Volume: 05, Issue: 04, Pages:2002-2010 (2014) ISSN : 0975-0290

# Delay-Efficient Energy and Velocity Conscious Non-Preemptive Scheduler (DEV-NS) For Mobile Ad Hoc Networks

**Dr. Anuradha Banerjee**
Assistant Professor of Computer Applications, Kalyani Govt. Engg. College, Kalyani. Nadia
Email: anuradha79bn@gmail.com

**Dr. Paramartha Dutta**
Professor, Dept. of Computer and System Science,Visva-Bharati University, Santiniketan
Email: Paramartha.dutta@gmail.com

**Abstract**

A mobile ad hoc network is an infrastructure-less network where the nodes are free to move independently in any direction. The nodes have limited power and their set of neighbours, both uplink and downlink, change with time due to the inherent dynamism of the nodes. When multiple message forwarding requests arrive at one particular node, it needs to arrange them within its queue according to a suitable scheduling algorithm that is energy efficient as well as concerned about the relative velocity of the associated nodes. The present article proposes one such scheduling algorithm DEV-NS (Delay-efficient Energy and Velocity Conscious Non-preemptive Scheduler) that instructs each node to assign weights to its uplink neighbours at regular intervals. Message forwarding request from the uplink neighbour with highest weight is served first. Simulation results clearly establish the fact that among all the state-of-the-art scheduling algorithms, DEV-NS produces much better result in terms of end-to-end delay, message cost, network throughput and connectivity ratio.

Keywords: Ad hoc network, delay-efficiency, energy, non-preemption, scheduling, velocity, weight.

Date of Submission: February 10, 2014 Date of Acceptance: March 06, 2014

## 1. INTRODUCTION

An ad hoc network is a collection of wireless nodes which form a temporary network without relying on an existing infrastructure or centralized administration. These networks are deployed mainly in emergency situations like battlefield, natural disasters like earthquake, floods etc. [1-5]. Many routing protocols have been proposed in ad hoc networks so far. In all of them, when the destination node is out of the radio-range of the source node the communication has to be multi-hop where some nodes act as router to bridge the gap between the source and destination nodes [6-10]. If a node receives multiple message forwarding requests, it serves one of them and stores the others in its message queue. The order in which these requests will be served is termed as a schedule. The job of a scheduler is to pick up that particular schedule that is expected to produce the best performance. The choice of scheduling algorithm has a significant effect on the overall performance of the route, especially when the traffic load is high [10-14].

There are different scheduling policies for different network scenarios. Different routing protocols use different methods of scheduling. Among them, FCFS (first-come-first-served) is quite heavily used. The drop-tail policy is used as a queue management algorithm in various scheduling algorithms for buffer management [13]. Except for the no-priority scheduling algorithm, all other scheduling algorithms give higher priority to control packets than to data packets. In no priority scheduling algorithm both control and data packets are served in FIFO (first-in-first-out) order.

Certain scheduling schemes depend on the size of the message and number of hops to traverse. In smallest message first (SMF) [11] algorithm, the packets are scheduled in ascending order of the size of messages of which they are a part. Packets belonging to smaller messages receive higher priority over the packets belonging to larger messages. In order to implement this scheme the total message size must be attached to each packet so that the scheduler can access this information while putting the packets in queue. In smallest remaining message first scheme ( SRMF) [12, 13] packets are ordered on the basis of the amount of message packets remaining to be sent after the current packet. On the other hand, in shortest hop length first (SHLF) scheduling [11, 14] the distance between the source and destination, measured in terms of the number of hops, influences the time a packet needs to reach its destination. The packet with the shortest hop is assigned highest priority. The scheduling decision is made independently at each node.

Some other priority schedulers [11, 14] concentrate on packet delivery ratio. If a node $n_j$ produces higher packet delivery ratio for a node $n_i$ compared to another node $n_k$, then a packet from $n_j$ will be assigned higher priority in message queue of $n_i$ compared to a packet from $n_k$. Age of a packet in also considered as an important parameter to avert starvation in some priority scheduling schemes. Energy efficient schedulers also consider [11] residual energy of source and destination nodes.

The present article focuses on creating a delay-efficient schedule consisting of data packet forwarding requests. Control packets are stored in a separate queue and served in a FIFO manner. The purpose of a delay-efficient scheduler is to produce a schedule that is expected to suffer from minimum delay. DEV-NS is a unique delay-efficient scheduler in the respect that it is concerned about the cost of route-rediscovery due to link breakage. Link breakage may take place due to complete exhaustion of battery power of a node or when a node goes out of the radio-range of one of its uplink neighbour that was its predecessor in the broken route. Also it estimates and takes into consideration the additional number of packets that could be forwarded within the waiting period of the node in the message queues of its downlink neighbours. All these not only produce a delay-efficient schedule but also decrease the message cost and improves network throughput. DEV-NS is based on the fact that if $n_i$ sends some message to $n_j$ which $n_j$ is supposed to forward to $n_k$ (i.e. $n_j$ is a downlink neighbour of $n_i$ and $n_k$ is a downlink neighbour of $n_j$) in a communication route then $n_i$ can send the next packet to $n_j$ only after $n_k$ acknowledges to $n_i$ the receipt of the earlier packet.

## 2. THE SCHEME OF DEV-NS

The scheme of DEV-NS places one particular message forwarding request $\Re_j$ of a node $n_j$ in the message queue of $n_i$ ($n_j \in UP_i(t)$ where t is the current time and $UP_i(t)$ is the set of uplink neighbours of $n_i$ at time t) based on the vulnerability index (to be referred to as v-index later on) and delay index (to be referred to as d-index later on) of one particular schedule. Consider a schedule S in the message queue of $n_i$ at time t is, S: $\Re_{j1}\ \Re_{j2}\ \Re_{j3} \dots \Re_{jp}$ . Here $\Re_{j1}$ and $\Re_{jp}$ are placed at the rear and front end of the queue. Here $n_{j1}, n_{j2}, n_{j3} \dots n_{jp}$ all are uplink neighbours of $n_i$ that has sent message forwarding requests to $n_i$ at time t. Assume that $\tau(\Re_{j1}), \tau(\Re_{j2}), \dots \tau(\Re_{jp})$ are the time duration required by $n_i$ for serving message forwarding requests $\Re_{j1}, \Re_{j1}, \dots \Re_{jp}$ respectively. If $\Re_j$ is placed before $\Re_{j1}$ i.e. in the rear end of the queue then another schedule S′ is formed s.t. S′: S: $\Re_j\ \Re_{j1}\ \Re_{j2}\ \Re_{j3} \dots \Re_{jp}$ . So the waiting time $T_j$ of $\Re_j$ in S′ is defined by,

$$T_j = \tau(\Re_{j1}) + \tau(\Re_{j2}) + \dots \tau(\Re_{jp})$$

$$\text{i.e. } T_j = \sum_{a=1}^{p} \tau(\Re_{ja}) \qquad (1)$$

**Definition 1: Vulnerable Node**

The node $n_i$ will be termed as a vulnerable node in schedule S′ provided the following conditions (i and iii) or (ii and iii) are true.

i) The expected residual energy of $n_j$ at time $(t+T_j)$ i.e. the timestamp when $\Re_j$ will be served by $n_i$, is less than or equal to 40% of it's total battery power $E_j$

ii) The relative velocity of $n_j$ w.r.t. $n_i$ averaged over a time interval from (t-TH) to t (where t is the current time) should be high. TH is less than or equal to the minimum timestamp of initiation of all the communication sessions alive at $n_j$ at time t, sharing the link from $n_j$ to $n_i$.

iii) The expected number of alive communication sessions through $n_j$ at time $(t+T_j)$ is greater than or equal to 1.

**Determining the expected residual energy of $n_j$ w.r.t. it's total battery power at time t**

Assume that at time t the consumed energy of $n_j$ is $e_j(t)$ and $n_j$ started to operate in the network at timestamp $tmp_j$. So, the energy depletion rate of $n_j$ is $\{e_j(t)/(t-tmp_j)\}$. Hence the expected residual energy of $n_j$ w.r.t. its total battery power $E_j$ at time $(t+T_j)$ is denoted by $\alpha_j(t+T_j)$ and defined in (2).

$$\alpha_j(t+T_j)=1-(e_j(t)\,(t+T_j-tmp_j))/(E_j\,(t-tmp_j)) \text{ ........(2)}$$

If $\alpha_j(t+T_j)$ is less than 0.4 then it serves the condition i) of $n_j$ to be a vulnerable node.

**Determining the relative velocity sensitivity of $n_j$ w.r.t. one of it's downlink neighbour $n_i$ averaged over the time interval (t-TH) to t**

From the definition of TH, $0<TH \leq min\_init_j(t)$ where $min\_init_j(t)$ is the minimum of the initiation timestamp of all those communication sessions alive at $n_j$ at time t.
Let the velocity of a node $n_j$ at time t is given by $v_j(t)$. So, the relative velocity of $n_j$ w.r.t. $n_i$ at time t is given by $(v_j(t) - v_i(t))$. The same relative velocity averaged over the time interval from (t-TH) to t is denoted by $V_{i,j}(t-TH,t)$ and defined in (3).

$$V_{i,j}(t-TH,t) = 1-(1/(TH+1))\sum_{k=0}^{TH}(1/(v_j(t-k)-v_i(t-k))) \text{ .........(3)}$$

If $V_{i,j}(t-TH,t) \geq 0.75$, then it serves the condition ii) of $n_j$ to be a vulnerable node.

**Determining the expected number of alive communication sessions through $n_j$ at time $(t+T_j)$**

Assume that the set of alive communication sessions through $n_j$ at time t is denoted by $AC_j(t)$ and the residual number of packets to be transferred for one such alive communication session CS is $rem\_pkt_{CS}(t)$ and the time of initiation of CS in $n_j$ is $init_j(CS)$. So, $init_j(CS)$ is the timestamp when the first data packet belonging to the communication session CS arrived at $n_j$. From the definition of $min\_init_j(t)$ mentioned earlier in this section, it is mathematically expressed in (4).

$$min\_init_j(t)=\min_{CS\in AC_j(t)}(\forall\, init_j(CS)) \text{ ............ (4)}$$

Also assume that between the time interval $min\_init_j(t)$ and t, the number of packets that has been transferred through $n_j$ corresponding to the communication session CS is $tot\_pkt_{j,CS}(t)$. Let the rate of packet forwarding of $n_j$ corresponding to the communication session CS within the interval between t and $(t+T_j)$ is given by $rt\_pkt_{j,CS}(t)$ and defined in (5).

$$rt\_pkt_{j,CS}(t)=tot\_pkt_{j,CS}(t)/(t-min\_init_j(t)) \text{ ..........(5)}$$

Then the number of packets expected to be forwarded by $n_j$ for the communication session CS within the time interval t and $(t+T_j)$ is $(rt\_pkt_{j,CS}(t)\ T_j)$. If $(rt\_pkt_{j,CS}(t)\ T_j) < rem\_pkt_{CS}(t)$ for at least one $CS \in AC_j(t)$, then it satisfies the condition iii) for $n_j$ being a vulnerable node.

**Definition 2: υ-index or vulnerability index of a node**

The υ-index or vulnerability index of $n_j$ at time t is the number of communication sessions expected to be alive in $n_j$ at time $(t+T_j)$. It is expressed as υ-$index_j(t)$.

**Definition 3: d-index or delay index of a node**

In order to illustrate the d-index or delay index of a node, consider node $n_{j1}$ in schedule S in the message queue of $n_i$ where $n_i$ is a downlink neighbour of $n_{j1}$ at time t, s.t. S: $\Re_{j1}\ \Re_{j2}\ \Re_{j3} \ldots \Re_{jp}$ . The waiting period of $n_{j1}$ in the message queue of $n_i$ is $T_{j1}$ and it is mathematically expressed as,

$$T_{j1} = \tau(\Re_{j2}) + \tau(\Re_{j3}) + \ldots \tau(\Re_{jp})$$

$$\text{i.e. } T_{j1} = \sum_{a=2}^{p}\tau(\Re_{ja}) \text{ ........................................(6)}$$

Throughout the time interval $T_{j1}$ the link from $n_{j1}$ to $n_i$ remains idle. If the idle time could be saved i.e. if $\Re_{j1}$ could be served by $n_i$ immediately after it arrived at the message queue of $n_i$, then $n_{j1}$ could transmit some more packets through the link from $n_{j1}$ to $n_i$. The maximum number of such packets is $(rt\_pkt_{j1,CS}(t)\ T_{j1})$ if $(rt\_pkt_{j1,CS}(t)\ T_{j1}) < rem\_pkt_{CS}(t)$; otherwise it is $rem\_pkt_{CS}(t)$. So the maximum time that could be saved in the ideal situation (when none of those packets face any delay in the message queue of $n_i$) is the time d-$index_{j1}(t)$ required to transfer those packets to the successor $n_{f(CS,i)}$ of $n_i$ in the communication session CS as per the geographical position of the involved nodes at time t. Let $T'_{i,CS}(t)$ denote the time required to transfer one such packet from $n_i$ to $n_{f(CS,i)}$ as per the geographical positions of $n_i$ and $n_{f(CS,i)}$ at time t. It is mathematically expressed in (7).

$$T'_{i,CS}(t) = dist_{i,\, f(CS,i1)}(t) / vl_s \quad .........................(7)$$

$dist_{a,b}(t)$ denotes the Euclidean distance between $n_a$ and $n_b$ at time t. $vl_s$ is the speed of the wireless signal.

So, $dist_{a,b}(t) = \sqrt{\{(x_a(t)-x_b(t))^2+(y_a(t)-y_b(t))^2\}}$

The geographical position of a node $n_a$ at time t is given by $(x_a(t), y_a(t))$.
$dist_{i,\, f(CS,i1)}(t) \leq R_i$ where $R_i$ is the radio-range of node $n_i$.

$$d\text{-}index_{j1}(t) = \begin{cases} (rt\_pkt_{j1,CS}(t)\ T_{j1}\ T'_{i,CS}(t)) & \text{if } (rt\_pkt_{j1,CS}(t)\ T_{j1}) < rem\_pkt_{CS}(t) \\ (rem\_pkt_{CS}(t)\ T'_{i,CS}(t)) & \text{otherwise} \end{cases} \quad (8)$$

**Definition 4: υ-index of a schedule**

υ-index of a schedule S: $\mathfrak{R}_{j1}\ \mathfrak{R}_{j2}\ \mathfrak{R}_{j3} \ldots \mathfrak{R}_{jp}$ at time t is denoted as $\upsilon\text{-}i_S(t)$ and defined in (9).

$$\upsilon\text{-}i_S(t) = \upsilon\text{-}index_{j1}(t)+ \upsilon\text{-}index_{j2}(t)+\ldots+ \upsilon\text{-}index_{jp}(t) \quad ...............(9)$$

$$\text{i.e. } \upsilon\text{-}i_S(t) = \sum_{a=1}^{p} \upsilon\text{-}index_{ja}(t)$$

**Definition 5: d-index of a schedule**

d-index of a schedule S: $\mathfrak{R}_{j1}\ \mathfrak{R}_{j2}\ \mathfrak{R}_{j3} \ldots \mathfrak{R}_{jp}$ at time t is denoted as $d\text{-}i_S(t)$ and defined in (10).

$$d\text{-}i_S(t) = d\text{-}index_{j1}(t)+ d\text{-}index_{j2}(t)+\ldots+ d\text{-}index_{jp}(t) \quad ..............................(10)$$

$$\text{i.e. } d\text{-}i_S(t) = \sum_{a=1}^{p} d\text{-}index_{ja}(t)$$

**Detecting the better among two schedules**

Consider two schedules S1 and S2 among which the better one is to be chosen at time t. Nine different cases can be there and the comparative analysis is based on following two propositions.
Proposition 1: The cost of messages for each route discovery in ad hoc networks, is $\{(\mu^{H+1}-1)/(\mu-1)-1\}$ where $\mu$ is the average number of downlink neighbours of a node.

Proof: The source of the communication session to be established broadcasts route-request (RREQ) packets to $\mu$ number of downlink neighbours. Each of those downlink neighbours broadcast the same RREQ packet to $\mu$ number of downlink neighbours again. The process continues till H number of hops where H is the maximum allowable ho count in the network. Hence the RREQ is forwarded to a total of $(\mu+\mu^2+\mu^3+\ldots+\mu^H)$ i.e. $\{(1+\mu+\mu^2+\mu^3+\ldots+\mu^H)-1\}$. Therefore the cost of messages due to route discovery is $\{(\mu^{H+1}-1)/(\mu-1)-1\}$.

Proposition 2: The minimum and maximum time required for route discovery in ad hoc networks are $H(R_{min}+R_{max})/2vl_s$ and $\{(\mu^H-1)/(\mu-1)\}\{(R_{min}+R_{max})/2vl_s$.

Proof: The way RREQ packets are forwarded, it gives rise to a tree structure as shown in figure 1. The source node of the communication session is root of the tree.

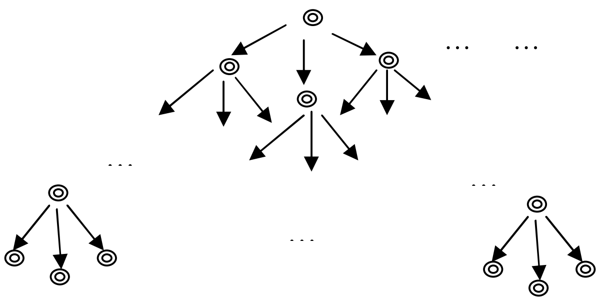

Figure 1: Tree structure of route discovery in ad hoc networks

Route discovery can be completed in minimum time provided the nodes in same level of the tree forward route-request at the same time. Number of levels in the tree is H. For communication between a non-leaf node in the tree and one of its downlink neighbour, the required time is $(R_{min}+R_{max})/2vl_s$, where $R_{min}$ and $R_{max}$ are minimum and maximum radio-ranges of the network. The significance of $vl_s$ has already been described. Therefore, the minimum time required for route discovery is $H(R_{min}+R_{max})/2vl_s$.

As far as the maximum time for route discovery is concerned, it corresponds to the situation where all nodes in all the levels broadcast route-request at different timestamp. So, all nodes which are downlink neighbour of same node, receive the RREQ at the same time but those who are downlink neighbours of different nodes, do not receive RREQ at the same time. So, the time required for the first level is 1 unit, for second level it is $\mu$ units, for the third level it is $\mu^2$ units and so on. Therefore, the maximum time required for route discovery is $\{(1+\mu+\mu^2+\mu^3+\ldots+\mu^{H-1})\{(R_{min}+R_{max})/2vl_s\}\}$ i.e. $\{(\mu^H-1)/(\mu-1)\}\{(R_{min}+R_{max})/2vl_s\}$.

On an average, the time required for route discovery $\{(\mu^H-1)/(\mu-1)+H\}\{(R_{min}+R_{max})/4vl_s\}$.

**Comparison between two schedules**

Nine possible cases are described below one after the other. The current time is t.

Case-1: $(\upsilon\text{-}i_{S1}(t) > \upsilon\text{-}i_{S2}(t))$ and $(d\text{-}i_{S1}(t) > d\text{-}i_{S2}(t))$

Let $\delta1=(\upsilon\text{-}i_{S1}(t)-\upsilon\text{-}i_{S2}(t))$ and $\gamma1= (d\text{-}i_{S1}(t) - d\text{-}i_{S2}(t))$

This case corresponds to the situation where schedule S1 produces more vulnerability and causes more delay to the network compared to schedule S2. So, without any doubt, schedule S2 is better among the two. Delay efficiency of S2 over S1 is $[\delta1\{(\mu^H-1)/(\mu-1)+H\}\{(R_{min}+R_{max})/4vl_s\}+\gamma1]$.

Case-2: $(\upsilon\text{-}i_{S1}(t) < \upsilon\text{-}i_{S2}(t))$ and $(d\text{-}i_{S1}(t) < d\text{-}i_{S2}(t))$

Let $\delta2= (\upsilon\text{-}i_{S2}(t)-\upsilon\text{-}i_{S1}(t))$ and $\gamma2=(d\text{-}i_{S2}(t)-d\text{-}i_{S1}(t))$

Here schedule S1 is chosen as the better one. The reasons are similar to case-1. Delay efficiency of S1 over S2 is $[\delta2\{(\mu^H-1)/(\mu-1)+H\}\{(R_{min}+R_{max})/4vl_s\}+\gamma2]$.

Case-3: $(\upsilon\text{-}i_{S1}(t) > \upsilon\text{-}i_{S2}(t))$ and $(d\text{-}i_{S1}(t) < d\text{-}i_{S2}(t))$

Let $\delta3=(\upsilon\text{-}i_{S1}(t)-\upsilon\text{-}i_{S2}(t))$ and $\gamma3= (d\text{-}i_{S2}(t) -d\text{-}i_{S1}(t))$

This case corresponds to the situation where the schedule S1 produces more vulnerability but lesser delay compared to S2. Vulnerable nodes also produce delay in the network since the sources of live communication sessions through them have to rediscover routes to respective destinations and this route discovery takes substantial time. The average time required for one such route discovery is $\{(\mu^H-1)/(\mu-1)+H\}\{(R_{min}+R_{max})/4vl_s\}$. So, the average time required for $\delta3$ number of route discovery is $[\delta3\{(\mu^H-1)/(\mu-1)+H\}\{(R_{min}+R_{max})/4vl_s\}]$. If $[\delta3\{(\mu^H-1)/(\mu-1)+H\}\{(R_{min}+R_{max})/4vl_s\}] > \gamma3$, then S2 is chosen as better, otherwise S1 is chosen.

Case-4: $(\upsilon\text{-}i_{S1}(t) < \upsilon\text{-}i_{S2}(t))$ and $(d\text{-}i_{S1}(t) > d\text{-}i_{S2}(t))$

Let $\delta4=(\upsilon\text{-}i_{S2}(t) -\upsilon\text{-}i_{S1}(t))$ and $\gamma4=(d\text{-}i_{S1}(t) - d\text{-}i_{S2}(t))$

This situation is similar to case-3. If $[\delta4\{(\mu^H-1)/(\mu-1)+H\}\{(R_{min}+R_{max})/4vl_s\}] > \gamma4$, then S1 is chosen as better, otherwise S2 is chosen.

Case-5: $(\upsilon\text{-}i_{S1}(t) > \upsilon\text{-}i_{S2}(t))$ and $(d\text{-}i_{S1}(t) = d\text{-}i_{S2}(t))$

Let $\delta5 = (\upsilon\text{-}i_{S1}(t) - \upsilon\text{-}i_{S2}(t))$

Schedule S2 is chosen as better. The delay efficiency of S2 is $[\delta5\{(\mu^H-1)/(\mu-1)+H\}\{(R_{min}+R_{max})/4vl_s\}]$.

Case-6: $(\upsilon\text{-}i_{S1}(t) < \upsilon\text{-}i_{S2}(t))$ and $(d\text{-}i_{S1}(t) = d\text{-}i_{S2}(t))$

Let $\delta6 = (\upsilon\text{-}i_{S2}(t) - \upsilon\text{-}i_{S1}(t))$

Schedule S1 is chosen as better. The delay efficiency of S1 is $[\delta6\{(\mu^H-1)/(\mu-1)+H\}\{(R_{min}+R_{max})/4vl_s\}]$.

Case-7: $(\upsilon\text{-}i_{S1}(t) = \upsilon\text{-}i_{S2}(t))$ and $(d\text{-}i_{S1}(t) > d\text{-}i_{S2}(t))$

Let $\gamma7 = (d\text{-}i_{S1}(t) - d\text{-}i_{S2}(t))$

Schedule S2 is chosen as better. The delay efficiency of S2 is $\gamma7$.

Case-8: $(\upsilon\text{-}i_{S1}(t) = \upsilon\text{-}i_{S2}(t))$ and $(d\text{-}i_{S1}(t) < d\text{-}i_{S2}(t))$

Let $\gamma8 = (d\text{-}i_{S2}(t) - d\text{-}i_{S1}(t))$

Schedule S1 is chosen as better. The delay efficiency of S2 is $\gamma8$.

Case-9: $(\upsilon\text{-}i_{S1}(t) = \upsilon\text{-}i_{S2}(t))$ and $(d\text{-}i_{S1}(t) = d\text{-}i_{S2}(t))$

Any one of S1 or S2 is chosen. The delay efficiency is 0.

**Placing one particular message forwarding request within a schedule**

Consider a schedule S in the message queue of $n_i$ at time t.

S: $\Re_{j1}\ \Re_{j2}\ \Re_{j3}\ ...\ \Re_{jp}$

Here $\Re_{j1}$ and $\Re_{jp}$ are present at rear and front end of the message queue of $n_i$ at time t. Message forwarding request $\Re_j$ comes from $n_j$ to $n_i$ at that time. Different schedules are possible now including $\Re_j$ in S. They are,

Schedule 1: $\Re_j\ \Re_{j1}\ \Re_{j2}\ \Re_{j3}\ ...\ \Re_{jp}$
Schedule 2: $\Re_{j1}\ \Re_j\ \Re_{j2}\ \Re_{j3}\ ...\ \Re_{jp}$
Schedule 2: $\Re_{j1}\ \Re_{j2}\ \Re_j\ \Re_{j3}\ ...\ \Re_{jp}$

…

Schedule (p+1): $\Re_{j1}\ \Re_{j2}\ \Re_{j3}\ ...\ \Re_{jp}\ \Re_j$

Among all these schedules the one that is best in terms of vulnerability and delay index, is elected. In case of availability of more than one such schedule any one is elected arbitrarily.

Proposition 3: The upper limit of the sum total of waiting time of message forwarding requests in the message queue of a node $n_i$ is given by $\{(\kappa+R_i/v_s)\ m_i\ (m_i-1)/2\}$ where $\kappa$ is the distance independent time period required to serve a message forwarding request, $m_i$ is the size of the message queue of $n_i$ and $R_i$ is the radio-range of $n_i$.

Proof: At most $m_i$ number of message forwarding requests can be present in the message queue of $n_i$. In this context, let the schedule S : $\Re_1\ \Re_2\ \Re_3\ ...\ \Re_{m_i}$ is being followed by ni. So $\Re_1$ is that particular message forwarding request that will be served last because it is present at the rear end of the queue. The message forwarding request $\Re_{m_i}$ will be served before others because it is present at the front end of the queue. Hence the waiting time of $\Re_1$ is maximum. It is denoted by $T_1$ and defined in (11).

$$T_1 = \tau(\Re_2) + \tau(\Re_3) + \ldots \tau(\Re_{m_i}) \qquad (11)$$

Actually, $\tau(\Re_q) = (\kappa+\text{dist}_{i,q'}(t)/v_s)$

$\tau(\Re_q)$ is the time required to serve the message forwarding request sent from $n_q$ to $n_i$. It has got two components – one is constant ($\kappa$) and the other part is dependent upon the distance between $n_i$ and $n_{q'}$ where $n_{q'}$ is that particular downlink neighbour of $n_i$ to which the message sent from $n_q$ to $n_i$ needs to be forwarded. The time required by the wireless signal to traverse from $n_i$ to $n_{q'}$ is given by $(\text{dist}_{i,q'}(t)/v_s)$ where $\text{dist}_{i,q'}(t)$ is the distance between $n_i$ and $n_{q'}$ at time t and as already mentioned, $v_s$ is speed of the wireless signal. For any downlink neighbour $n_{q'}$ of $n_i$ $\text{dist}_{i,q'}(t)$ is less than or equal to $R_i$. So, $\tau(\Re_q) \leq (\kappa+R_i/v_s)$.

Therefore, $T_1 \leq (m_i-1)\ (\kappa+R_i/v_s)$
$T_2 \leq (m_i-2)\ (\kappa+R_i/v_s)$
$T_3 \leq (m_i-3)\ (\kappa+R_i/v_s)$
…
$T_{m_i-1} \leq (m_i-1)\ (\kappa+R_i/v_s)$

So, the total waiting time $W_i$ of all the message forwarding requests in the message queue of $n_i$ is defined in (12).
$W_i \leq (\kappa+R_i/v_s)(1+2+3+\ldots+(m_i-1))$
i.e. $W_i \leq (\kappa+R_i/v_s)\ m_i(m_i-1)/2 \qquad (12)$

## 3. ADVANTAGES PRODUCED BY DEV_NS

<u>DEV_NS reduces the message cost of the underlying routing protocol.</u>

Unlike the other scheduling algorithms existing in literature, DEV_NS concentrates on reducing the number of vulnerable nodes as well as the time of completion of communication sessions.

Case-1: Reducing the number of vulnerable nodes in a schedule reduces the cost of communication.

Proof: Let the total set of vulnerable nodes in a schedule S at time t be denoted as $UL_S(t)$. The set of alive communication sessions through $n_j \in UL_S(t)$ is given by $AC_j(t)$. For each such communication sessions if route discovery process needs to be initiated, the corresponding cost of messages, considering all the vulnerable nodes, is as follows:

$$\sum_{n_j \in UL_S(t)} |AC_j(t)|\ (\mu+\mu^2+\mu^3+\ldots+\mu^H)$$

Case-2: Reducing the time of completion sessions i.e. end-to-end delay reduces the cost of messages.

Proof: Consider one particular communication route RT: source = $n_1 \rightarrow n_2 \rightarrow n_3 \rightarrow n_4 \rightarrow \ldots \rightarrow n_d$ = destination. Assume that t2 denotes the timestamp of completion of communication of RT under DEV_NS in the chosen schedule and t1 denotes the least timestamp of completion of RT under DEV_NS corresponding to all the schedules other than the best one. Since DEV_NS always prefers the schedule producing least end-to-end delay (considering the presence of vulnerable nodes), so definitely t2 < = t1. If there is at least one node $n_j$ in RT that is not vulnerable at time t2 but may become vulnerable at time t1, then that will incur the additional cost of route discovery amounting to $|AC_j(t)| (\mu+\mu^2+\mu^3+\ldots+\mu^H)$. Let VL(RT,t1,t2) denotes the set of nodes in RT that were not vulnerable at time t2 but may become vulnerable at time t1. So, the additional message cost incurred for delaying the completion of communication in route RT from time t2 to time t1, is given by

$$\sum_{n_j \in VL(RT,t1,t2)} |AC_j(t)| (\mu+\mu^2+\mu^3+\ldots+\mu^H)$$

DEV_NS increases the data packet delivery ratio of the underlying routing protocol.

Reduction in cost of messages in DEV_NS automatically generates lesser signal contention and collision producing higher message packet delivery ratio.

DEV_NS increases the connectivity ratio of the network.

Reduction in cost of messages in DEV_NS significantly decreases the energy consumption in nodes. As an obvious result, the death rate of nodes greatly decrease preventing network partitioning up to a great extent, compared to other state-of-the-art scheduling schemes existing in literature. This improves the network connectivity ratio preserving the provision of good quality routes as much as possible.

## 4. SIMULATION RESULTS

The simulation is performed using ns-2 [15] simulator. An ad hoc network is modelled with nodes placed randomly within 1500×1500 square meter area. Each simulation is run for 600 seconds of simulation time. A free-space propagation model is used. Each source transmits data packets at a minimum rate of 2 packets per second and maximum rate of 20 packets per second. Three different underlying protocols are used – AODV, DSR and FAIR. The number of nodes in different simulation runs are 100, 200, 300, 400 and 500. The transmission ranges vary between 50-200 meters. The MAC protocol used is IEEE 802.11g. Size of packets is 80 bytes. In various simulation runs three different mobility models has been used. They are random waypoint, random walk and Gaussian.

### Performance metrics

The performance of the proposed scheduling scheme DEV-NS w.r.t. its state-of-the-art competitors ESS (energy-based scheduling scheme) and FPR (fuzzy-priority scheduler) is evaluated using the following metrics:

i) Cost of messages – It indicates the total message cost in the network throughout the simulation period.
ii) End-to-end delay – The end-to-end delay is averaged over all surviving data packets from their respective sources to destinations.
iii) Packet delivery ratio – It is the ratio of the number of packets received successfully and total number of messages transmitted.
iv) Network connectivity ratio – it is defined as the ratio of total number of connected nodes in the network to the number of disjoint nodes.

### Results and explanations

Figure 2 demonstrates the improvements produced by DEV-NS in terms of message cost compared to ESS and FPR. The reason behind the superiority of DEV-NS is that unlike its competitors it is concerned about the residual battery power of nodes and their relative velocity. The principle of DEV-NS is based on the concept that if a substantial number of alive communication sessions pass through one particular node having a high rate of energy depletion then message forwarding requests from that node shouldn't be kept waiting for long in message queue of some other node. Similarly, if velocity of a node $n_i$ relative to one of its successor downlink neighbour $n_j$ (at least one alive communication session through $n_i$ is using the link from $n_i$ to $n_j$) is high and a significant number of alive communication sessions pass through it then its message forwarding requests should be readily served;

otherwise it may happen that after $n_i$'s message forwarding request is served it's battery may become un-operational due to excessive exhaustion while some communication sessions are still alive in it or the link from $n_i$ to $n_j$ may break while some alive communication sessions through $n_j$ are still using the link from $n_i$ to $n_j$. These incomplete communication sessions will face link breakage and this happens much more in FPR and ESS compared to DEV-NS. In order to repair those broken links new route-request packets are injected into the network that terribly increases the message cost. Increased message cost causes more message contention and collision which prevents an increased number of data packets from reaching their respective destinations. Hence, automatically the packet delivery ratio decreases. It is evident from figure 3 that DEV-NS produces much more packet delivery ratio than ESS and FPR.

Since DEV-NS suffers from much lesser route-rediscovery sessions, it saves the battery power of network nodes that would otherwise have been wasted for repairing a route discovered earlier. As far as the connectivity ratio is concerned, a node may remain disjoint from the whole network for a significant period of time mostly due to the consumption of its entire battery power or consumption of battery power of all of its uplink and downlink neighbours or if it doesn't have any uplink or downlink neighbours due to its geographical position, mobility etc. Disjointness of a node from the network due to battery exhaustion is much lesser in DEV-NS as shown in figure 4 whereas disjointness due to mobility or geographical position is prominent in all the competitor protocols.

As far as the end-to-end delay is concerned performance enhancement produced by DEV-NS is tremendous. The selection of schedule in DEV-NS is completely based on the comparative delay produced by different schedules. The period T for which a message forwarding request from ni waits in the message queue of some other node nj, during that period ni could perform some more tasks that has practically been delayed by time period T. Except that particular message forwarding request present at the front end of the message queue of nj, all other requests have to wait till all its predecessors are served. DEV-NS computes the sum total of delay that is suffered by all message forwarding requests in the message queue of some node. Also it computes the additional expected delay that some of its alive communication sessions might have to face in order to repair the broken links. Reasons for breakage of links have been discussed earlier. Among all possible schedules, the one that suffers from minimum delay is elected by DEV-NS. So, inevitably the end-to-end delay generated by DEV-NS among its competitor scheduling policies and also substantially smaller by the same produced by them. The improvement is graphically illustrated in figure 5.

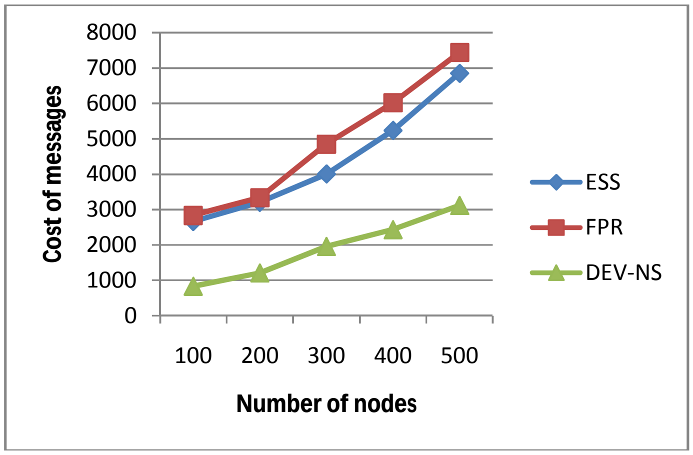

Figure 2: Graphical representation of cost of messages vs number of nodes

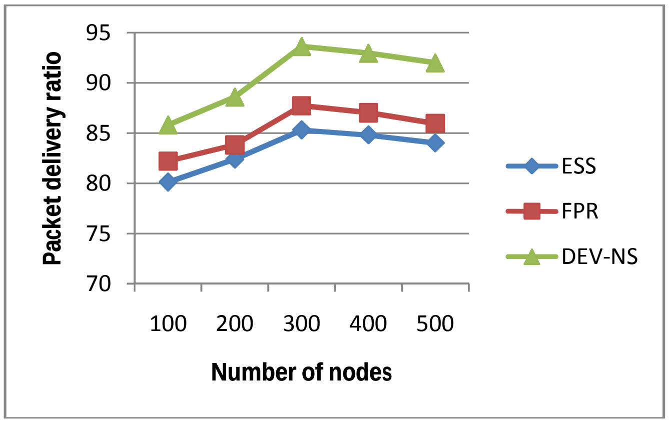

Figure 3: Graphical demonstration of packet delivery ratio vs number of nodes

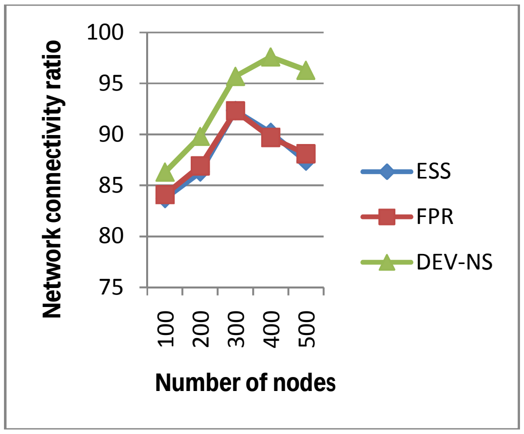

Figure 4: Graphical representation of network connectivity ratio vs number of nodes

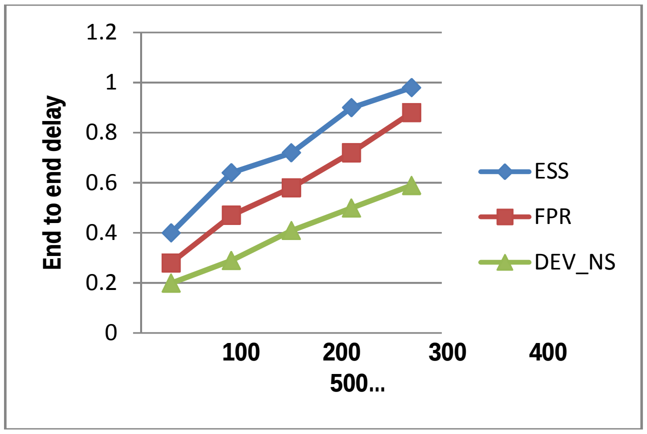

Figure 5: Graphical representation of end to end delay vs number of nodes

## CONCLUSION

Reducing the cost of route discovery is extremely important from the perspective of ad hoc networks because the nodes in ad hoc networks are battery powered. The lesser energy a node will spend for redundant route discovery, the greater will be its stored energy which it can meaningfully utilize for transmitting data packets from source to destination. DEV_NS is a scheduling mechanism that targets exactly that area. It is energy efficient, reduces end-to-end delay, improves data delivery ratio and automatically averts network partitioning, as much as possible. As a future scope, we shall look forward to enhance DEV_NS to include real time scheduling facility.

References:

[1]. Royer E., Perkins C: Multicast operation of the ad hoc on-demand distance vector outing protocol, In proceedings of MobiCom (August 2003)
[2]. Banerjee, P. Dutta, Fuzzy-controlled Adaptive and Intelligent Route Selection for Ad Hoc Networks, European Journal of Scientific Research, vol 45 no. 3, pp. 367-382, 2010
[3]. Banerjee, P. Dutta, A survey of unicast routing protocols for mobile ad hoc networks, International Journal of Engineering, Science and Technology, vol. 2 no. 10, pp. 5594-5604, 200
[4]. V. Kanodia et. Al., Distributed priority scheduling and medium access in ad hoc networks, vol. 8 no. 5, pp. 455-466, 2002
[5]. Gomathy and S. Shanmugavel, Fuzzy based priority scheduler for mobile ad hoc networks, ICN 2004, Gosier, Guadelope, 2004
[6]. S. Ghasempour et Al., A priority scheduler based QoS for dynamic source routing protocol using fuzzy logic in mobile ad hoc networks, The Journal of Mathematics and Computer Science, vol. 3 no. 3, pp.329-338,2011
[7]. Guodong Sun, Guofu Qiao and Lin Ziao, Efficient link scheduling for rechargeable wireless ad hoc networks, EURASIP journal on Wireless Communication and Networking, 2013, doi: 10.1186/1687-1499-2013-223
[8]. Inlab.lab.asu.edu/publications/Yinsha-12.pdf
[9]. L. Levin, M. Segal, Interference free energy efficient scheduling in wireless ad hoc networks, Ad hoc networks, vol. 11, issue 1, Jan 2013, pp. 201-212
[10]. Annadurai, Review of packet scheduling algorithms in mobile ad hoc networks, International Journal of Computer Appications, vol. 15, no. 1, Feb 2011
[11]. R. Kakaraparthi, R. Duggirala, D. P. Agarwal, Efficient message scheduling in ad hoc networks, IEEE NetComm 2000
[12]. Modiano, Scheduling packet transmissions in a multi-hop packet switched network based on message length, In proceedings of International Conference on Computer Communications and Networks, pp. 350-357, 1997
[13]. X. Yang and N. Vaidya, Priority scheduling in wireless ad hoc networks, Springer, Wireless Networks, vol. 12, 273-286, 2006
[14]. www.isi.edu/nsnam/ns