames Transmitted (ignoring retransmissions)}}
\]

Payload and Plots:

Payload at Application layer is 4 bytes, overheads are set according to standard (20IP+13MAC+6PHY). Using the Metrics.txt files obtained from the simulations, plots are plotted. (Ref Appendix)
7. Theoretical Analysis for 1 node

Consider scenario with 1 sensor, so there will be no collisions. As discussed in the CSMA/CA algorithm a successful packet transmission involves Random Backoff, CCA, turnaround time, packet transmission, turnaround time, ACK packet + IFS.

Random backoffs generated are governed by BE which is incremented every time CCA fails. As we have no collisions, BE is not incremented and so random backoffs are always in the range $[0, 2^{BE} - 1]$ slots (1 slot = 20 $T_s$ or 20 symbols) with BE=3. So random backoffs generated are always between [0,7] slots.

In slotted CSMA/CA, for packet transmission to start the channel needs to be clear of activity in two successive CCAs. Note that there will no collisions or CCA failures since the scenario has only one sensor. Each CCA is 8 symbols long and starts at the backoff boundary. After the 2$^{nd}$ CCA we need radio turnaround time, which is 12 symbols long, to change from receiver to transmitter before packet transmission starts. So 1$^{st}$ CCA takes only 8 Symbols in a backoff slot and 2$^{nd}$ CCA starts at the next backoff boundary followed by turnaround time. So 2$^{nd}$ CCA + turnaround time takes 20 Symbols or 1 slot and therefore (1$^{st}$ CCA) + (2$^{nd}$ CCA + turnaround time) takes 2 slots.

Packet size is 43 Bytes and Data rate defined by 802.15.4 for 2.4GHz bandwidth is 250Kbps and so time taken to transmit the packet is $\frac{43 \times 8}{250 + 1000} = 1.376 ms = \frac{1.376}{0.32} slots = 4.3 slots$. Each slot is 20 symbols and each symbol takes 16 $\mu$s and so each slot is 20*16 $\mu$s=0.32ms. After the packet transmission, sensor waits for the ACK packet and the state of the transceiver has to be changed from transmitter to receiver which is 12 Symbols = 0.6 slots. This state change can be completed in the last slot of packet transmission, so packet transmission + turnaround time is 5 slots.

ACK packet is 11 bytes and so takes $\frac{11 \times 8}{250 + 1000} = 0.352 ms = \frac{0.352}{0.32} slots = 1.1 slots$ for transmission. After receiving ACK packet, sensor waits for 2 slots (i.e., IFS, Inter Frame Spacing so that MAC sub-layer has sufficient time to process the data received from Physical layer) before attempting again. The first slot in IFS is taken as the last slot in ACK packet as it takes only 2 Symbols. So ACK packet + IFS takes 3 slots.

| Random Backoffs     | 3.5 slots
| 2 CCA + Turnaround Time | 2 slots
| Packet Transmission + Turnaround Time | 5 slots
| ACK Packet +IFS | 3 slots
| Total | 13.5 slots

Throughput: 1 packet is transmitted for every 13.5 slots, throughput = $\frac{1 \text{ packet}}{13.5 \text{ slots}} = 231.48 \text{ Kbps}$

Attempt Rate: 1 attempt is made in 3.5slot random backoffs and 2slot CCAs, so $\beta = \frac{1}{3.5+2} = 0.18$

Discard Probability: For 1 node case there will be no collisions, so $P_{\text{discard}} = 0$
8. Results

Simulations are run for 100s with BO as 14 and parameters are calculated as given above.
Simulation is run for 100s with BO as 14 and backoff multiplier (p) as 3.

<table>
<thead>
<tr>
<th>NetSim Simulation</th>
<th>Reference Paper [1]</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Throughput vs Number of nodes" /></td>
<td><img src="image2.png" alt="Throughput vs number of nodes" /></td>
</tr>
<tr>
<td><img src="image3.png" alt="Attempt Rate vs Number of nodes" /></td>
<td><img src="image4.png" alt="Attempt Rate vs number of nodes" /></td>
</tr>
<tr>
<td><img src="image5.png" alt="Pdiscard vs Number of nodes" /></td>
<td><img src="image6.png" alt="Pdiscard vs number of nodes" /></td>
</tr>
</tbody>
</table>
Simulation is run for 100s with BO as 14 and macMinBE as 5 and MaxBE as 7

<table>
<thead>
<tr>
<th>NetSim Simulation</th>
<th>Reference Paper [1]</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Throughput vs Number of nodes" /></td>
<td><img src="image2" alt="Throughput vs Number of nodes" /></td>
</tr>
<tr>
<td><img src="image3" alt="Attempt Rate vs Number of nodes" /></td>
<td><img src="image4" alt="Attempt Rate vs Number of nodes" /></td>
</tr>
<tr>
<td><img src="image5" alt="Pdiscard vs Number of nodes" /></td>
<td><img src="image6" alt="Pdiscard vs Number of nodes" /></td>
</tr>
</tbody>
</table>
9. Inference

From the plots, we see that changing the backoff parameters (backoff multiplier, ranges of backoff exponent i.e., macMinBE and MaxBE), leads to a higher throughput as compared with the 802.15.4 standard. As the attempt rate ($\beta$) increases, probability of a packet collision increases and so throughput decreases. Changing the backoff parameters decreases the attempt rate ($\beta$), as we notice in the plots, and therefore the collisions decrease leading to higher throughput. This decrease in collisions is evident from the discard probability graphs.

NetSim simulation results validate the theoretical analysis ([1]) of improved throughputs with modifications to the 802.15.4 Standard’s back-off parameters. We believe these suggestions could be investigated by IEEE 802 international task force prior to releasing the newer revisions of the standard.

Bibliography


Appendix 1: Python Plots

Simulations are run for different scenarios (from 1 node to 60 nodes) and all the metrics files are saved. Using python’s matplotlib module we plot throughput plot as shown in below source code.

```python
# Count the packets attempted transmission (ignoring retransmissions)
# Count the packets discarded i.e., first row in *_Discard.txt as -1 or -2
import os, math, string, matplotlib.pyplot as plt, numpy as np
def CorrectSimTime(Sim_Time,BO):
    SuperFrameLength = 15.36* math.pow(2,BO);
    BeaconLength = SuperFrameLength/16.0;
    if(Sim_Time - SuperFrameLength*int(Sim_Time/SuperFrameLength) > BeaconLength):
        NewSim_Time = Sim_Time - BeaconLength*(int(Sim_Time/SuperFrameLength)+1);
    else:
        NewSim_Time = Sim_Time - BeaconLength*(int(Sim_Time/SuperFrameLength));
    return NewSim_Time;
def PlotGraph(Y1,X1,Ytitle,Xtitle,Title,FigPath,YLimit,plotShow):
    sortedMetrics = sorted(dict(zip(X1,Y1)).items(), key=lambda x: x[0]);
    i=0;X=[];Y=[];
    while i < len(sortedMetrics):
        X.append(sortedMetrics[i][0]);Y.append(sortedMetrics[i][1]);
        i = i+1;
    plt.ylim([0,YLimit]);plt.plot(X,Y, ',-')
    plt.xlabel(Xtitle);plt.ylabel(Ytitle);
    plt.title(Title);plt.savefig(FigPath);
    if plotShow == 1:
        plt.show();
    plt.clf();
```

©2012. Tetcos. This document is protected by copyright, all rights reserved.
Contact: sales@tetcos.com
```python
plotShow = 1; PacketSize = 43; BO = 14;
fp = open('Path.txt', 'r'); Path = fp.readline() + '\n'; fp.close();

dllEndArray = [];
dllEndCount = 0;
for filename in os.listdir(Path+'Metrics'):
    try:
        dllEndArray[index(filename.rsplit('_')[0])];
    except:
        dllEndCount = dllEndCount + 1;
dllEndArray.append(filename.rsplit('_')[0]);

if dllEndCount == 1:
    Nodes = [[], []]; Beta = [[], []]; Throughput = [[], []];
if dllEndCount == 2:
    Nodes = [[], [], []]; Pdiscard = [[], [], []]; Beta = [[], [], []]; Throughput = [[], [], []];
if dllEndCount == 3:
    Nodes = [[], [], [], []]; Pdiscard = [[], [], [], []]; Beta = [[], [], [], []]; Throughput = [[], [], [], []];

Betaylimit = input('Beta y limit:');
Dir = r'Discard';
FrameTxTime = 8; #(Packet_Size*0.1)+2; # in Milliseconds
SlotTime = 20*0.016; # in Milliseconds

for filename in os.listdir(Path+Dir):
    NumberOfNodes = int(filename.rsplit('_')[1]);
    print 'Reading '+ Dir+'\n' + filename;
    fp = open(Path+Dir+'\n' + filename, 'r');
    dataAll = fp.readlines(); fp.close();
    fp = open(Path+'Metrics' + '\n' + filename.replace('Discard', 'Metrics') + 'r');
    Sim_Time = int(fp.readline().rsplit(',')[1]);
    fp.close();
    Frames = [1]; FramesSuccesCount = 0; FramesDiscarded = 0;
    FramesTransmitted = [0]*NumberOfNodes;
    BackoffCount = [0]*NumberOfNodes;
    for data in dataAll:
        simData = data.rsplit('\n');
        NodeNumber = int(data[2])-2;
        if int(data[0]) < 0:
            FramesDiscarded = 1;
        BackoffCount[NodeNumber] = int(data[3]);
        if int(data[0]) == 0:
            FramesSuccesCount = 1;
        BackoffCount[NodeNumber] = int(data[3])+1;
        if int(data[0]) >= 0:
            FramesTransmitted[NodeNumber] = 1;
    dllIndex = dllEndArray.index(filename.rsplit('_')[0]);
    Nodes[dllIndex].append(NumberOfNodes);

Pdiscard[dllIndex].append(FramesDiscarded*(1.0/(FramesSuccesCount+FramesDiscarded)));
Beta[dllIndex].append(np.average((1.0*np.array(BackoffCount))/(CorrectSimTime(Sim_Time, BO)/SlotTime-(np.array(FramesTransmitted)*FrameTxTime))));
Throughput[dllIndex].append(np.average(FramesSuccesCount)*1000.0/(CorrectSimTime(Sim_Time, BO)));

avgThroughput = sum(np.array(Throughput))/len(Nodes);
PlotGraph(avgThroughput, Nodes[0], 'Throughput', 'Number of Nodes', 'Throughput vs Number of nodes', Path='Throughput.png', 300, plotShow);

avgPdiscard = sum(np.array(Pdiscard))/len(Nodes);
PlotGraph(avgPdiscard, Nodes[0], 'Pdiscard', 'Number of Nodes', 'Pdiscard vs Number of nodes', Path='Pdiscard.png', 1, plotShow);

avgBeta = sum(np.array(Beta))/len(Nodes);
PlotGraph(avgBeta, Nodes[0], 'Beta', 'Number of Nodes', 'Attempt Rate vs Number of nodes', Path='Beta.png', Betaylimit, plotShow);
```