

Tetcos

Wireless Sensor Network White Paper

Performance Evaluation of Multi-hop Wireless Sensor Networks

Authors:

Vikas Neelam *IIT Madras*

Pranav Vishwanathan *Tetcos*

Shashikanth Suman *Tetcos*

Table of Contents:

	Description	Page No.
1.	Abstract	2
2.	The IEEE 802.15.4 for wireless Sensor Networks	2
2.1.	The IEEE 802.15.4 Physical Layer	2
2.2.	The IEEE 802.15.4 Medium Access Control	3
2.3	Introduction to WSN	3
2.4	PAN Coordinator	4
2.5	Unslotted CSMA/CA Algorithm	4
3.	Introduction to NetSim	5
4.	Model	5
5.	WSN Simulation in NetSim	6
6.	Calculations	9
7.	Results	10
	Bibliography	
	Appendix	

1. Abstract:

One class of applications envisaged for the IEEE 802.15.4 LR-WPAN (low data rate-wireless personal area network) standard is Wireless Sensor Networks for monitoring and control applications. Wireless Sensor Networks provide a new paradigm for sensing and sending information. Current WSNs typically communicate directly with a centralized controller or a satellite, thus communication between the sensor and controllers is based on a single hop. An on-going area of research is, where WSN nodes or terminals that communicate with each other forming a multi-hop network. Such WSNs could change their topology dynamically when connectivity among the nodes varies with the time due to node mobility. In this paper, we compare NetSim's performance metrics for a multi hop Wireless Sensor Network scenario with a theoretical analysis of the IEEE 802.15.4 standard.

2. The IEEE 802.15.4 for wireless Sensor Networks:

Low rate, low power consumption and low-cost consumption are the key points that lead to the specification of the IEEE 802.15.4 standard. Actually the IEEE 802.15.4 protocol specifies the Medium Access Control (MAC) sub layer and physical layer for the Low-Rate Wireless Private Area Networks. Even though standard was not developed for the sake of Wireless Sensor Networks, it is still suitable for them because Sensor Networks can be built from the Low-Rate Wireless Private Area Networks.

According to IEEE 802.15.4 for a device (sensor) there can be at most three operational modes

- Personal Area Network (PAN) Coordinator: This is the principal controller of the entire network this device identifies its own network, to which other devices may be associated.
- Coordinator: The Coordinator has no capability of creating its own network; A Coordinator does the synchronization services through transmission of beacons. Such a coordinator must be associated to a PAN Coordinator.
- A simple Device: A device (sensor) which is neither a PAN nor Coordinator.

The IEEE 802.15.4 supports two different types of devices.

- Full Functional Device(FFD) : A FFD is a device which supports all the three operational modes
- Reduced Function Device (RFD): The RFD is intended for the applications such as a passive infrared sensor, they do not need to synchronize services, they do not need to identify the network, and they are associated with single FFD at a time.

2.1: The IEEE 802.15.4 Physical Layer

The Physical layer of IEEE 802.15.4 is responsible for the data transmission and reception. This layer is in charge of the following tasks.

- 1) Activation and deactivation of the radio transceiver: The radio transceiver operates in one of the three modes transmitting, receiving or sleeping. Upon the request of the Medium Access Control sub layer it is either turned on or off. The turn around time from transmitting to receiving and vice versa should be no more than 12 symbol periods according to the standard (each symbol corresponds to 4 bits).

- 2) Energy detection(ED): It is an estimation of received signal power with in the bandwidth of an IEEE 802.15.4 channel. The energy detection time should be equal to 8 symbol periods.
- 3) Clear Channel Assessment (CCA): This assessment is responsible for telling whether the medium is busy or idle. The CCA is performed in three operational modes :
 - i) *CCA mode1: Energy above threshold* .CCA shall report a busy medium upon detecting any energy above the ED threshold.
 - ii) *CCA mode2:Carrier sense only* .CCA shall report a busy medium only upon the detection of a signal compliant with this standard with the same modulation and spreading characteristics and which may be higher or lower than the ED threshold.
 - iii) *CCA mode3: Carrier sense with energy above threshold*. CCA shall report a busy medium using a logical combination (logical operator may be AND or OR) of detection of a signal with the modulation and spreading characteristics and energy above the ED threshold.
- 4) Link Quality indication: This measures the Strength or quality of the received packet. It measures the quality of the received signal on the link.
- 5) Channel Frequency Selection: The IEEE 802.15.4 defines 27 different wireless channels. A network can support only part of the channel set. Hence, the physical layer should be able to tune its transceiver into a specific channel request by a higher layer.

2.2: The IEEE 802.15.4 Medium Access Control (MAC)

The MAC sub-layer of the IEEE 802.15.4 protocol provides an interface between the physical layer and the higher layer protocols of LR-WPANs. The most important characteristic of IEEE 802.15.4 MAC Layer is that it has the same common features with the IEEE 802.11 protocol such as the use of CSMA/CA as a channel access protocol. The MAC Layer supports two operational modes that may be selected by the coordinator

- Beacon-enabled mode: beacons are periodically generated by the coordinator to synchronize the attached devices and to identify the PAN. In beacon-enabled mode the devices (sensors) will send their data by using Slotted CSMA/CA.
- Non Beacon-enabled mode: In this mode the devices will simply send their data by using Unslotted CSMA/CA.

The MAC sublayer is responsible for the following tasks

- Generating beacons if the device is a coordinator.
- Synchronizing the beacons.
- Supporting PAN association and disassociation.
- Supporting the device security.

2.3 Introduction to WSN:

A WSN is a collection of sensors which communicate over the wireless channel. These sensor-devices have computational processing ability (i.e. CPU power), wireless receiver and transmitter technology and a power supply. These are used to monitor various physical phenomena e.g. temperature, fluid

levels, vibration, strain, humidity, acidity, pumps, generators to manufacturing lines, aviation, building maintenance and so forth. WSN find application in many areas including structural engineering, agriculture and forestry, healthcare, logistics, transportation, and military applications. Wired sensor networks have long been used to support such environments and, until recently, wireless sensors have been used only when a wired infrastructure is infeasible, such as in remote and hostile locations. In addition, the cost of installing, terminating, testing, maintaining, trouble-shooting, and upgrading a wired network makes wireless systems potentially attractive alternatives for general scenarios.

IEEE 802.15.4 based wireless sensor networks have witnessed explosive growth in the recent past, it is not unreasonable to expect that in 10-15 years the world will be covered with Wireless Sensor Networks with access to them via Internet. This new technology has unlimited potential because of 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.

2.4 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* nodes use Slotted CSMA/CA algorithm for transmitting packets *else* nodes use Unslotted CSMA/CA.

2.5 Unslotted CSMA/CA Algorithm:

The IEEE 802.15.4 defines two versions of CSMA/CA mechanisms

- The slotted CSMA/CA version.
- The Unslotted CSMA/CA version.

A packet transmission begins with a random backoff which is sampled uniformly from 0 to $2^{macminBE} - 1$ followed by a CCA. A CCA failure starts a new backoff process with the backoff exponent raised by one, i.e., to $macminBE+1$, provided it is lesser than the maximum backoff value given by $macmaxBE$. The maximum number of successive CCA failures for the same packet is governed by $macMaxCSMABackoffs$, exceeding which the packet is discarded at the MAC layer. A successful CCA is followed by the radio turnaround time and packet transmission. If the receiver successfully receives the packet i.e., without any collision or corruption due to PHY layer noise, the receiver sends an ACK after waiting for the radio turnaround time. A failed packet reception causes no ACK generation. The transmitter infers that the packet has failed after waiting for $macAckWaitDuration$ and retransmits the packet for a maximum of $aMaxFrameRetries$ times before discarding it at the MAC layer.

Key Packet parameters:

Each packet is characterized by 2 variables NB&BE

Variable	Definition
----------	------------

NB	<i>Number of backoffs</i> the node has undergone while attempting the current transmission, initialized to 0 before every new transmission
BE	<i>Backoff exponent</i> is related to how many backoff periods (0 to $2^{BE} - 1$) a device has to wait before attempting to assess the channel.

Parameter	Value
macminBE	3
macmaxBE	5
macMaxCSMABackoffs	4
aMaxFrameRetries	3

Algorithm Flow:

Unslotted CSMA/CA	
Step1	NB(=0), BE are initialized.
Step2	MAC layer shall delay for a random number of backoff periods in the range (0 to $2^{BE} - 1$)
Step3	MAC will request PHY to perform CCA.
Step4 (Channel is busy)	MAC sub layer shall increment both NB and BE by one, ensuring that BE shall be no more than macmaxBE and if NB is greater than macMaxCSMABackoffs then the packet is discarded else return to step 2.
Step5 (Channel is Idle)	MAC sub layer starts transmission.

3. Introduction to NetSim

NetSim is 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

The network consists of Sensors and a base station. We consider time invariant links and static (immobile) nodes. We assume that Sensors will generate the traffic according to the independent point Poisson processes. These constitute the arrival process for the sensor network. Each sensor transmits the data to the next hop node according to the topology. The intermediate nodes along a route may be relays in which case they simply forward the incoming traffic, or they may be sensors which transmit

their own packets as well as the received packets. Based on the network congestion, the nodes may discard packets due to consecutive failed CCAs.

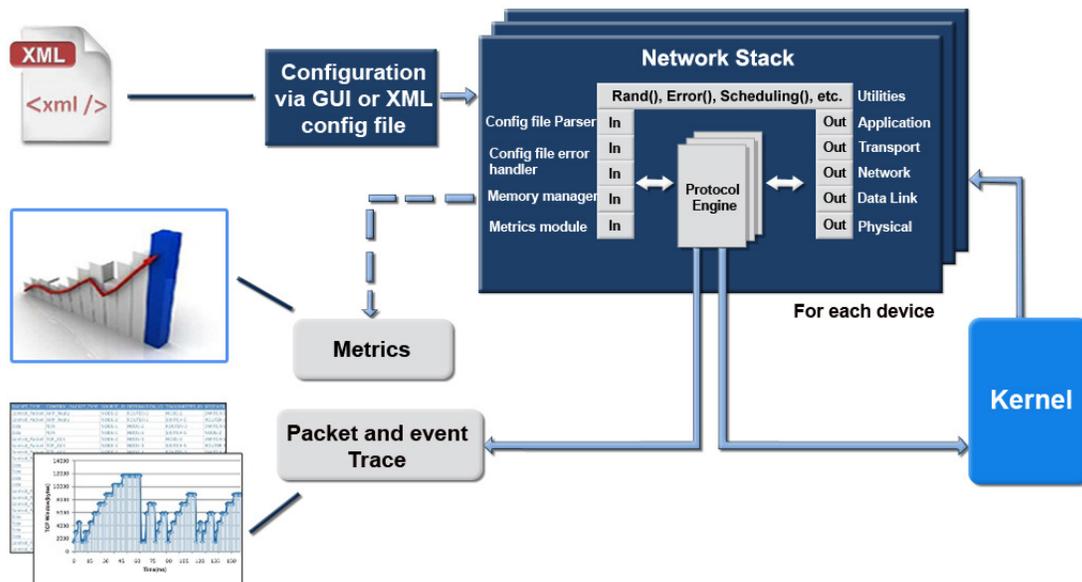
Using [1] as reference we have modified NetSim WSN to meet the following simplifications.

Simplifications
All packets are of same fixed length and hence the same data transmission duration.
All packets involved in the collision get corrupted.
The packet error rate at each link is fixed and known.
Packets flow only from the sensors to the PAN Coordinator.
The acknowledgments are not considered.
A node's CCA succeeds when there is no transmission by any node in its CS range at the time of initiation of its CCA.
Frame Retransmissions are not considered.

5. WSN Simulation in NetSim:

In NetSim, 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 an automated placement model or via drag and drop) and they sense the agents based on the sensor's range and agent's position.

Events in NetSim: NetSim is a discrete event simulator, with a "network-stack" as shown in the diagram.



WSN's central algorithms operate in the MAC layer and Physical layer. Therefore, in NetSim, DataLink-OUT, Physical-OUT, Physical-IN and DataLink-IN events are to be considered and modified. In NetSim's sensor.c file we modify the code to generate *packet arrival rates* at each node by using the standard Poisson distribution.

DataLink-OUT Event: If the arriving packet is a Data packet, and if the Mac buffer is empty the packet is added to the buffer and Unslotted CSMA/CA process is initialized with the default parameters and the frame will go to random backoff. However, if the Mac buffer is not empty the packet is added at the end of Mac buffer. The packets which are received from the predecessor nodes also follow the same procedure as described above. If the arriving packet is a control packet, then depending on the event type different actions are performed:

1. **CarrierSenseStart Event:** The Sensor will perform CCA at the 1st (start) symbol of the CCA duration. If the CCA succeeds then CarrierSenseEnd Event will be pushed into the Event Queue, the failure of CCA will increment the backoff exponent and the number of backoffs, a new backoff process is started provided the incremented number of backoffs is less than the macMaxCSMABackoffs.
2. **CarrierSenseEnd Event:** The sensor will perform the CCA at the 8th (end) symbol of the CCA duration. If the CCA succeeds, CSMADataTransfer Event will be pushed into the Event Queue, the failure of CCA will increment the backoff exponent and the number of backoffs, a new backoff process is started provided the incremented number of backoffs is less than the macMaxCSMABackoffs.
3. **CSMADataTransfer Event:** This event will change the sending node's status to "transmission busy" (TRX_ON_BUSY in NetSim) and pushes the Physical-OUT Event into the Events Queue.

Physical-OUT Event: Based on the packet size the datatransmission end time is calculated. Each and every node present in the carrier sense range of the transmitting node is checked with the status TRX_ON_BUSY , if at that time any other node is found to be transmitting $n_NC_ErrorFlag$ is set to 1. A Physical-IN event is pushed into the Event Queue, after the datatransmission end time and turn around time of the sending node is set to idle .

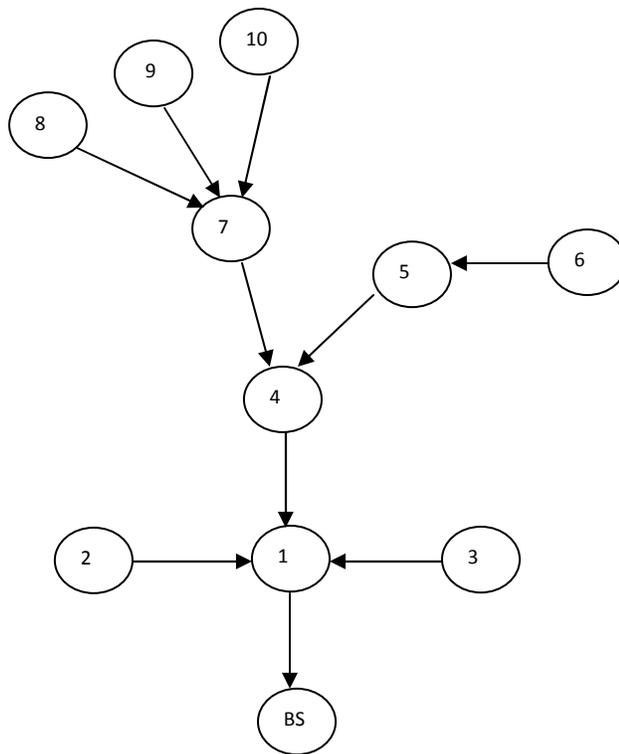
Physical-IN Event: A packet is discarded when there is another transmitting node in the carrier sense range of the receiving node or if $n_NC_ErrorFlag$ was set to 1. If the packet is discarded a control packet is pushed into the Event Queue which will set the receiver status to idle. However, if the packet is not discarded DataLink-IN Event is pushed into the Event Queue.

DataLink IN Event: This event will set the receiver status as idle.

Collision Model: A receiver might conclude error on receiving a packet, if the receiver is receiving a packet from a node and there is another transmission in the carrier sense range of a receiver (even though the packet is not intended for it). Note that the receiver concludes collision of all the packets that are intended for it.

Configuration file & Scenario:

tree-n10-CS9-PERO - implies that the network is a tree with 10 nodes excluding the base station and all of them are source nodes such that on an average 9 nodes are in carrier sense range of a node and the PER on all the links is the same and is equal to 0. In this 10 node tree topology all the nodes are sources and there are no hidden nodes.



Routing graph of tree **tree-n10-CS9-PERO**

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:

Tetcos_NetSim -> Network_Configuration -> Device_Count

```
<DeviceCount="1"Type="SinkNode" />
```

```
<DeviceCount="10"Type="Sensor" />
```

- X and Y Position: (SinkNode and Sensors)
 - Tetcos_NetSim -> Network_Configuration -> Device_Configuration -> Device-> Pos_2D
- Physical layer properties:(SinkNode)
 - Tetcos_NetSim>Network_Configuration>Device_Configuration><DevicId="1"Port_Count="1"Type="SinkNode">-><PortId="1">-><LayerType="Physical_Layer">-> IEEE802_15_4_Phy_Properties
 - Properties specific to PHY layer are set like **CCAMode, FrequencyBand_MHz etc.,**
- MAC layer properties:(SinkNode)
 - Tetcos_NetSim->Network_Configuration->Device_Configuration><DevicId="1"Port_Count="1"Type="SinkNode">-><PortId="1">><LayerType="DataLink_Layer">-> IEEE802_15_4_Mac_Properties
 - Properties specific to MAC layer are set like **AckRequest, Beacon Order etc.,**
- IP Address:(Sensors)
 - Tetcos_NetSim->Network_Configuration->Device_Configuration-><DevicId="1"Port_Count="1"Type="Sensor">-><PortId="1">-><LayerType="Network_Layer">

6. Calculation

Probability of CCA Failure α :

Definition:

The probability of CCA failure is the probability of the occurrence of at least one transmitting node in the CS range of node I , given that node I performs a CCA.

$$\alpha = \frac{\text{Number of CCA'S failed}}{\text{Number of CCA'S attempted}}$$

Packet failure probability γ :

Definition:

A transmitted packet can fail to be decoded by its intended receiver due to a collision, or due to noise. A packet transmitted by a node suffers a collision (and hence fails to be decoded by the receiver) if there is an overlap with a transmission by the other node.

$$\gamma = \frac{\text{Number of frames collided}}{\text{Number of frames transmitted}}$$

Or

$$\gamma = 1 - \frac{(\text{Number of frames successfully received})}{\text{Number of frames transmitted}}$$

Attempt Rate, β :

Definition:

β is the rate of performing CCAs in backofftimes.

$$\beta = \frac{\text{Number of CCA'S attempted}}{\text{Time (in backoff slots) spent in backoff}}$$

The node non-empty probability, q :

Definition: steady state probability of a non empty queue

$$q = \frac{\text{Total Non empty Queue time}}{\text{Simulation time}}$$

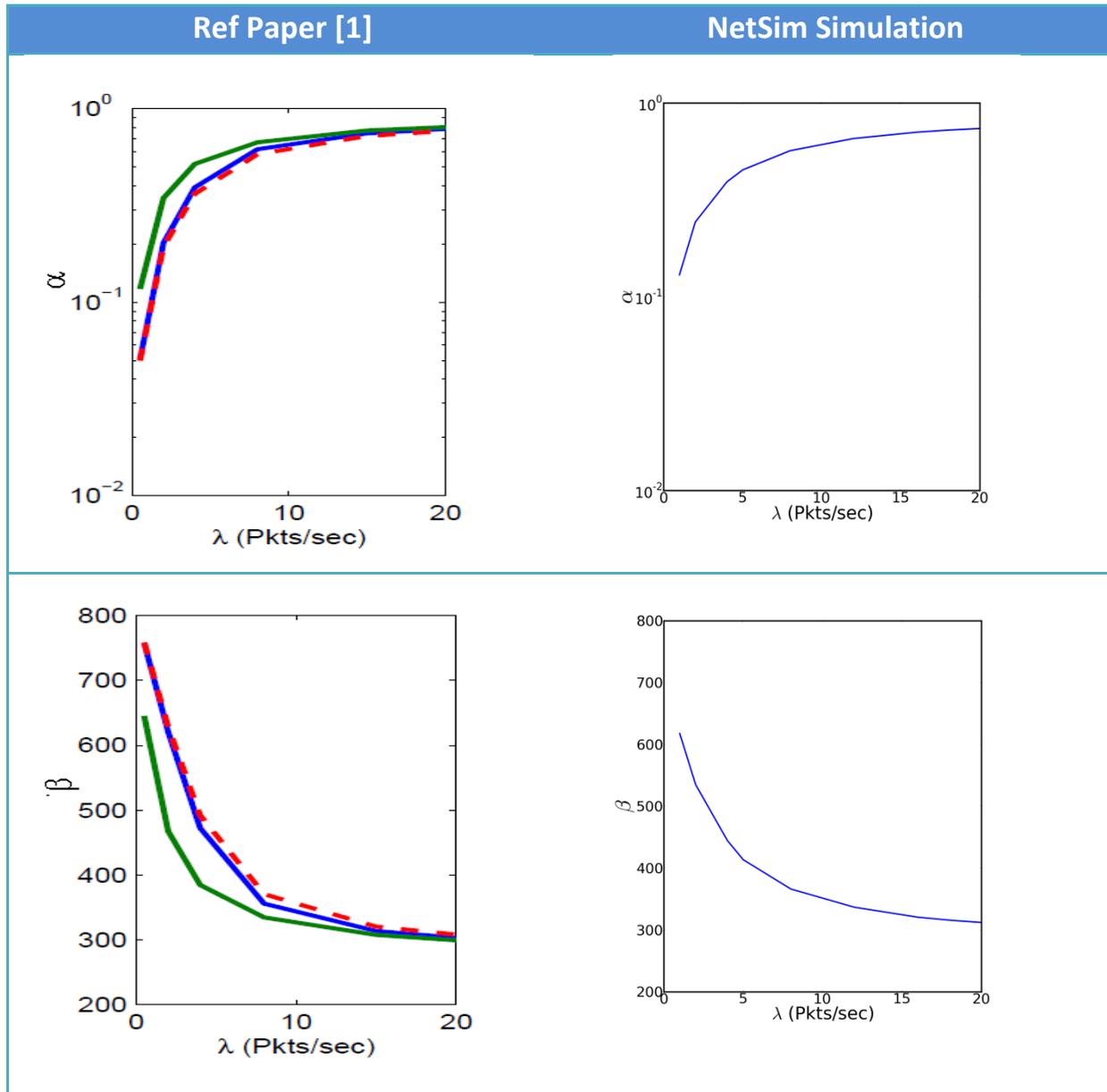
Goodput, θ :

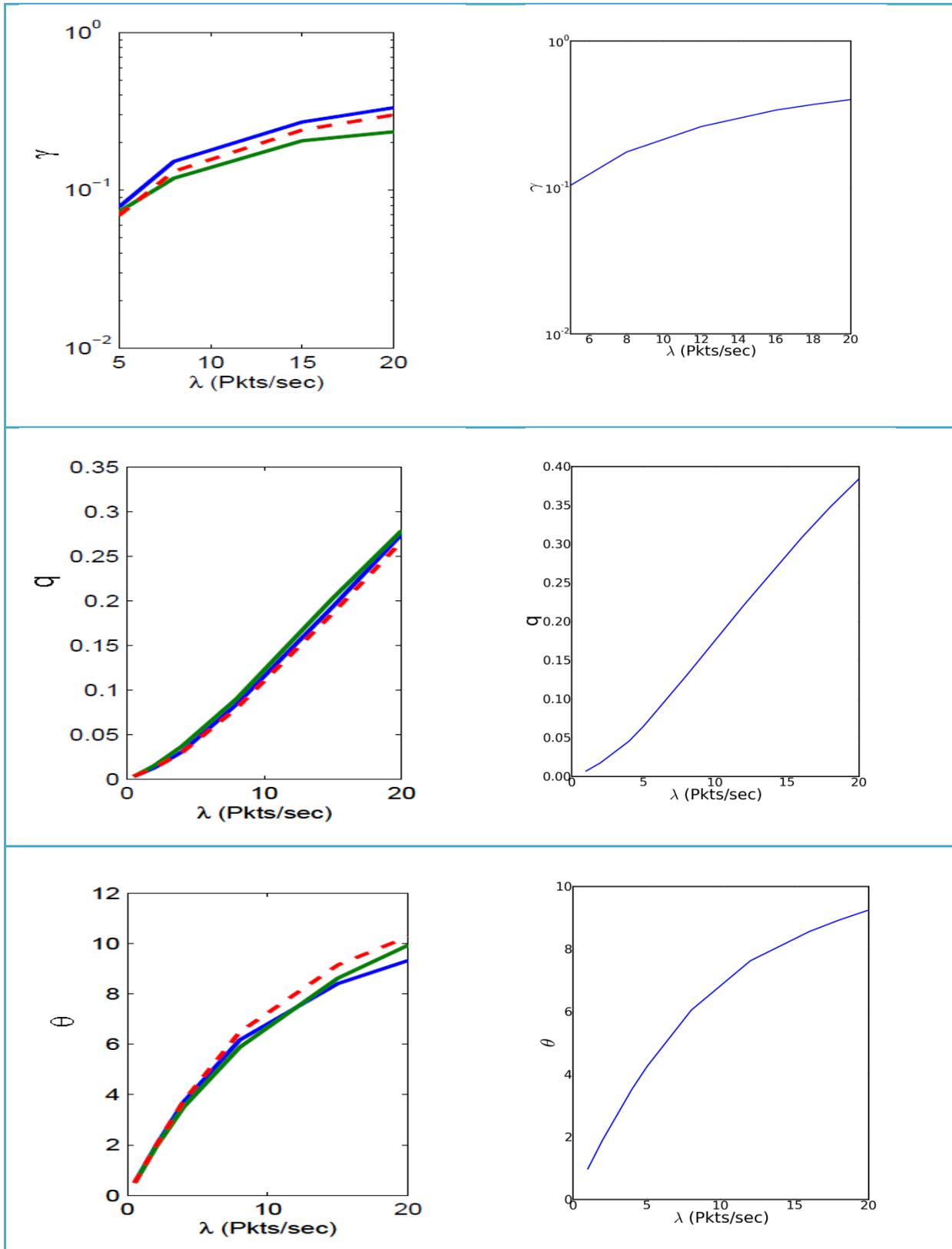
Definition: It is the rate of packet transmission by the node which are successfully received at the next hop receiver.

$$\theta = \frac{\text{Total number of Frames received successfully at the nexthop}}{\text{Simulation time}}$$

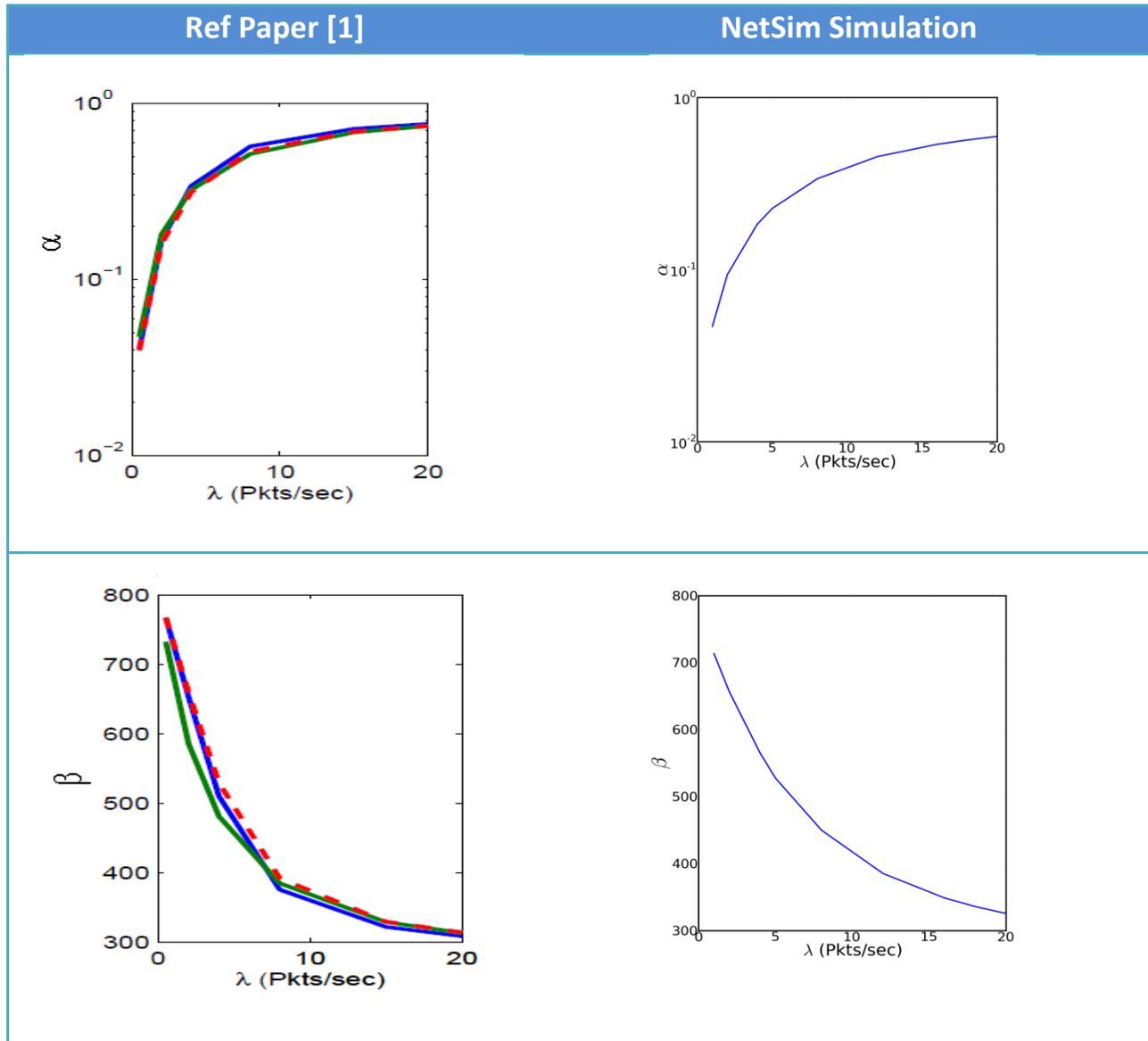
7. Results

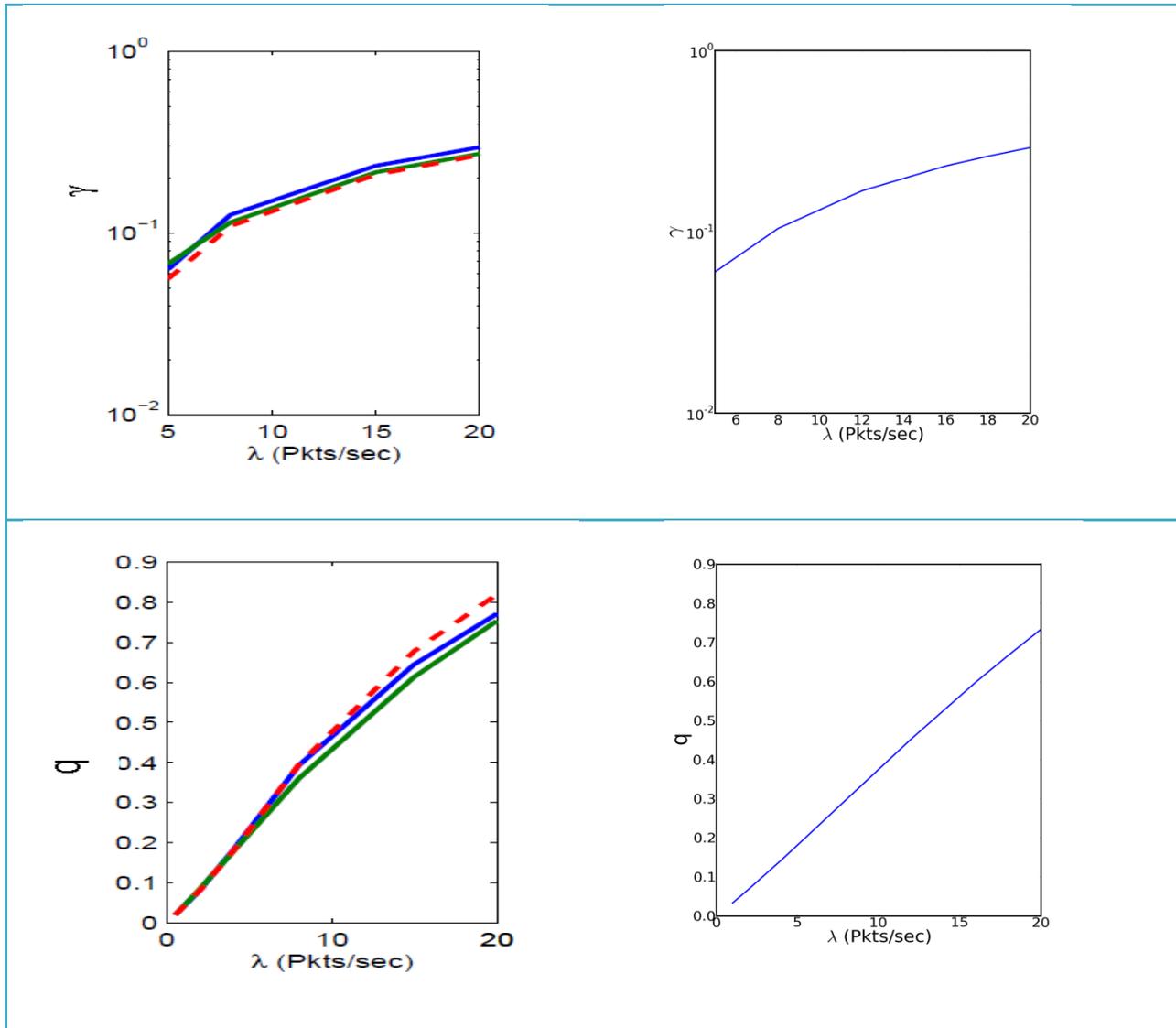
2nd Node Metrics





4th Node Metrics





Appendix 1: Python Plots

Using python's matplotlib module we plot various graphs using the source code given below

```
import os
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d
from scipy.interpolate import spline
### NOTE: Change these variables as needed
MetricsPath='./Output/';
Path=MetricsPath+'2_node_AllMetrics/'; # location of files with Metrics
Path1=MetricsPath+'2_node_FrmTx/';
Header='GenRate\tmilli{T\tBackoff\tNE-Que}\tCCA{T\tSucc\tFail}\tFrm{T\tDiscrd\tSucc}\tCollCnt\n';
SaveImagesPath=MetricsPath+'2_node_AllMetrics/'; # location where plots should be saved to
SaveMetricsFile = MetricsPath+'AllMetrics.txt'; # Name of file for saving aggregate metrics
plotShow=0; # If set as 1, shows all plots
TimeConversionFactor=1000;# conversion factor needed for making units of time in text files ass seconds

Y_limits = [];

## Customised plot function
def myplot(X,Y,ShowRnot,Ylabel,Xlabel,Title,PlotName):
    global Y_limits
    if(PlotName == SaveImagesPath+'beta.png'):
        plt.figure(num=None, figsize=(5.42,6.56), dpi=96, facecolor='w', edgecolor='k')
        plt.plot(X,Y)
        plt.ylim(200,800)
    if(PlotName == SaveImagesPath+'throughput.png'):
        plt.plot(X,Y)
    if(PlotName == SaveImagesPath+'q.png'):
        plt.plot(X,Y)
        plt.ylim((0,0.4))
    if(PlotName == SaveImagesPath+'alpha.png'):
        plt.ylim((0.01,1))
        plt.semilogy(X,Y)
        plt.ylim(0.01,1)
    if(PlotName == SaveImagesPath+'gamma.png'):
        plt.semilogy(X,Y)
        plt.xlim((5,20))
    if len(Y_limits) != 0:
        plt.ylim(Y_limits)
        Y_limits=[]
    plt.ylabel(Ylabel);
    plt.xlabel(Xlabel);
    plt.title(Title);
    plt.savefig(PlotName);
    if plotShow == 1:
        plt.show();
    plt.clf();
AllMetrics=[]; # Stores all metrics in one big matrix
GenRates=[]; # Stores list of Generation Rates
for filename in os.listdir(Path):
    GenRates.append(int(filename.rsplit('.')[1]));
```

```

AllMetrics.append([]);
GenRates=list(np.sort(GenRates));
for filename in os.listdir(Path):
    print 'Reading '+filename;
    Data=np.loadtxt(Path+filename,dtype='float',delimiter='\t',skiprows=1);
    Data=Data[:,1:(Data.size/len(Data))];
    FrmTxData=np.loadtxt(Path+filename.replace('AllMetrics','FrmTx'),dtype='float',delimiter='\t',skiprows=1);
    Data=np.column_stack([Data,FrmTxData[:,2],FrmTxData[:,1]]);
    AllMetrics[GenRates.index(int(filename.rsplit('_')[1]))]=np.average(Data,axis=0);

## Copying header from txt files
fp=open(SaveMetricsFile,'w');
fp.write(Header);
fp.close();

## Writing AllMetrics into a text file
fp=open(SaveMetricsFile,'a');
AllMetrics = np.array(AllMetrics);
TempAllMetrics = np.column_stack([GenRates,AllMetrics]);
#TempAllMetrics = np.vstack([Header,TempAllMetrics]);
np.savetxt(fp,np.round_(TempAllMetrics),fmt='%d',delimiter='\t');
fp.close();
##### Plots #####
## beta :: CCA Attempt Rate, given that node is in backoff
## beta = (CCA Attempted)/(Time spend in Backoff) i.e., Column-3/Column-1
#Y_limits = [700,900];
myplot(GenRates,(TimeConversionFactor)*AllMetrics[:,3]/AllMetrics[:,1],plotShow,r'$\beta$',r'$\lambda$ (Pkts/sec)',r'$\beta$(CCA attempt rate in backoff) vs $\lambda$',SaveImagesPath+'beta.png');

## alpha :: CCA Failure Probability
## alpha = (CCA Failed)/(CCA Attempted) i.e., Column-5/Column-3
#Y_limits=[0,1];
myplot(GenRates,AllMetrics[:,5]/AllMetrics[:,3],plotShow,r'$\alpha$',r'$\lambda$ (Pkts/sec)',r'$\alpha$(CCA Failure Prob) vs $\lambda$',SaveImagesPath+'alpha.png');

## q :: Steady State Prob of NE Queue
## q = (Non-Empty Queue Time)/(Simulation Time) i.e., Column-2/Column-0
#Y_limits=[0,1];
myplot(GenRates,AllMetrics[:,2]/AllMetrics[:,0],plotShow,'q',r'$\lambda$ (Pkts/sec)',r'q(Steady State Prob of NE Queue) vs $\lambda$',SaveImagesPath+'q.png');

## theta :: Goodput
## theta = (FramesTransmittedSuccessfully)/(SimulationTime) i.e., Column-8/Column-0
#Y_limits=[0,max(GenRates)];
myplot(GenRates,(TimeConversionFactor)*AllMetrics[:,8]/AllMetrics[:,0],plotShow,r'$\theta$',r'$\lambda$ (Pkts/sec)',r'$\theta$(Goodput) vs $\lambda$',SaveImagesPath+'throughput.png');

## gamma :: Packet failure Probability
## gamma = 1 - ((Frames Succ Rcvd)/(Frames Transmitted)) i.e., 1- (Column-8/Column-6)
#Y_limits=[0,1];
myplot(GenRates,1-(AllMetrics[:,8]/AllMetrics[:,6]),plotShow,r'$\gamma$',r'$\lambda$ (Pkts/sec)',r'$\gamma$(Packet Failure Prob) vs $\lambda$',SaveImagesPath+'gamma.png');
#myplot(GenRates,(AllMetrics[:,9]/AllMetrics[:,6]),plotShow,r'$\gamma$',r'$\lambda$ (Pkts/sec)',r'$\gamma$(Packet Failure Prob) vs $\lambda$',SaveImagesPath+'gamma.png');

```

Bibliography

1. *An Analytical Performance Model for Beaconless IEEE 802.15.4 Multi-Hop Networks* by **Sanjay Motilal Ladwa**, 2012, M.Tech Thesis, Indian Institute of Science, IISc, Bangalore.
2. *Performance evaluation of an IEEE 802.15.4 sensor network with a star topology.* **Singh, Chandramani K., Kumar, Anurag and Ameer, P.M.** 2008, *Wireless Networks* 14, pp. 543–568.
3. *IEEE 802.15.4 for Wireless Sensor Networks: A Technical Overview* by **Anis Koubaa, Mario Alves, Eduardo Tovar.**