Development of a Congestion Adaptive Routing Algorithm for MANET Throughput Optimization

Rebecca Nyasuguta Arika[1], A. Mindila[2], W. Cheruiyot[3]
Jomo Kenyatta University of Agriculture and Technology – Nairobi, Kenya.

Abstract – Congestion is one of the most crucial restrictions of wireless ad-hoc network. This is because congestion can greatly deteriorate the performance of whole mobile ad hoc network. The current design of MANET routing protocols does not handle congestion in an efficient manner. This is because they handle congestion in a reactive manner. This means that the current routing protocols allow congestion to occur, which will then be detected by congestion control. However, dealing with congestion in reactive manner results in longer delay and an unnecessary packet loss and requires significant overhead if the new route recalculation is required. The motivation for the development of this congestion adaptive routing algorithm for throughput optimization is guided by the realization that congestion is the principal cause for packet loss, long delay, and high overhead in mobile ad hoc networks. To achieve the paper objective, prototyping was employed where a model mobile ad hoc network was designed and simulated. The model fine tuning involved node weight usage in path cost computations. The results obtained indicated that the developed fared well in congestion control since the number of dead links remained at the zero level throughout the observation period, implying that all nodes remained usable throughout the transmission duration. Moreover, the sum of energy required for data transmissions reduced significantly after the path establishment.

Index Terms – MANET, Throughput, Delay, Congestion Control.

1. INTRODUCTION

Congestion in mobile ad hoc networks has the ability to hamper network transmissions such that we have delays and packet losses. This may eventually lead to the wastage of time and energy on congestion recovery (Reddy et al., 2015). An analysis of the current mobile ad hoc network (MANETs) designs reveals that the routing they implement is not congestion adaptive.

This means that these routing protocols permit congestion to happen, which is afterwards detected by congestion control. However, as Kaur and Singhai (2015) illustrate, dealing with congestion in this reactive manner has the effect of introducing longer network delays and unnecessary packet loss. Moreover, the recovery from congestion requires significant overhead for new route recalculations.

According to Amin et al., (2014), this dilemma becomes more predominant in intensive transmission of heavy traffic such as multimedia data. In these transmissions, congestion is more probable and the negative impact of packet loss on the service quality is more consequential.

In an ideal routing protocol, routing should not only be aware of, but also be adaptive to, network congestion. Therefore, as Malhotra and Kaur (2016) noted, routing protocols which are adaptive to the congestion status of a mobile ad hoc network can greatly improve the network performance. Many protocols which are congestion aware and congestion adaptive have been proposed by various researchers as discussed in the following section.

2. RELATED WORK

Researchers have proposed quite a number of adaptive congestion routing protocols. These congestion routing protocols, according to Shrivastava, et al., (2011) include Congestion Adaptive Routing Protocol (CRP), Efficient Congestion Adaptive Routing Protocol ECARP), Congestion Aware Routing Protocol (CARP), Congestion Aware Distance Vector (CADV) , Congestion Aware Routing plus rate Adaptation (CARA), and Congestion Aware Routing Protocol for Mobile Ad-hoc Network (CARM).

CRP is a congestion adaptive unicast routing protocol for MANETs. This protocol tries to evade congestion from occurring in the first place. In this approach, every node appearing on a route issues a warning to its previous node when prone to be congested (Subburamm and Khader (, 2012).

Effectively, this protocol employs the additional paths called bypass for circumventing the potential congestion area to the first non congested node on the primary route. In so doing, it reduces packet delay. Moreover, it tries to reduce bypass to minimize protocol overhead.

Therefore, Kumaran, and Sankaranarayanan (2014) discuss that the traffic is split over bypass and primary and adaptively to network congestion. In so doing, power consumption is efficient, congestion is resolved beforehand and at the same time there is small packet loss rate.

The drawback of this protocol is that it requires regular congestion monitoring, primary route discovery and bypass discovery, which majorly use broadcasts that can further lead to access rate deteriorations.

In ECARP, the protocol ensures high availability of substitute routes and reduces the rate of stale routes. This protocol works best when Adaptive on Demand Distance Vector (AODV) is used as the underlying routing protocol. (Rajeswari and Venkataramani, 2013) point out that this is normally realized by increasing the parameters of routing protocols that normally take more time for link recovery.

These parameters are *active_route_time-out, route_reply_wait_time, reverse_route_life, TTL_start, TTL_increment, TTL_threshold* and *delete_period.* The main challenge with this protocol is that it necessitates the frequent checking of the occupancy of link layer buffer of node. Moreover, it requires that the Number of packet buffered in bufferbe computed for each of the checked occupancy.

In their study, Chan and JieLee (2014) explain that the base use of CARA protocol is Dynamic Source Routing (DSR). In this approach, the route discovery mechanism of the dynamic source routing is varied. This protocol seeks to find the bypass route for congested zones or nodes. To accomplish this, it combines the average media access control (MAC) utilization and the instantaneous transmission queue length to indicate the congestion level of nodes in the network.

This requires that when the data source wants to communicate with the destination node, it broadcasts RREQ packets. When intermediary node receives RREQ, it checks its congestion level. If the congestion level is higher, it rejects the RREQ. When RREQ arrives at the destination node, though destination node is congested or not it handles the RREQ and replies RREP.

In this way, a route without congested node is established between the source and the destination. The setback of this protocol is that if the node has many packets waiting in the queue, it causes long packet latency or even dropping of packets.

According to Aamir and Zaidi (2013), CARM is a congestion aware routing protocol for mobile ad hoc networks that utilizes a metric incorporating data rates, MAC overhead and buffer delay to control the congestion. It introduces a new parameter called Weighted Channel Delay (WCD) to measure congestion level.

Thereafter, it adopts a route Effective link Data-rate Category (ELDC) to avoid the Mismatched data-rate route (MDRR) problem. It main challenge is that it necessitates that the source node broadcast RREQ packets with ELDC and WCD information when it attempts to transmit data to the destination. These broadcasts can lead to further network congestions.

In the case of CADV, (Kaur and Singhai, 2015) discuss that this protocol is based on proactive protocol, destination sequenced distance vector (DSDV). In this approach, every host maintains a routing table that contains a distances from itself to possible destinations. A mobile host in ad –hoc network acts like a single server queuing system. Delay in sending packet is related with congestion. Its main setbacks are that its overhead is unacceptable when the network is large.

In their study, Singh et al., (2015) explain that CARP is an on-demand routing protocol that utilizes information gathered from MAC layer to discover congestion free routes. It employs combined weight matrix in its standard cost function to check for the congestion level. The multiple paths are computed during the route discovery.

Its demerit is that it necessitates for frequent re-calculation of node weight matrix, which then assigns a cost to each link in the network and select maximum throughput paths. This can call for additional intensive utilization of network resources such as memory and CPU cycles.

3. STUDY APPROACH

The research simulation consisted of several mobile nodes and one base station. The idea was to transit a packet from any one of the nodes to the base station. To achieve this, a number of nodes will form an ad hoc network for the forwarding and final delivery of the packets. Figure 1 shows the design of the simulation set up that was employed for this research.

This figure shows that there is only one base station (indicated in green) while there are several mobile nodes or stations. The X-axis represents the horizontal coordinates of the mobile nodes while the Y- axis represents the vertical coordinates of the mobile nodes.

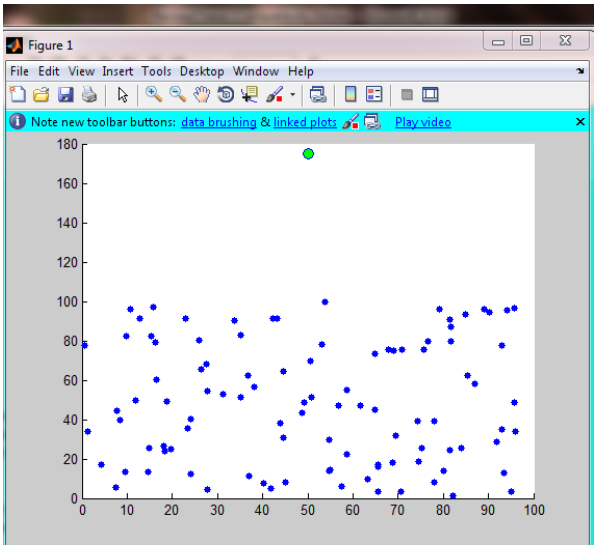

Figure 1: Simulation Design

All the calculations in the following sub-sections employed a modified version of this design setup as the basis for their computations.

A. Mobile Node Clustering

The mobile node wishing to transfer some data to the base station does so through intermediaries, which are the mobile nodes located between itself and the base station. The chosen path dictates the intermediary mobile nodes that are to be employed for packet delivery. Figure 2 shows the design for this clustering.

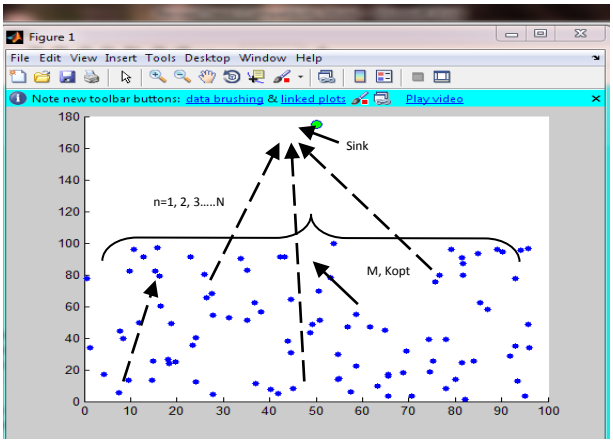

Figure 2: Mobile Node Clustering Design

Figure 3 gives a snippet of the code that was used to achieve node clustering. The parameters that are used for this clustering include *N* (which represents the number of nodes in the network), *M* (representing how the mobile node should move within the coverage area), *Kopt* (representing the Optimum path to be taken.

The value of *Kopt* is taken to be a function of the number of MANET nodes, network architecture, required energy, available space, number of available paths to base station and path cost. There is some operator overloading with the second *Kopt*, which essentially rounds off the calculated path to the nearest decimal value in X, Y format.

Another equally important variable included in the optimum path calculation is the *dBs*, which represents the total distance of the node under consideration, from the base station. The simulation design shown in Figure 1 was used as the basis for the development of the algorithm below. The distance from the base controller and the optimum path to be taken, according to Rimer and Hancke (2016) is given by:

dBS = sqrt(netArch.Sink.x ^ 2 + netArch.Sink.y ^ 2)……..(i)

where *dBS* is the distance from the base station, *sqrt* is the square root, *netArch* is the MANET network architecture, and *sink* is the technical term for the base station, *x* and *y* refers to the *x* and *y* Cartesian coordinates.

*Kopt = sqrt(N) / sqrt(2*pi)……..(ii)*

Where *Kopt* is the Optimum path to be taken, *N* is the number of nodes in the network and *pi* is taken to be 3.14.

```
## CLUSTER OPTIMIZATION##
function kOpt = clusterOptimum(netArch, nodeArch, dBS)% function to optimally cluster network nodes
% calculate the optimum values for number of nodes
%  Input:
%      netArch    network model... % MANET Network architecture
%      nodeArch   nodes model ...% MANET Network nodes architecture
%      dBS        length from base station ...% Represents the 'path cost' where the base station is the
MANET controller
%  Example:
%      dBS = sqrt(netArch.Sink.x ^ 2 + netArch.Sink.y ^ 2);% Node location  along the X and Y cartesian
coordinates, SINK is the base station coordinates
%      numClusters    = clusterOptimum(netArch, nodeArch, dBS);...% 'node weight' representing the
number of MANET  nodes clusters
%
   N   = nodeArch.numNode; %.... number of MANET nodes
   M   = sqrt(netArch.Yard.Length * netArch.Yard.Width); %..Description how the MANET nodes should
move in the coverage area
   kOpt = sqrt(N) / sqrt(2*pi) * ...%..Optimum path to be taken [function of:- number of MANET
nodes,network architecture,required energy, available space,number of available paths to BS,& path cost]
          sqrt(netArch.Energy.freeSpace / netArch.Energy.multiPath) * ...
          M / dBS ^ 2;
   kOpt = round(kOpt);% ...Round off the calculated path to the nearest decimal value in X, Y format
end % Terminate the function
```

Figure 3: Mobile Node Clustering Snippet

A. Figure Sketching

The mobile nodes within the MANET coverage area are free to move both in the X and Y axes. Therefore, the X axis was represent time while the Y axis was used to represent either sum of energies, number of dead nodes or the number of packets delivered as shown in Figure 4.

This research provided mechanisms for the physical representation of the captured data in a MANET environment. This was in the form of figures whose part of the source code is shown in Figure 5. These figures include sum of energy versus time, number of dead nodes versus time and number of packets delivered per unit time. These parameters are crucial since they contribute to the overall MANET network congestion. As an illustration, when a given path has very large sum of energies, the packets delivered through that path take long to reach the destination, and in some cases, may be dropped. The number of dead nodes indicate the number of unusable links, which if high, may lead to massive MANET congestion. On the other hand, the number of packets delivered represents throughput, which if high, serves as an indication of low MANET congestion.

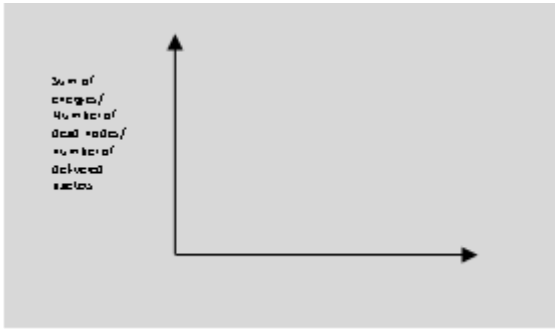

Figure 4: Figure Sketching Design

The snippet shows that the plots were made in a Cartesian coordinate system, using the Y –axis and the X-axis. Among

other variables specified in the source code is the vector of x data, vector of y data, time, font weight of the labels, font name of the figure labels, font size of the labels and font color.

Method calling programming feature was then employed to consolidate all these information to come up with clear figures for the three entities: sum of energies, number of dead nodes and number of packets delivered.

```
### CREATE THREE FIGURE FOR [NUMBER OF PACKETS SENT TO BS], [NUMBER OF DEAD
NODES] & [SUM OF UTILIZED ENERGY FOR MOTION] ###
function createfigure(X1, Y1, Y2, Y3)
%CREATEFIGURE(X1,Y1,Y2,Y3)
%  X1:  vector of x data
%  Y1:  vector of y data
%  Y2:  vector of y data
%  Y3:  vector of y data

% Create figure
figure1 = figure(2);
% Create sub-plot
subplot1 = subplot(1,3,3,'Parent',figure1);
box(subplot1,'on');
hold(subplot1,'all');

% Create plot
plot(X1,Y1,'Parent',subplot1,'LineWidth',2,'Color',[0 1 0]);

% Create x-label
xlabel('Time','FontWeight','bold','FontSize',11,'FontName','Cambria');% ....Calibrating the X-axis

% Create y-label
ylabel('Sum Of Energy','FontWeight','bold','FontSize',11,...% ....Calibrating the y-axis
    'FontName','Cambria');
```

Figure 5: Figure Sketching Snippet

A. Energy Calculations for Selected Path

As the packet travels from the source to the destination, energy is required to help it traverse the mobile nodes to the base station as shown in Figure 6. The snippet of the source code shown in Figure 7 was employed for energy calculations.

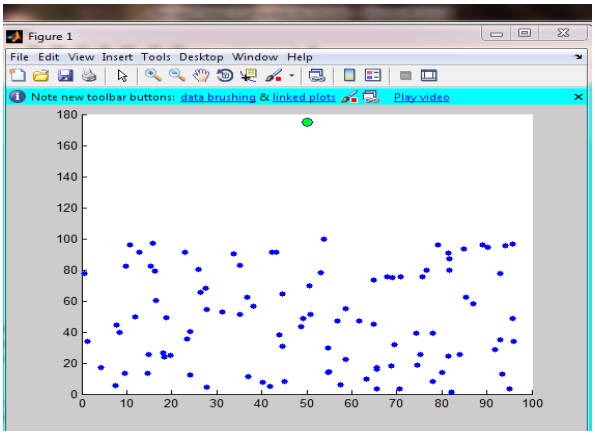

Figure 6: Energy Calculations for Selected Path Design

The variables that are included in the calculation of the sum of energy include number of available paths *(n)*, energy emitted (*ETX*), energy received from base station (*ERX*), aggregate energy for assembling the packets (*EDA*), energy for all paths (*Emp*), total energy for the whole MANET architecture (*Efs*), and the size of the packet to be transmitted (*packetLength*). The sum energy is then calculated as the total of energy for data aggregation and the energy for transferring packets from the source to the base station. That is:

Sum of energy = data aggregation energy + packet transfer energy……..(iii)

Implying that:

Sum of energy = ETX+EDA)……(iv)

A product of the packet length, energy for all paths, and distance give the effective sum of energy for a given path as shown by equation (v):

*Energy for selected path = ((ETX+EDA) * packetLength + Emp * packetLength * (distance ^ 4))……(v)*

Figure 7 illustrates the fact that there is some looping through the source code in that for each packet transferred in a MANET environment, the sum of energy for each of the available path through which packets may be sent has to be computed.

```
### CALCULATION OF THE SUM OF ENERGIES FOR SELECTED PATHS ####

function clusterModel = dissEnergyCH(clusterModel, roundArch)

    nodeArch = clusterModel.nodeArch;
    netArch  = clusterModel.netArch;
    cluster  = clusterModel.clusterNode;

    d0 = sqrt(netArch.Energy.freeSpace / ...
            netArch.Energy.multiPath);
    if cluster.countCHs == 0 % Initial conditions
        return
    end
    n = length(cluster.no); % Number of available paths
    ETX = netArch.Energy.transfer;  % Energy emitted
    ERX = netArch.Energy.receive;   % Energy received from BS
    EDA = netArch.Energy.aggr;      % Aggregate Energy for assembling the packets
    Emp = netArch.Energy.multiPath; % Energy for all paths
    Efs = netArch.Energy.freeSpace; % total Energy for the whole MANET architecture
    packetLength = roundArch.packetLength; % Determining the size of the packet to
be transmitted
    ctrPacketLength = roundArch.ctrPacketLength;
    for i = 1:n
        chNo = cluster.no(i);
        distance = cluster.distance(i);
        energy = nodeArch.node(chNo).energy;
        % Energy for data aggregation  + Energy for transferring the packets to BS
        if(distance >= d0)
            nodeArch.node(chNo).energy = energy - ...
                ((ETX+EDA) * packetLength + Emp * packetLength * (distance ^ 4));
        else
            nodeArch.node(chNo).energy = energy - ...
                ((ETX+EDA) * packetLength + Efs * packetLength * (distance ^ 2));
        end
        nodeArch.node(chNo).energy = nodeArch.node(chNo).energy - ...
            ctrPacketLength * ERX * round(nodeArch.numNode /
clusterModel.numCluster);
    end

    clusterModel.nodeArch = nodeArch;
end
```

Figure 7: Energy Calculations for the Selected Path Snippet

A. Packet Delivered Plots

One of the plots that were crucial for the evaluation of the developed algorithm was that of the number of packets delivered across the MANET environment. As Figure 8 demonstrates, the parameters that were employed as input variables included the clustering algorithm that was called via programming method calling technique, number of possible

clusters, MANET model, and the mobile nodes model (the way the mobile nodes are distributed within the coverage area). As Figure 8 shows, the calculation of the number of packets delivered involves the determination of the cluster model, node clustering model, number of clusters, all of which are factored in to compute this value. The resulting plot is that of the number of packets delivered to the base station against time. The section that reads:

if r == 1

par.packetToBS(r) = clusterModel.numCluster; } = ……(vi)

……(vi)

else

par.packetToBS(r) = par.packetToBS(r-1) + clusterModel.clusterNode.countCHs;

implies that when the available number of paths from the node to the base station is 1, then the total number of packets delivered to the base station is equivalent to the number of available clusters. However, for the case of multiple paths, the total number of packets delivered is equal to the sum of the packets delivered in each available channel.

```
### PLOT OF DATA AGAINST Time  ###
function fig(data, r, n, yLabel, Title)
% plot data against round, uses method call
   figure(2);
   subplot(1, 3, n);
   plot(1:r, data);
   xlabel('Time');
   ylabel(yLabel);
   title(Title);
end

### CREATION OF NEW CLUSTERS BASED ON SELECTED OPTIMUM PARAMETERS ###
|
|
function clusterModel = newCluster(netArch, nodeArch, ...
                clusterFun, clusterFunParam, p_numCluster)
% Create the network architecture with required parameters
%
%   Input:
%       clusterFun          Function name for clustering algorithm.
%       clusterFunParam     Parameters for the cluster function
%       numCluster          Number of clusters (CHs)
%       netArch             Network model
%       nodeArch            Nodes model
%   Example:
%       clusterModel = newCluster();
%
   |
   |
function par = plotResults(clusterModel, r, par)
   nodeArch = clusterModel.nodeArch;
   netArch = clusterModel.netArch;


   %%%%% number of packets sent from clusters to BS
   if r == 1
      par.packetToBS(r) = clusterModel.numCluster;
   else
      par.packetToBS(r) = par.packetToBS(r-1) + clusterModel.clusterNode.countCHs;
   end
   % Figure packet to BS
%     fig(par.packetToBS, r, 1, '# of packets sent to BS nodes', ...
%         'Number of packet sent to BS vs. time');
```

Figure 8: Packet Delivered Plot Snippet

A. Number of Dead Nodes Plots

In this research, a node was considered dead if it is no longer part of the cluster that actively acts as intermediary for packet delivery to the base station. Figure 9 illustrates a snippet for plotting number of dead nodes with time.

```
   |
   |
%%%%% Number of dead nodes
    par.numDead(r) = nodeArch.numDead;
    % Figure number of dead node
%     fig(par.numDead, r, 2, 'number  of dead nodes', 'Number of dead node vs.
time);
   |
   |
```

Figure 9: Number of Dead Nodes Snippet

This figure demonstrates the fact that number of dead nodes is a function of the network architecture. This means that depending on how the MANET is structured in terms of node distribution (mobile node model), some of the network mobile nodes will be actively involved in packet forwarding while the rest will be just redundant. The redundant mobile nodes are labeled ‘dead nodes’ for this research.

4. RESULTS AND DISCUSSION

The proposed algorithm that would optimize network throughput based on node weight to calculate the path cost was developed in Matlab integrated development. Unlike OLSR whose throughput is based on bits transferred per second, this algorithm utilized the concept of number of actual packets delivered to the destination from the source as the basis for throughput determination.

Moreover, the developed algorithm provided a module for indicating the number of unused nodes that result when the network topology changes, and whose routes should be avoided in order to reduce the number of packets dropped across the network. It was also important to have a module for indicating the sum of energy utilized to transmit packets to the destination.

A. Packet Delivery Ratio

In this algorithm, instead of using bits per second as a basis for determining the network throughput, the actual number of packets delivered across the mobile ad hoc network was utilized as the basis as shown in Figure 10.

The packet delivery ratio demonstrates the number of packets that were transferred from the source node, through the mobile node intermediary to the base station.

As this Figure demonstrates that the number of packets sent over the network was observed for a total of 20 seconds. A graph was then plotted to show the variations of the packet transmission rate against time. The graph indicates that number of packets sent increases exponentially to a maximum value, above which the transmission rate falls a bit and then starts rising exponentially again. Figure 11 shows the results of interpolation of Figure 10 that was presented earlier.

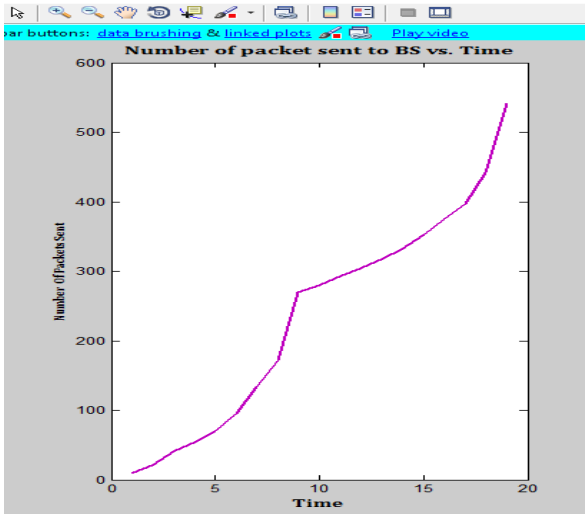

Figure 10: Packet Delivery Ratio

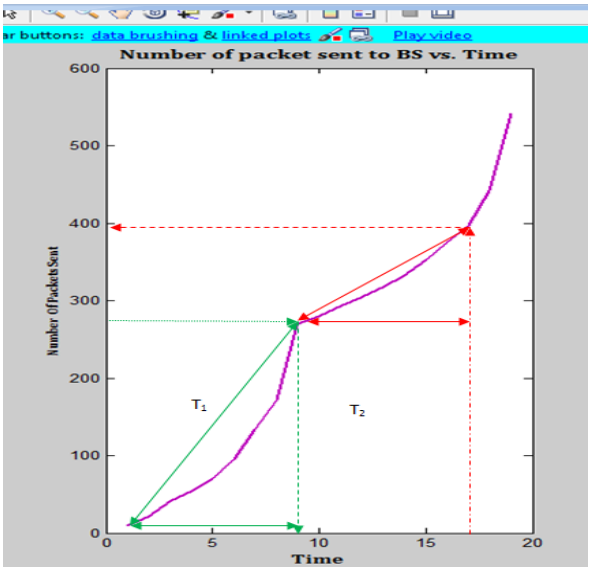

Figure 11: Packet Delivery Ratio Interpolation

Considering the X axis at instant of 8th second and the corresponding Y axis value, then it is clear that the number of packets delivered were 280. After this instant, the graph experiences a slow growth up to the 17th second. After this instant, once again, the graph experiences a sharp growth.

The gradient of the graph under T_1 gives the number of packets delivered per unit time. Considering the graph under T_1, and taking number of packets delivered to be *NPD*, and gradient to be *Grad*, we have:

$\Delta Y = [280 - 8]$ ……(vii)

$\Delta X = 8 - 2$

Number of packets delivered per unit time under T_1:

$= \Delta Y / \Delta X]$ ……(viii)

$=272/6$

$=45$

$NPD\ Grad\ (T_1) = 45$

Similarly under T_2, the gradient of the graph gives the number of packets delivered per unit time. Considering the graph under T_2, we have:

$\Delta Y = [398 - 280]$

$\Delta X = 17 - 8$

Number of packets delivered per unit time under T_2:

$= \Delta Y / \Delta X$ ……(ix)

$=118/9$

$=13$

$NPD\ Grad\ (T_2) = 13$

The consequences of these gradients are that more packets are transferred under T1 compared to T2, that is:

$NPD\ Grad\ (T_1) > NPD\ Grad\ (T_2)$.

In both T_1 and T_2, the point at which the fall occurs represents the congestion point. However, this congestion point lasts for a few number of seconds after which the transmission rates resumes it exponential growth.

Therefore, the developed algorithm indeed avoids congestion by adapting to the existing network conditions and only momentarily reducing the number of packets transmitted so as to guard against packet loss.

This is in contrast to OLSR where network nodes move continuously in an effort to determine the best location and hence path for transmitting packets. Therefore, the developed algorithm use less energy in transmissions compared to OLSR.

B. Number of Dead Links

In MANETs, dead nodes represent dead links and whose paths packets should never be sent. A node was regarded dead if it moves in such a way that it becomes isolated from the rest of the MANET nodes. A plot of number of dead nodes against time is shown in Figure 12.

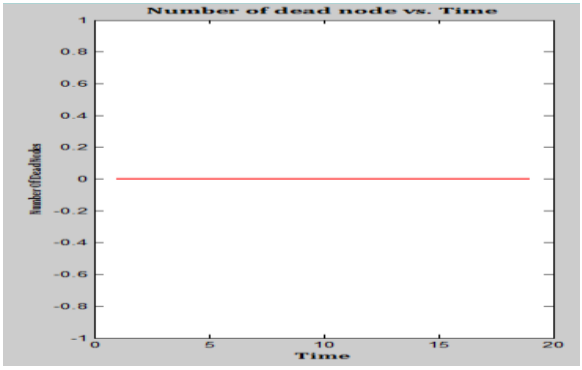

Figure 12: Number of Dead Links

Figure 9 indicates that the number of dead nodes remained constant at zero level for all number of packets sent and number of transmissions. This can be demonstrated by computing the gradient of the graph, which will give number of dead nodes per unit. To achieve this, Figure 12 was modified as follows to yield Figure 13.

Considering area under T_1, and taking number of dead nodes to be *NDN*, and gradient to be *Grad* then:

$$\Delta Y = [0 - 0]$$

$$\Delta X = 10 - 2$$

Number of dead nodes per unit time under T_1:

$$= \Delta Y / \Delta X \quad \ldots\ldots(x)$$

$$=0/8$$

$$=0$$

$$NDN \ Grad\ (T_1) = 0$$

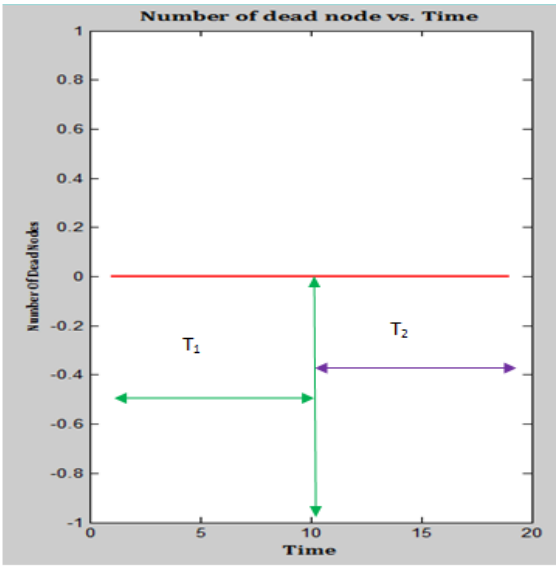

Figure 13: Number of Dead Links Gradient

Considering area under T_2, then:

$$\Delta Y = [0 - 0]$$

$$\Delta X = 18 - 10$$

Number of dead nodes per unit time under T_2:

$$= \Delta Y / \Delta X \quad \ldots\ldots(xi)$$

$$=0/8$$

$$=0$$

$$NDN \ Grad\ (T_2) = 0$$

The consequences of these gradients are that the number of dead nodes per unit time was similar for the two instances T_1 and T_2 under consideration. That is:

$$NDN \ Grad\ (T_1) = NDN \ Grad\ (T_2)$$

The implication of this is that all network nodes are active throughout the transmission time. Therefore, the developed algorithm has ideal node weight based path cost calculation as all the calculated paths are usable and the network packets can traverse through any of them to reach the destination.

C. Sum of Energy

An ideal routing algorithm is the one that ensures least energy possible for the transmission of packets across the network. This is important because most of the mobile ad hoc networks devices are powered by small batteries which are rechargeable.

In such scenarios, the low the energy usage the longer the duration the mobile device will be during the transmission process. Figure 14 shows a graph of the sum of energy against the transmission time.

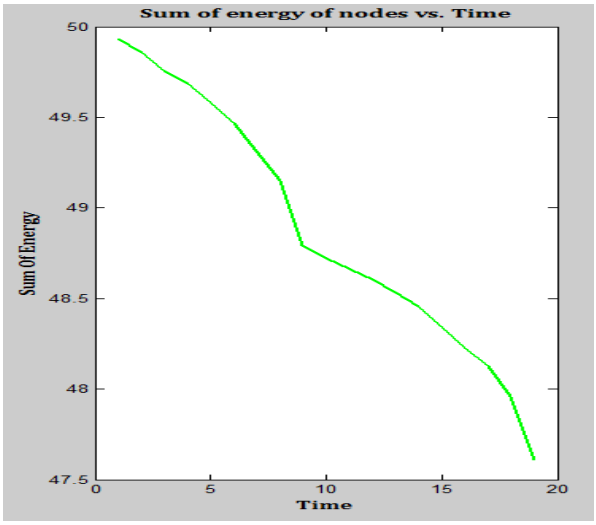

Figure 14: Sum Of Energy

Figure 14 shows that the sum of energy employed for packet transmission experiences slight fluctuations as the transmission time increases. The variations can be attributed to change of transmission times which are accompanied by change of path that packets employ to travel across the network.

For instance, a packet may take a slightly congested path for one round and a less congested path during another round.

A slightly congested path will require more energy than a path that has no congestion at all. This figure demonstrates that the highest amount of energy is 50, implying that the worst path requires a sum of 50 units of energy.

Since the sum of energy exhibit slight fluctuations, the developed algorithm has great performance in path cost calculations. The gradient of this graph can give a trend on the sum of energy per unit time. This can be achieved if Figure 14 is modified to yield Figure 15 as follows.

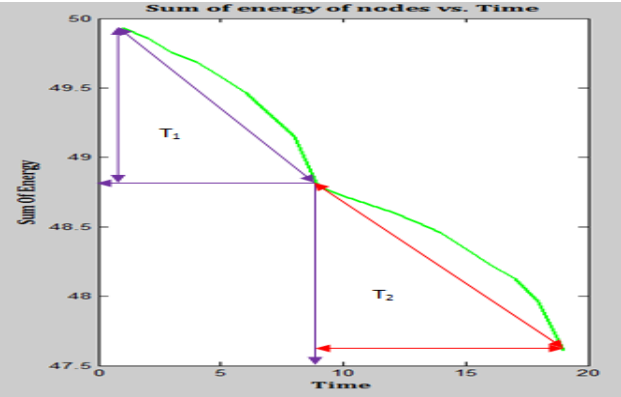

Figure 15: Sum Of Energy Gradient

Considering area under T_1, and taking sum of energy to be *SOE*, and gradient to be *Grad* then:

$$\Delta Y = [49.99 - 48.78]$$

$$\Delta X = 9 - 2$$

Sum of energy per unit time under T_1:

$$= \Delta Y / \Delta X$$

$$=1.21/7$$

$$=0.17$$

$$SOE\ \ Grad\ (T_1) = 0.17$$

Considering area under T_2, then:

$$\Delta Y = [48.78 - 47.62]$$

$$\Delta X = 19 - 9$$

Sum of energy per unit time under T_2:

$$= \Delta Y / \Delta X$$

$$=1.16/10$$

$$=0.116$$

$$SOE\ \ Grad\ (T_2) = 0.116$$

The interpretation of this is that the gradient under T_1 is more than the gradient under T_2. That is:

$$SOE\ \ Grad\ (T_1) > SOE\ \ Grad\ (T_2)$$

Therefore, the sum of energy per unit for the initial packet transmissions is more than the sum of energy at later phases of packet transmissions. The consequence of this is that during the initial start up transmissions, the MANET nodes have not leant the network conditions and they consume quite some energy to transmit packets through non-optimum links. However, after the ideal paths have been established, the transmissions of subsequent packets take less energy.

5. CONCLUSIONS AND RECOMMENDATIONS

This paper has discussed the existing congestion control algorithms in a mobile ad hoc network and their limitations. It has been noted that it is imperative to have adaptive solution for congestion in the network if packet losses, number of dead links, sum of energy for packet transmissions, and network delays were to be minimized. The existing routing protocols proposed for MANETs have their strengths and certain limitations. For instance, router assisted congestion control approaches and end to end rate based flow control mechanisms create complexity at routers. Moreover, hop-by-hop congestion control approaches bring on scalability problems.

Therefore, a well-balanced congestion control system to be employed for the stability and optimized performance of the wireless network was developed to address the problems inherent in current congestion control protocols. This developed protocol was demonstrated to perform well in terms of reducing the number of dead links in a MANET. Additionally, the sum of energy required for data transmissions were observed to reduce significantly upon route establishment.

Moreover, the number of packets delivered across the network was shown to reduce slightly after congestion detection, after which the transmission rates started rising steadily.

REFERENCES

[1] G. Reddy, V. Podili & K. Sekharaiah (2015). A Survey on Issues and Challenges in Congestion Adaptive Routing in Mobile Ad Hoc Network. Global Journal of Computer Science and Technology: E Network, Web & Security.

[2] N. Kaur and R. Singhai (2015). Review on Congestion Control Methods for Network Optimization in MANET. International Journal of Computer Applications (0975 – 8887) Volume 121 – No.7.

[3] M. Amin, K. Fard, S. Karamizadeh, M. Aflaki (2014). Enhancing Congestion Control To Address Link Failure Loss over Mobile Ad-Hoc Network. International Journal of Computer Networks & Communications (IJCNC) Vol.3, No.5, pp.177-192.

[4] G. Malhotra and K. Kaur (2016). Optimization Of Throughput In MANATs Against Black Hole Attack Using Genetic Algorithm. IJEEE, Volume 3, Issue 3.

[5] L. Shrivastava, S. Tomar, and S. Bhadauria. (2011). A Survey on Congestion Adaptive Routing Protocols for Mobile Ad-hoc Networks. Int. Journal of Computer Theory and Engineering, vol. 3, Issue 2, pp. 189-196.

[6] S. Subburamm and P. Khader (2012). Efficient Broadcasting using Preventive Congestion Mechanism in Mobile ad Hoc Network. European Journal of Scientific Research, Vol.83, No.2, pp.302-313.

[7] T. Kumaran, and V. Sankaranarayanan (2014). Early congestion detection and adaptive routing in MANET. Egyptian Information Journal, Elsevier, Vol.12, pp 165-175.

[8] S. Rajeswari, and Y. Venkataramani (2013). Congestion Control and QOS Improvement for AEERG protocol in MANET. International Journal on AdHoc Networking Systems (IJANS) Vol. 2, No. 1, pp.13-21.

[9] B. Soelistijanto P. Howarth (2013). Transfer Reliability and Congestion Control in Opportunistic Networks: A Survey. IEEE Communications Surveys and Tutorials.

[10] Y. Chan and H. JieLee (2014). A Hybrid Congestion Control for TCP over High Speed Networks. Sixth International Conference on Genetic and Evolutionary Computing.

[11] M. Aamir and A. Zaidi (2013). A Buffer Management Scheme for Packet Queues in MANET. Tsinghua Science And Technology ISSNll1007-0214ll01/10llpp543-553 Volume 18.

[12] A. Singh, M. Xiang, Z. Yasir (2015). Enhancing Fairness and Congestion Control in Multipath TCP.Main technical program at IFIP WMNC.

[13] Rimer and P. Hancke (2016). Calculation of an Optimum Mobile Sink Path in a MANETs.