Chapter 4

A Delay Optimized – Energy Efficient Routing Algorithm using Van Emde Boas Tree
Chapter 4

A Delay Optimized - Energy Efficient Routing Algorithm using van Emde Boas Tree

4.1 Introduction

Energy efficiency is a predominant requirement for battery operated wireless nodes. Energy efficiency in wireless ad hoc networks can be optimized either by minimizing total energy consumption or maximizing network lifetime. In Chapter 3, an energy efficient routing algorithm for wireless ad hoc networks that aims to maximize the lifetime of the network has been proposed. Energy efficient routing algorithms minimize energy consumption at nodes and energy saving is associated with increased delay. This delay is unacceptable for many applications where information on nodes need to be transmitted to the sink node in time. Therefore, the challenge is to minimize both the total energy consumption as well as delay. However the two goals, namely minimum energy consumption and minimum delay, are usually conflicting in nature and thus, a trade-off in protocol design to provide QoS in an energy efficient manner rather than focusing only on QoS ignoring energy consumption is a required solution.

This chapter mainly aims to minimize total energy consumption of the network and end to end network delay in wireless ad hoc networks where data packets are transmitted from sources to the sink via multihop communication. Nevertheless simultaneous minimization of total energy consumption and delay is not always possible since, in many cases, minimization of one of the two usually requires
4. A Delay-Optimized Energy Efficient Routing Algorithm

sacrificing the other. Here, a Delay Optimized-Energy Efficient Routing Algorithm (henceforth called as DO-EERA) is being proposed to make a trade-off between total energy consumption and end to end network delay while selecting the next hop. DO-EERA balances the total energy consumption of the network and network delay by taking into account link energy consumption and link delay as a part of next hop selection process. That selection decision defines a next hop list for each node and favours a neighbor in the list (as a next hop) that minimizes both the energy consumption as well as delay. In addition, a weight factor is introduced to adjust the impact of energy and delay on the DO-EERA operation. The proposed method works through a series of iterations and terminates after a finite time. It is also shown that when the algorithm stabilizes, it produces delay optimized energy efficient routing paths from the sources to the sink. Moreover, using van Emde Boas tree [156] data structure, the proposed method is solvable in \(O(|V|^2 \log \log |V|)\) time (where \(|V|\) is the number of nodes in a network). Simulation experiments show that the proposed DO-EERA attains a good trade-off between total energy consumption of the network and end to end network delay in comparison to minimum energy routing (MER), [109] and [149].

Rest of this chapter is organized as follows. Section 4.2 presents the related works in brief for completeness. Section 4.3 describes network model and energy consumption model. Delay analysis is also briefly described in Section 4.3. The definition of the problem is stated in Section 4.4. The proposed delay optimized-energy efficient routing algorithm (DO-EERA) is described in Section 4.5. Correctness and complexity of DO-EERA is also addressed in Section 4.5. Simulation results are presented and analysed in Section 4.6. Finally, Section 4.7 concludes the chapter.

4.2 Related Works

Numerous algorithms have been proposed in the literature to improve energy efficiency [106,109,114,132-133] and to decrease delay [139-141,144]. MST based PEDAP [109] and minimum energy routing (MER) find energy efficient paths without incorporating delay in route construction process. Optimization of delay has been considered in several works [136-137,142]. Providing QoS in an energy
efficient manner is a new challenge both to prevent excessive energy usage of nodes and to transfer the information quickly to the sink. DERP [149] has been proposed for minimizing delay in terms of hop count. However, DERP does not consider energy consumption of node in next hop selection decision. Further, bounded delay based energy efficient routing algorithms have been proposed in [146-148] to improve QoS in a network. Moreover, several routing algorithms [150,153,155] provide delay guarantee on the time the packets reach to the sink while maximizing the lifetime of the network. Further details of these algorithms have been discussed in Section 2.6.2 in Chapter 2.

4.3 Preliminaries

This section derives the models that are used throughout this chapter.

4.3.1 Basic Network Model Used Here

A Wireless ad hoc network is modelled as an undirected graph \( G = (V, E) \) where \( V \) is the set of static nodes including the sink (SN) and \( E \) is set of links. Euclidian distance between any two nodes \( u \) and \( v \) is denoted by \( d_{uv} \). Each node in a network is characterized by a transmission range \( R_{max} \) which effectively defines the neighbors of that node - two nodes are said to be the neighbors if they are within the range of each other. Same as before, \( N_u \) is the set of 1-hop neighbor nodes of node \( u \). Here, \( N_u = N_u^D \cup N_u^U \) where \( N_u^D \) is the set of downstream neighbors of node \( u \) and \( N_u^U \) is the set of upstream neighbors of node \( u \).

The definition of upstream neighbor and downstream neighbor are given below for completeness.

**Definition 4.1**

**Downstream Neighbor Node:** A neighbor node \( v \) is said to be a downstream neighbor of node \( u \) i.e., \( v \in N_u^D \) if \( u \) chooses \( v \) as its next hop node for forwarding the packets to the sink.
Definition 4.2

**Upstream Neighbor Node:** A neighbor node \( v \) is said to be an upstream neighbor of node \( u \) i.e., \( v \in N_u^U \) if \( u \) receives the packets from node \( v \).

In addition, a set of nodes \( U \) (\( U \subseteq V \)) that generate data packets are called *sources*. For ease of analysis, it is assumed that all the nodes generate same number of packets. The number of packets passing through \((u, v) \in E\) is denoted by \( F_{uv} \), while \( R_{uv} \) is the maximum link capacity for link \((u, v)\). Without loss of generality, the link capacity is considered to be same for all the links i.e. \( R_{uv} = R_{vu} = R \).

### 4.3.2 Delay Analysis - End to End Delay and End to End Network Delay

Link delay has a great impact on end to end delay. Generally, four kinds of delay contribute to link delay: *queuing delay*, *transmission delay*, *processing delay* and *propagation delay*. This chapter mainly considers following delays: transmission delay \((T_D)\) over the link and queuing delay \((Q_D)\) at the node. Thus, the link delay \( D_{uv} \) over the link \((u, v)\) can be expressed as

\[
D_{uv} = T_D + Q_D
\]

\[
= \frac{F_{uv}}{R_{uv}} + \frac{F_{uw}}{R_{uv} - F_{uv}}
\]  

(4.1)

End to End delay (the time taken by a source to successfully deliver a packet to the sink) \([138, 147]\) is the summation of link delays that a packet experiences along the path from source to the sink. Hence, end to end delay from node \( u \) to \( SN \), will henceforth be referred to as *delay from node u* and is expressed by Eq. (4.2):

\[
D_u = \sum_{(u,v) \in p_u} b_{uv} D_{uv}
\]

(4.2)

where \( p_u \) is routing path from node \( u \) to the sink and \( b_{uv} \) is a binary variable. For each link \((u, v)\), binary variable \( b_{uv} \) is set to 1 when \((u, v)\) is on the routing path \( p_u \). Therefore, end to end network delay (the maximum end to end delay in data packet transmission in the network) \([147]\) is given as

\[
D_{tot} = \max_{u \in U} D_u
\]

(4.3)
4.3.3 Energy Consumption Model

The work presented in this chapter uses the energy model proposed in [18] for calculating energy consumption at a node. According to [18], energy consumed by wireless nodes is the sum of energy consumed for transmission and reception. Therefore, the energy costs for transmitting ($E_{tx}$) and/or receiving ($E_{rx}$) a $l$-bit packet between two nodes being distance $d$ apart can respectively be expressed as

$$E_{tx}(l, d) = lE_{elc} + lE_{amp}d^\alpha$$

(4.4)

$$E_{rx}(l) = lE_{elc}$$

(4.5)

where $E_{elc}$ represents energy per bit required by the transmitter/receiver electronics and $E_{amp}$ represents energy dissipated for transmitter amplifier and $\alpha$ is the usually path loss constant (especially, $2 \leq \alpha \leq 4$).

Therefore, based on Eq. (4.4) and Eq. (4.5), amount of energy consumed ($E_{uv}$) for transmitting a packet of size $l$-bit over the link $(u, v)$ is given by

$$E_{uv} = lE_{elc} + lE_{amp}d_{uv}^\alpha + lE_{elc}$$

(4.6)

Since path $p_u$ consists of multiple links $(u, v) \in \mathcal{E}$, total energy consumption along the path $p_u$ is

$$E_{pu} = \sum_{(u,v)\in p_u} b_{uv} E_{uv}$$

(4.7)

Now, using the nomenclature defined in this thesis, total energy consumption at node $u$ is denoted as $E_u$. However, $E_u$ includes only energy dissipation for transmission and reception of packets (i.e., it does not include energy consumption for overhearing). As mentioned in Chapter 3, $F_u$ and $F_v$ are traffic load on node $u$ and node $v$ respectively in bits. Therefore, if compression is not supported, total energy consumption at node $u$ is given by

$$E_u = \sum_{v \in N^u} F_v E_{elc} + F_u (E_{elc} + E_{amp}d_{uv}^\alpha)$$

(4.8)

Hence, the total energy consumption of the network, denoted by $E_{tot}$, can now be written as

$$E_{tot} = \sum_{u \in \mathcal{V}} E_u$$

(4.9)
Before proceeding further with the proposed work, a brief description of van Emde Boas tree is given in the next subsection for completeness.

4.3.4 Van Emde Boas Tree

Van Emde Boas tree is a data structure which implements an associative array of size $m$-bits where $m = \log M$ and $M$ is the size of the universe. It performs all the operations in $O(\log m)$ time or equivalently in $O(\log \log M)$.

To handle the universe $(M)$, $\sqrt{M} + 1$ sub-arrays are created where each of the sub array is itself van Emde Boas tree. Let, $\sqrt{M}$ subarrays are $a[0], a[1], \ldots a(\sqrt{M} - 1)$. Each sub-array $a[i]$ can handle a range of size $\sqrt{M}$ from the universe. Here, $a(\text{high}(x))$ is used as key to find the element in the sub-array where $x$ is an element of size $m$-bits and $\text{high}(x)$ gives most significant bits of $x$. There is also a summary array of size $\sqrt{M}$. In this structure, $a(\text{low}(x))$ is used to locate the element in summary array where $\text{low}(x)$ gives least significant bits of $x$. Now, for each value of $i$, if $a[i]$ is non empty then $i$ is written in summary array. Each sub-array $a[i]$ stores two values $a.\text{min}$ and $a.\text{max}$. The minimum value is stored in $a.\text{min}$ and maximum value is stored in $a.\text{max}$. If the value stored in $a.\text{min}$ is empty then that the corresponding subarray is empty. Each subarray also has a pointer that points to summary array.

Generally, a tree is represented in memory in an array. Van Emde Boas tree layout is mainly the collection of a set of complete binary trees. Let, height of the complete binary tree be $H$. Then, there are $(2^H - 1)$ number of nodes in the tree. The tree is divided into two sub-trees: top sub-tree and a set of bottom sub-trees.

The top sub-tree at height $\frac{H}{2}$ contains at most $(2^\frac{H}{2} - 1)$ nodes. In memory, the top sub-tree would be laid out and then each of the bottom sub-trees in order. Unlike the regular tree (e.g., breath first tree), it follows the sub logarithmic search time in order to access any of the child subtrees or to find and extract a minimum value element or find a successor as well as predecessor element. Further details are available in [156].
4.4 Problem Definition

For a given wireless ad hoc network which is represented as an undirected connected graph $G = (V, E)$ with $|V|$ number of sources, the main objective is to find a directed tree $A = (V, E')$ which minimizes both $E_{tot}$ as well as $D_{tot}$. Two objectives can be formalized as

$$\text{Minimize } \{ E_{tot} = \sum_{u \in V} E_u \}$$  \hspace{1cm} (4.10)

$$\text{Minimize } \{ D_{tot} = \max_{u \in V} D_u \}$$  \hspace{1cm} (4.11)

Subject to:

(i) For each $(u, v) \in E$, $b_{uv} = 1$ or $0$  \hspace{1cm} (4.12a)

(ii) For each $u, v, w \in V$, $\sum_{w \in N^u_v} F_{wu} b_{ku} - \sum_{w \in N^u_d} F_{uv} b_{uv} = 0$  \hspace{1cm} (4.12b)

(iii) For each $(u, v) \in E$, $0 \leq F_{uv} b_{uv} \leq R_{uv}$  \hspace{1cm} (4.12c)

(iv) For each $u \in V$, $\sum_{v \in N^u_v} F_v E_{elc} + F_u (E_{elc} + E_{amp} d^2_{uv}) \leq E_0$  \hspace{1cm} (4.12d)

A brief description of the conditions stated above is in order.

First, for each link $(u, v)$ in the network, an integer variable $b_{uv}$ is used to denote whether this link belongs to the path or not. Constraint (i) guarantees that $b_{uv} = 1$ if the link $(u, v)$ is selected for at least one of the path. Constraint (ii) represents traffic flow reservation property for an intermediate node – it first receives the packets from its upstream neighbors and then forwards them to the downstream neighbor. Constraint (iii) expresses the capacity reservation property which means that traffic flow over the link always be bounded by given link capacity. Finally, Constraint (iv) signifies that the total energy consumption at each node (written in left hand side) should not exceed initial energy ($E_0$).

4.5 Details of DO-EERA

This section presents the Delay Optimized - Energy Efficient Routing Algorithm (DO-EERA) that trades-off energy consumption in a network and end to end network delay. In other words, DO-EERA aims to minimize both the total energy consumption in the network and end to end network delay for the purpose of
forwarding the data packets from sources to the sink. Two parameters: (i) next hop list and (ii) link cost are proposed that affect both total energy consumption as well as the end to end network delay.

### 4.5.1 Next Hop List

Next hop list of node \( u \), denoted by \( N_{List}(u) \), is all or a subset of downstream neighbors to which \( u \) can forward the packet. In principle, \( N_{List}(u) \) should contain a set of candidate next hop nodes that can quickly deliver the packet to the sink. Any node \( v \in N_{List}(u) \) if the following conditions are satisfied:

1. Node \( v \) must be a downstream neighbor of node \( u \) so that excessive energy consumption as well as unnecessary delay for forwarding the packets to the sink can be avoided.
2. Maximum tolerable delay from node \( v \) is less than delay from node \( u \) i.e.,
   \[
   F_u D_v < D_u \quad (4.13)
   \]
3. Delay from node \( v \) should be smaller than delay from node \( u \) i.e.,
   \[
   D_{uv} + D_v < D_u \quad (4.14)
   \]

### 4.5.2 Link Cost Function

For each node \( v \in N_{List}(u) \), a link cost function is assigned. Energy cost and delay cost are the two main parameters in the cost function. Let, the cost function \( C_L(u,v) \) indicates node \( v \) ’s eligibility (or priority) as the next hop. At iteration \( r \), delay cost from node \( u \) to the sink via node \( v \) is given by

\[
D_u^v = D_{uv} + D_v (r - 1) \quad (4.15)
\]

where the first term is the link delay over \( (u,v) \) (c.f. Eq. (4.1)) and the second term is the delay from node \( v \) at iteration \( (r - 1) \).

Similarly, the energy cost from node \( u \) to the sink via node \( v \) is given by

\[
E_u^v = E_{uv} + E_{pv} (r - 1) \quad (4.16)
\]
where the first term denotes the energy consumption over link \((u, v)\) (c.f. Eq. 4.6) and the second term \(E_{p_v}(r - 1)\) denotes energy consumption of path \(p_v\) at iteration \((r - 1)\).

Therefore, from Eqs. (4.15) and (4.16), the link cost \(C_L(u, v)\) can be defined as

\[C_L(u, v) = D_u^v + E_u^v\]

\[= (D_{uv} + D_v(r - 1)) + (E_{uv} + E_{p_v}(r - 1))\]  \hspace{1cm} (4.17)

Since minimum energy consumption and minimum delay are usually conflicting goals, it is required to find a good trade-off between them. To achieve it, a weight factor, denoted by \(t\), is introduced in the proposed cost function. In Eq. (4.18), weight factor \(t\) determines the weight of energy consumption versus delay. Thus, the link cost function \(C_L(u, v)\) for \((u, v)\) is given by

\[C_L(u, v) = (1 - t) (D_{uv} + D_v(r - 1)) + t(E_{uv} + E_{p_v}(r - 1))\]  \hspace{1cm} (4.18)

4.5.2.1 Remarks on \(t\) : A New Feature for Balancing between Energy & Delay

The proposed weight factor \(t\) adjusts the impact of energy consumption and delay on routing decision. Weight factor \(t\) is a constant which lies between 0 and 1. In Eq. (4.18), the higher value of \(t\) gives more emphasis on the second term and the cost function minimizes energy consumption. On the other hand, for a smaller value of \(t\), the first term in the function dominates the second term and the function minimizes delay. The effect of weight factor \(t\) on DO-EERA in terms of energy and delay is detailed in Section 4.6.2.

4.5.2.2 Choice of Normalizing Coefficients: \(E_{tot}\) & \(D_{tot}\)

One of the important issues in the proposed DO-EERA is the choice of normalizing coefficients to normalize the parameters in the proposed cost function so that it can attain flexibility between them. However, coefficients must be chosen in such a way that it would not affect the optimization process. Taking these factors into consideration, \(E_{tot}\) and \(D_{tot}\) are chosen as normalizing coefficients for energy cost
and delay cost parameters respectively. With this choice, the normalized energy cost and normalized delay cost would be

\[
\text{Normalized } E_u^v = \frac{(E_{uv} + E_{pv}(r-1))}{E_{tot}(r-1)}
\]

\[
\text{Normalized } D_u^v = \frac{(D_{uv} + D_{pv}(r-1))}{D_{tot}(r-1)}
\]

Therefore, Eq. (4.18) can be rewritten as

\[
C_L(u, v) = (1 - t)[\text{Normalized } D_u^v] + t[\text{Normalized } E_u^v]
\]

\[
= (1 - t)\frac{(D_{uv} + D_{pv}(r-1))}{D_{tot}(r-1)} + t\frac{(E_{uv} + E_{pv}(r-1))}{E_{tot}(r-1)}
\]

(4.19)

Algorithm 4.1 shows the pseudo-code of the energy and delay based next hop selection algorithm, which selects the neighbor with the minimum \( C_L(u, v) \) as the next hop node.

\[\textbf{Algorithm 4.1: Next hop Selection for Node u}\]

1. \( N\_List(u) \) is the set of all or a subset of downstream neighbors of node \( u \)
2. for each node \( v \) in \( N\_List(u) \) do
3. \hspace{1em} Compute \( D_{uv} \) and \( E_{uv} \) using Eq. (4.1) and Eq. (4.6) respectively
4. \hspace{1em} Get \( D_{pv}(r - 1) \) and \( E_{pv}(r - 1) \)
5. \hspace{1em} Compute \( C_L(u, v) \) using Eq. (4.19)
6. end for
7. Select node \( v \) as next hop where \( v = \arg\min\{C_L(u, v) \mid v \in N\_List(u)\} \)
4.5.3 Implementation of DO-EERA

Before the sources generate the packets and forward them to the sink, DO-EERA will be executed to find the next hop nodes that minimize both the total energy consumption of the network as well as end to end network delay. DO-EERA is centralized in nature i.e., the optimization is performed by the sink node which will have all the necessary information to perform the task. The proposed DO-EERA is briefly described through Algorithm 4.3 in Section 4.5.3.4. It mainly consists of two phases stated in two consecutive sub-sections:

1. **Initialization Phase** (Section 4.5.3.1): This phase constructs an initial routing tree.
2. **Iterative Next Hop Node Selection phase** (Section 4.5.3.2): In this phase, each node selects the next hop node.

### 4.5.3.1 Initialization

In this phase, the proposed algorithm uses a shortest path algorithm to construct a routing tree of \( V \) rooted at the sink. The routing tree \( A \) contains minimum energy path from each source node \( u \) to the sink. For each path \( p_u \), the sink computes \( E_{p_u} \) as well as \( D_u \). After acquiring the routing information associated with each node, the sink builds the routing table RTAB with \((|V| - 1)\) number of entries, one entry for each node \( u \). RTAB contains following routing information associated with each node:

<table>
<thead>
<tr>
<th>Node id</th>
<th>next_hop(u)</th>
<th>( F_u(r) )</th>
<th>( D_u(r) )</th>
<th>( E_{p_u}(r) )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Node id</strong>: Identification number of node ( u )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>next_hop(u)</strong>: Next hop node of node ( u )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>( F_u(r) )</strong>: Traffic load on node ( u ) at iteration ( r )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>( D_u(r) )</strong>: Delay from node ( u ) at iteration ( r )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>( E_{p_u}(r) )</strong>: Total energy consumption along path ( p_u ) at iteration ( r )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4.5.3.2 Iterative Next Hop Node Selection

This phase works through a series of iterations. In DO-EERA, each iteration has two steps: (i) determination of next hop list and (ii) cost function based next hop selection. At the $r^{th}$ iteration, for each node $u$, following routing variables are updated:

- $D_u(r)$: Delay from node $u$ at iteration $r$
- $E_{p_u}(r)$: Total energy consumption along path $p_u$ at iteration $r$
- $E_u(r)$: Total energy consumption at node $u$ at iteration $r$
- $F_u(r)$: Traffic load on node $u$ at iteration $r$

These routing variables are updated considering the routing variables of each neighbor at $(r - 1)^{th}$ iteration. Before updating the variables, node $u$ checks the next hop list $N_{List}(u)$ and then calculates link cost $C_{l}(u,v)$ considering $E_{p_v}(r - 1)$ and $D_v(r - 1)$ of each node $v$ in $N_{List}(u)$. Here, the node with the minimum link cost is considered to be as the next hop. Once the node determines its next hop, the sink updates the following two routing variables considering the routing information available at $r^{th}$ iteration:

- $D_{tot}(r)$: End to end network delay at iteration $r$
- $E_{tot}(r)$: Total energy consumption of the network at iteration $r$

and finally propagates all updates in the network.

4.5.3.3 Routing Messages

Three different types of routing messages are used by the proposed DO-EERA: hop selection message (next_hop_select_msg), reply message and feedback message. These messages are mainly used in next hop selection phase of routing algorithm. The format of each routing message is given below for completeness.

**Hop Selection Message**

Hop selection message (next_hop_select_msg) is used by node having largest delay in order to find its next hop node.
4. A Delay-Optimized Energy Efficient Routing Algorithm

<table>
<thead>
<tr>
<th>Node id</th>
<th>$N_u^D$</th>
<th>$N_u^I$</th>
</tr>
</thead>
</table>

Figure 4.1: Hop selection message

Figure 4.1 illustrates `next_hop_select_msg`. The fields are defined as follows:

- **Node id**: Identification number of node $u$
- **$N_u^D$**: List of downstream neighbors of node $u$
- **$N_u^I$**: List of upstream neighbors of node $u$

**Reply Message**

After receiving hop selection message (`next_hop_select_msg`), only downstream neighbors of the sender node use this message to send the routing information to the sender. Reply message, shown in Figure 4.2, has the following fields:

<table>
<thead>
<tr>
<th>Node id</th>
<th>$D_v(r-1)$</th>
<th>$E_{pv}(r-1)$</th>
</tr>
</thead>
</table>

Figure 4.2: Reply message

- **Node id**: Identification number of neighbor node $v$
- **$D_v(r-1)$**: Delay from node $v$ at iteration $(r-1)$
- **$E_{pv}(r-1)$**: Energy cost of path associated with node $v$ at iteration $(r-1)$

**Feedback Message**

Feedback message is used by the sender node after receiving reply messages from its downstream neighbors. This message contains the routing information of neighbor nodes.

Figure 4.3 shows the feedback message. The fields of this message are given as follows:

<table>
<thead>
<tr>
<th>Node id</th>
<th>$D_v(r-1)$</th>
<th>$D_{uv}$</th>
<th>$E_{uv}$</th>
<th>$E_{pv}(r-1)$</th>
</tr>
</thead>
</table>

Figure 4.3: Feedback message
4. A Delay-Optimized Energy Efficient Routing Algorithm

- **Node id**: Identification number of neighbor node \( v \)
- **\( D_v(r-1) \)**: Delay from node \( v \) at iteration \( (r-1) \)
- **\( D_{uv} \)**: Link delay over \((u, v)\)
- **\( E_{uv} \)**: Energy cost of \((u, v)\)
- **\( E_{p_v}(r-1) \)**: Energy consumption of path associated with node \( v \) at iteration \( (r-1) \)

4.5.3.4 Detailed Description of DO-EERA

Let, \( S \) be a one dimensional array which is initially empty. At first, all the nodes are initialized as unmarked and inserted into \( S \). At the \( r^{\text{th}} \) iteration, the sink selects the unmarked node (say, node \( u \)) having largest delay; it then sends notification message to that node along the path \( p_u \). On receiving the notification from the sink, node \( u \) broadcasts 'next_hop_select_msg' message to its 1-hop neighbor nodes (\( N_u \)). A neighbor node \( v \) discards the message only if \( v \in N^1_u \). Otherwise, 'reply' message is send to the sender node \( u \). After receiving 'reply' message from all downstream neighbors, node \( u \) runs Delay-Minimization Algorithm. The pseudo code of Delay-Minimization procedure is given in Algorithm 4.2.

This algorithm determines a set of candidate next hop nodes that can quickly forward the data packets to the sink and put them into \( N_{\text{List}}(u) \). If \( N_{\text{List}}(u) \) is empty, then node \( u \) is marked. This marked node is deleted from \( S \) and algorithm enters into new iteration; otherwise, node \( u \) will send 'feedback' message to the sink. Feedback message of node \( u \) contains routing information associated with each node \( v \) in \( N_{\text{List}}(u) \). When the sink receives a feedback message from node \( u \), it calculates \( C_{l}(u, v) \) for each node \( v \in N_{\text{List}}(u) \) and selects node \( v \) with the minimum \( C_{l}(u, v) \) as the next hop. Once the next hop is selected, the sink joins the next hop node and node \( u \). At the end of selection process, all the marked nodes become unmarked and algorithm enters into next iteration. The iteration will continue until there is no more unmarked node is available in \( S \).
Algorithm 4.2: Delay – Minimization

1. for each node $v \in N_u^{D}$ do
2. \hspace{1cm} if $(D_v(r-1) + D_{uv} < D_u(r-1)) \& (F_u D_v(r-1) < D_u(r-1))$ then
3. \hspace{1cm} $N\_List(u) \leftarrow N\_List(u) \cup \{v\}$
4. \hspace{1cm} end if
5. \hspace{1cm} end for
6. Node $u$ sends 'feedback' message to the sink
7. return ($N\_List$)

Algorithm 4.3: DO – EERA: Delay Optimized – Energy Efficient Routing Algorithm

// List of Parameters Used

// $S$ is a list that contains unmarked nodes
// $r$ is iteration number
// $A$ is routing tree that contains routing paths from the sources to the sink
// next_hop() is an array that contains energy efficient next hop nodes
// $T_r$ is routing tree of $V$ at $r^{th}$ iteration

// List of Functions Used

// ExtractMax() [156] selects the node with maximum end to end delay
// Join () [156] adds a link between two nodes

Phase 1: Initialization Phase

1. $S \leftarrow V$
2. $r \leftarrow 2$
3. $p_u \leftarrow$ Minimum energy path for each node $u \in V$ to $SN$
4. $A = \{p_u\} \forall u \in V$
5. $T_{r-1} \leftarrow A$

Phase 2: Iterative Next Hop Node Selection Phase

6. Compute $D_u(r-1), F_u(r-1), E_{p_u}(r-1), \forall u \in T_{r-1}$
7. \hspace{1cm} while ($S! = \phi$)
8. \hspace{2cm} $u \leftarrow $ ExtractMax ($S$)
9. \hspace{2cm} $w \leftarrow $ next_hop($u$) on $T_{r-1}$
4. A Delay-Optimized Energy Efficient Routing Algorithm

10. Node $u$ broadcasts ‘$next\_hop\_select\_msg$’ to $N_u$
11. for each node $v \in N_u$ do
12. if $v \in N_u^U$ receives the message
13. Discard the message
14. else
15. Node $v$ sends ‘reply’ to node $u$
16. end if
17. end for
18. if $(N_u^D \neq \emptyset)$
19. Delay – Minimization ()
20. else
21. repeat steps (24) - (26) and go to step (7)
22. end if
23. if $(N\_List(u) = \emptyset)$ then
24. Mark node $u$
25. $T_r \leftarrow T_{r-1}$
26. $S \leftarrow S - \{u\}$
27. else
28. Remove the link between node $u$ and node $w$ from $T_{r-1}$
29. for each node $v \in N\_List(u)$ do
30. $C_L(u, v) \leftarrow (1 - t) \frac{(D_{uv} + D_v (r-1))}{D_{tot}(r-1)} + t \frac{(E_{uv} + E_{pv}(r-1))}{E_{tot}(r-1)}$
31. end for
32. $next\_hop(u) = \text{argmin}_{v \in N\_List(u)} C_L(u, v)$
33. join $(u, next\_hop(u))$
34. $T_r = T_{r-1} \cup \{u, next\_hop(u)\}$
35. Unmark all the marked nodes on $T_r$
36. Insert unmarked nodes into $S$
37. $r \leftarrow r + 1$
38. repeat steps (6) - (37)
39. end if
40. end while
41. return $(T_r)$
4.5.4 Correctness and Complexity of DO-EERA

First some useful lemmas are presented.

**Lemma 4.1** Number of comparison to find the next hop list $N_{\text{List}}(u)$ of any node $u$ is $\leq 3\lceil(N_u - 1)/2\rceil$

**Proof:** Let $D_v$ denote the delay from node $v$ at iteration $r$. Moreover, the link delay $D_{uv}$ is same for each link $(u, v)$ for each node $u$. Let, $d_{\text{list}}_u$ be the list that contains end to end delay of each neighbor of node $u$. Specially, delay list $d_{\text{list}}_u = [D_v]$ for each node $v \in N_u$. Using simultaneous minimum and maximum algorithm [156], it is possible to determine the next hop list $N_{\text{List}}(u)$ that would minimize the end to end network delay. Therefore, the next hop list $N_{\text{List}}(u)$ can have the following properties:

**Case 1:** if $D_u - D_{uv} < \min(d_{\text{list}}_u)$ then $N_{\text{List}}(u) \leftarrow \emptyset$

**Case 2:** if $(D_u - D_{uv} > \max(d_{\text{list}}_u)) \& (D_u > F_u \max(d_{\text{list}}_u))$ then $N_{\text{List}}(u) = |N_u - 1|$

**Case 3:** if $\min(d_{\text{list}}_u) \leq D_u - D_{uv} < \max(d_{\text{list}}_u)$ then $N_{\text{List}}(u) = \{v \in N_u - 1| D_v < D_u - D_{uv} \& F_u D_v < D_u\}$

Therefore, number of comparison is $\leq 3\lceil(N_u - 1)/2\rceil$.

**Proposition 1:** Let, $N_{\text{List}}(u) = \{v \in N_u - 1| D_v < D_u - D_{uv} \& F_u D_v < D_u\}$. Then $N_{\text{List}}(u)$ is unique and non-empty.

(i) Next hop list $N_{\text{List}}(u)$ may contain neighbor node $v \in N_u - 1$ that satisfies $(D_v < D_u - D_{uv} \& F_u D_v < D_u)$.

(ii) Next hop list $N_{\text{List}}(u)$ does not contain any neighbor node $v \in N_u - 1$ that satisfies $(D_v > D_u - D_{uv})$.

**Corollary:** If $N_{\text{List}}(u) \subset N_u - 1$ and $N_{\text{List}}(u) = \{v \in N_u - 1| D_v < D_u - D_{uv} \& F_u D_v < D_u\}$, the time complexity to find $N_{\text{List}}(u)$ is $O(|N_u|)$. 

Lemma 4.2  Algorithm ensures loop freedom.

Proof: At every iteration, a routing tree is constructed. Each node maintains the list of its downstream neighbors as well as upstream neighbors. On receiving ‘next_hop_select_msg’ message from a sender, each neighbor checks whether the node is the upstream neighbor of that sender or not. If the neighbor node is the upstream neighbor node of the sender, it will simply discard the message. On the other hand, only downstream neighbor nodes send reply messages to the sender node. At the end of iteration, a new routing tree is formed and all the nodes in the tree update their corresponding upstream neighbors as well as the downstream neighbors and thus ensure loop freedom.

Lemma 4.3  After $O(|V|\log |V|)$ pre-processing, next hop selection process takes $O(\log \log |V|)$ time.

Proof: At an iteration, the proposed algorithm needs to maintain two set of nodes, $S$ (where $S \subseteq V$) and $N_List(u)$ (where $N_List(u) \subseteq N_u$) for a node $u \in V$. In pre-processing, the nodes in both $S$ and $N_List(u)$ are sorted in chronological order. Search operation is performed on $S$ for finding the node with largest end to end delay. Searching is also performed on $N_List(u)$ for finding the node with smallest link cost. There are $O(|V|)$ search operations for the entire algorithm. If balanced binary search tree is used to store each of these two sets in which end to end delay and link cost are used as keys, then the algorithm takes $O(|V|\log |V|)$ time.

Here, van Emde Boas tree is used to search an element in a list and also to extract an element from that list. In pre-processing, the nodes are sorted by their end to end delay value and for each node; an integer $k$ is assigned as its key if the node is the $k^{th}$ one in the above sorted order. Thus, the position of all such keys form the integer set $M = \{1, 2, 3, ..., k\}$. Using van Emde Boas tree, each operation takes $O(\log \log |V|)$. Thus, after $O(|V|\log |V|)$ pre-processing, the next hop selection process for a node takes $O(\log \log |V|)$ time.
Lemma 4.4  The proposed routing algorithm is solvable in $O(|V|^2 \log \log |V|)$ time.

Proof:  According to Lemma 4.3, after $O(|V| \log |V|)$ pre-processing, the running time of next hop selection for a node becomes $O(\log \log |V|)$. Now, for any node $u$, it is known that $|N_u| \leq |V| - 1$. Therefore, next hop selection process run at most $(|V| - 1)$ times to find next hop node. Hence, the running time of next hop selection process for each node $u \in V$ is bounded by $O(|V| \log \log |V|)$. In the $V^{th}$ iteration, the algorithm stops and finds a final trade-off between the total energy consumption of the network and end to end network delay. Since each node takes $O(|V| \log \log |V|)$ time, the total running time of the proposed algorithm is $O(|V|^2 \log \log |V|)$.

Table 4.2 shows the time complexity for optimization using different data structures.

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Time complexity for pre-processing</th>
<th>Time complexity for optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Search</td>
<td>$O(</td>
<td>V</td>
</tr>
<tr>
<td>van Emde Boas tree</td>
<td>$O(</td>
<td>V</td>
</tr>
</tbody>
</table>

4.6  Performance Evaluation

In this section, the performance of the proposed DO-EERA is evaluated through simulations and the results are compared with MER, MST based PEDAP [109] and DERP [149] with respect to various performance metrics. The simulation experiments are conducted using MATLAB 8200a. Total energy consumption of the network and end to end network delay are considered as primary performance metrics to evaluate the effectiveness of the proposed algorithm. Another performance metric is delay variation, which is defined as the difference in maximum end to end delay and minimum end to end delay in the network.
4. A Delay-Optimized Energy Efficient Routing Algorithm

4.6.1 Simulation Setup

In this simulation study, it is assumed that 50 static nodes are randomly deployed in an area of $100 \text{ m} \times 100 \text{ m}$. Packets are generated at sources and relayed to the sink. The sink is chosen randomly. Moreover, the sink is responsible only for receiving the packets. The network is simulated by varying the number of nodes $|V|$ from 50 to 100. For each value of $|V|$, ten different topologies are generated and then values are averaged. Unless otherwise specified, transmission range of each node is 35 m and path loss constant is 2. Values of the necessary parameters used in simulations are listed in Table 4.3.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area</td>
<td>$100 \text{ m} \times 100 \text{ m}$</td>
</tr>
<tr>
<td>Type</td>
<td>Random</td>
</tr>
<tr>
<td>Network architecture</td>
<td>Homogeneous</td>
</tr>
<tr>
<td>Number of nodes ($</td>
<td>V</td>
</tr>
<tr>
<td>Transmission range ($R_{\text{max}}$)</td>
<td>35 m,40 m,45 m,50 m,55 m,60 m,65 m,70 m,75 m</td>
</tr>
<tr>
<td>Number of sources</td>
<td>49,59,69,79,89,99</td>
</tr>
<tr>
<td>Packet generation rate</td>
<td>1 packet / sec</td>
</tr>
<tr>
<td>Packet length ($l$)</td>
<td>1000 bits</td>
</tr>
<tr>
<td>Link capacity ($R$)</td>
<td>25 Mbps</td>
</tr>
<tr>
<td>Path loss constant ($\alpha$)</td>
<td>2, 3, 4</td>
</tr>
<tr>
<td>$E_{\text{elec}}$</td>
<td>50 nJ/bit</td>
</tr>
<tr>
<td>$E_{\text{amp}}$</td>
<td>100 pJ/bit / m²</td>
</tr>
</tbody>
</table>
4.6.2 Simulation Results and Analysis

End to End Network Delay

End to end network delay as a function of number of nodes for different routing algorithms is presented in Figure 4.4. In this experiment, the number of nodes is varied from 50 to 100. Figure 4.4 shows that the end to end network delay of the DO-EERA has significantly improved over MST based PEDAP, MER and DERP. Due to increased hop count of the routes, both PEDAP and MER have higher end to end network delay than DO-EERA. Though, initially the delay of DERP is lower, it increases with the number of nodes. On the other hand, the performance of DO-EERA is much better with the deployment of the more and more nodes. This reason behind this phenomenon is that the proposed method not only minimizes the hop count but also distributes the traffic load as evenly as possible among the nodes and consequently, reduces the queuing delay and transmission delay in the network.

![Figure 4.4: End to end network delay versus number of nodes](image-url)
Total Energy Consumption

Figure 4.5 presents the total energy consumption of the network as a function of number of nodes for MER, PEDAP, DERP and the proposed DO-EERA. In this experiment, the energy consumption of all schemes exhibits exactly reverse tendency for delays (shown in Figure 4.4). As seen from Figure 4.5, when the number of nodes increases, the total energy consumption decreases. Though the total energy consumption of MER is smallest, it has higher network delay. However, the proposed DO-EERA outperforms other two routing algorithms. This is due to the fact that, PEDAP increases the total energy consumption of the network due to the increased hop count of the routes. DERP neglects the impact of link energy consumption in next hop selection process. As a result, it increases total energy consumption in the network. By comparison, DO-EERA achieves a better trade-off between end to end network delay and total energy consumption of the network than MER and the other two routing algorithms.

![Figure 4.5: Total energy consumption versus number of nodes](image-url)
Delay Variation

Figure 4.6 presents the delay variation as a function of number of nodes for the proposed DO-EERA and the other three routing algorithms. Figure 4.6 reveals that the delay variation of the proposed method is better than PEDAP and MER. This is due to the fact that DO-EERA balances the traffic load at nodes more effectively, which consequently reduces the delay variation in a network. It can be seen that the delay variation curve of DO-EERA is very close to that of DERP. However, the total energy consumption of DO-EERA has lower than that of DERP as shown in Figure 4.4.

![Graph showing delay variation versus number of nodes](image)

Figure 4.6: Delay variation versus number of nodes

Trading off Energy Consumption with End to End Network Delay for Various Weight Factor

Figure 4.7 (a), 4.7 (b) and 4.7 (c) demonstrate a trade-off between total energy consumption and end to end network delay with 50-node network for the proposed
DO-EERA for various weight factor \( (t) \) and different values of the path loss constant \( (\alpha) \). Here, path loss constant \( \alpha \) is varied from 2 to 4. It is clear from the figures that the weight factor \( t \) in DO-EERA has a significant impact on the network performance. From Figure 4.7 (a) to Figure 4.7 (c), it is observed that end to end network delay decreases with \( t \), while total energy consumption of the network increases with \( t \). Figure 4.7 (a) and 4.7 (b) show that trade-off between the total energy consumption and end to end network delay becomes balanced when value of \( t \) approximately equals to 0.65. On the other hand, Figure 4.7 (c) displays that the trading off overall energy consumption with end to end network delay is better when value of \( t \) is set to either 0.15 or 0.8 (approximately). Note that, the two curves cross each other regardless the value of \( \alpha \) which signifies trade-off between total energy consumption and end to end network delay.

![Figure 4.7](image)

Figure 4.7: Trading-off total energy consumption with end to end network delay for different values of path loss constant. (a) Path loss constant \( \alpha = 2 \)
4. A Delay-Optimized Energy Efficient Routing Algorithm

Figure 4.7: Trading-off total energy consumption with end to end network delay for different values of path loss constant. (b) Path loss constant $\alpha = 3$ (c) Path loss constant $\alpha = 4$
Trading off Energy Consumption with End to End Network Delay for Various Transmission Range

Figure 4.8 and Figure 4.9 present end to end network delay and total energy consumption of the network as a function of transmission range with 50-node network for MER, DERP and the proposed DO-EERA respectively. In both experiments, the transmission range is varied from 35 m to 75 m. For the proposed method, results are obtained using various $t$ values. Figure 4.8 shows that the end to end network delay of different algorithms decreases as the transmission range increases. The larger value of transmission range implies smaller hop count and thereby, minimizes the delay. However, the end to end network delay of the proposed method lies between MER and DERP. As shown in Figure 4.8, DO-EERA reduces the end to end network delay when smaller value of $t$ is used.
4. A Delay-Optimized Energy Efficient Routing Algorithm

It can be seen from Figure 4.9 that the total energy consumption of the network increases as the transmission range increases. With larger transmission range, DERP finds the route with larger hop distance and thus consumes more energy. However, the proposed DO-EERA is more energy efficient than DERP. The reason behind this result is that link energy consumption is considered in next hop selection in DO-EERA, which minimizes total energy consumption of the network. From Figures 4.8 and 4.9, it can be said that by adjusting the value of $t$, DO-EERA finds more energy efficient as well as minimum delay routes compared to other methods. In other words, DO-EERA achieves a good trade-off between them.

4.7 Conclusion

Energy efficiency and end to end network delay are two important parameters for wireless ad hoc networks when the data packets should be reached to the sink in energy efficient as well as timely manner. However it is difficult to minimize both the metrics simultaneously, it suggests a protocol design that makes a trade-off between them. In this chapter, a delay optimized energy efficient routing algorithm (DO-EERA) is proposed to achieve an efficient trade-off between total energy
consumption of the network and end to end network delay. DO-EERA balances total energy consumption and network delay while selecting efficient next hop node. A cost function is designed to select the next hop. The proposed cost function takes into account both energy and delay for balancing these metrics. Furthermore, a weight factor is introduced to weight impact of energy cost and delay cost on DO-EERA operation. Extensive simulations show that the proposed method can achieve a good trade-off between total energy consumption of the network and end to end network delay. The next chapter considers the effect of interference in networking in designing routing algorithms.