Annexure I

The architecture of the simulator is described in this section. Few snapshots showing the functioning of the simulator are also given at the end of this section.

A. Simulator Architecture

A simulator program for simulating the routing protocols for wireless sensor network is developed using C++. This kind of simulator development is motivated by the work presented in [75]. The entire simulator is divided into several modules. Each module takes input regarding different parameters of the network and the protocols to be simulated. Subsequently these modules produce results which are used for analysis of the protocols.

Various modules of the protocols are: Deployment module, Topology construction module, Mobility management module, Medium access control module, Routing module, Data transfer module, Transmission error handling module, Energy expenditure computing module, and Throughput computing module. In this section, the overall architecture of the simulator is discussed. The functions involved in each module of the simulator are mentioned. The input and corresponding output of the modules are discussed.

Deployment module: The specification i.e., dimension of the sensor field is taken from the user as an input. This module specifies the sensor field with respect to a two dimensional coordinate system. This module also deploys the sensor nodes randomly in the sensor field. Each sensor node is represented by a point in the two dimensional coordinate system. This point representing a sensor node is chosen randomly from within the specified sensor field area. The Base Station is also considered as a point only but located outside the sensor field. The number of sensor nodes to be deployed in the field is specified by the user at the beginning of the simulation. The sensor field is represented through a two dimensional array. Each point in this 2D array is a sensor node. Similarly the Base Station is also represented by a pair of coordinate values.

Topology construction module: The sensor network is represented by a graph. This graph consists of some vertices and some edges. The sensor nodes deployed by the deployment module are the vertices of the graph representing the sensor network. Depending on the physical distance between two sensor nodes and their corresponding radio ranges a communication link is established between the two nodes. If the physical distance (i.e., Euclidean distance) between two nodes $m$ and $n$ is lesser or equal to the radio range of the sensor node $m$ then a directional link is established from $m$ to $n$ ($m \rightarrow n$). Similar computation is done for $n$ also, in order to establish the reverse link from $n$ to $m$. This link is nothing but an edge between $m$ and $n$ in the graph. Thus the edges are formed among all the nodes and the Base Station. The topology construction module creates the graph considering all the deployed sensor nodes and the Base Station. Lists are used for
replacing the wireless sensor network after establishing the links. Figure 1 shows an instance of the network topology after deployment of the nodes in the field.

![Network topology](image)

Figure 1: Network topology constructed after deployment of the nodes

**Mobility management module**: The sensor nodes as well as the Base Station move with certain velocity. It is assumed that the sensor nodes as well as the Base Station follow random waypoint mobility model [59]. The coordinate values of the nodes and the Base Station change according to the displacement of these nodes. The function \( \text{displacement}(\cdot) \) changes the coordinate values of the nodes according to the random waypoint mobility model with respect to time. It is assumed that the nodes do not cross the specified sensor field borders and the Base Station moves within a specified region. The nodes may hit a border and then it reflects back towards the specified sensor field. The new locations of the nodes and also the Base Station are computed after a regular time interval which is specified at the time of simulation. The function \( \text{link \_compute}(\cdot) \) computes the connectivity of the nodes while it is called. This computation is done depending on the locations (i.e., coordinate values) of the nodes. Lists are used for storing the links in the network.

**Medium access control module**: The sensor field is partitioned into several virtual clusters according to the routing protocol under simulation. Each cluster contains several sensor nodes with different roles such as gateway, cluster head, deputy cluster head, ordinary sensor nodes etc and these roles are determined and assigned as per the routing protocol. Again depending on the routing protocol the cluster members are assigned different medium access control slots. This module is responsible for storing medium access control slots assigned to different nodes in the field. According to the routing protocol under consideration the nodes are given different time slots to communicate their data packets. Generally a particular cluster setup is valid for an optimum time interval which sometimes called as cycle. Each node inside a cluster is given a medium access time slot considering this cycle and it is determined by the routing protocol itself. A data structure (i.e., array) is maintained for each node in which the medium access control information is stored. This information subsequently is used for
further data and control information communication. The medium access control information is updated according to the policies of the routing protocol under evaluation.

Figure II Different modules constituting the Simulator

**Routing module**: This module takes the route information between a pair of sender/receiver node from the user as input. The routing protocol under simulation determines the route between a sender and receiver node. The sender node usually is a sensor node deployed in the sensor field and the receiver or destination node is the Base Station. The intermediate nodes in such a route are determined by the routing protocol under simulation. Then the user gives this intermediate node information to the simulator through external interface. The routing module takes this input and considers this route information for further transport of the data packets and the subsequent energy expenditure analysis. For each sender node a data structure (i.e., array) is maintained in which the route information (i.e., the intermediate nodes in the route towards the Base
Station) is stored. This information is updated according to the policies of the routing protocol under evaluation.

**Data transfer module**: This module keeps track of the data transferred by each node. Depending on the data rate considered in a sensor network setup and according to the routing policies of a particular routing protocol, different sensor nodes transmit as well as receive different amount of data. There are two functions namely, `transmit()` and `receive()`, designed for each node in the field. These two functions keep track of the amount of data transfer (i.e. transmit and receive) happening through each node in the field. The amount of the data transfer is computed analytically by considering parameters like data rate, different errors in transmission, and routing policies. Error computation is discussed in detail in the transmission error handling module. The values of the variables which store the data received and data transferred by each node in the field are updated over time and this part is carried out by the functions `transmit()` and `receive()`. Theses values are further considered for computing energy expenditure and throughput under the influence of a particular routing protocol. This module takes input from the medium access control module, routing module and transmission error handling module.

**Energy expenditure computing module**: This module keeps track of the energy expenditure by each sensor node in the field. There are several sources of energy expenditure such as data transmission, data reception, data processing, idle listening, collision, overhearing etc. Out of all these sources, idle listening, collision and overhearing are the causes of energy waste. There is a function in this simulator namely, `energy_consumption()`, which is designed to compute total energy expenditure incurred at each node due to various sources of energy expenditure. In this simulator, four agents are designed for four different kinds of energy expenditure sources. These agents are `transmit()`, `receive()`, `data_processing()` and `idle listening()`. We do not consider energy waste due to collision and overhearing at this moment. Thus the `energy_consumption()` function at each node computes the total energy expenditure considering agent wise energy expenditure at each node. The results of this module are used in energy expenditure analysis and energy efficiency analysis of the routing protocol under evaluation.

**Transmission error handling module**: Error occurs during data transmission. The two major kinds of errors in such a wireless sensor network setup are link error and node error. As already mentioned, the routing protocol determines the routes in the network. The transmission error handling module considers all those nodes and links involved in a route and keeps track of the error that may occur in these nodes and links. There are two functions namely, `node_error()` and `link_error()`, designed for keeping track of the errors while a routing protocol is under simulation. The function `node_error()` at each node computes the node error and its effect on the data to be transmitted from that node. The function `link_error()` at each link computes the link error and its impact on the communicated data through that particular link. The user specifies the node error probability and link error probability at the time of simulation which are taken as input by the functions `node_error()` and `link_error()` respectively. This module helps in computing actual amount of data received by a node from a particular link (after
considering the link error). It also helps in computing actual amount of data to be transmitted by a node after considering node error. The results of this module are subsequently used by the data transfer module and throughput computing module.

Throughput computing module: Throughput is the measure which indicates the actual volume of data delivered successfully at the destination node through a particular route against transmission of some data from a particular source node. This module considers each session of data transmission between a sender node (sensor node) and the destination node (Base Station) under the influence of the routing protocol which is under evaluation. Then it computes the throughput of that routing protocol. For computation of throughput across a particular route, all the intermediate nodes are considered along with the source node. Transmission error handling module computes the node error and link error that may occur in the intermediate nodes and the links in the route. Due to these errors the final volume of data delivered at the end of the route i.e., the Base Station is reduced. Thus throughput of a particular route is computed as the ratio between the final volume of data delivered at the Base Station and the actual volume of data transmitted by the source node. It is important to mention that the final volume of data delivered at the Base Station depends on the node error and link error involved in a particular route.

B. Some interactions with the simulator

Following are some snapshots showing the functioning of the simulator. Figure III shows how various input for different parameters are taken from user in order to create the topology of the network. Figure IV shows the interface through which different parameters are collected regarding radio range of the nodes, velocity of the nodes and the interval at which the new locations of the sensor nodes are to be computed. Figure V shows the positions of different nodes at a particular instant of time along with the respective velocity of the nodes. Figure VI shows a sensor network setup in which different sensor nodes are directly connected to their respective cluster head nodes.

![Figure III Taking various input for topology construction](image-url)
Figure IV Taking various input regarding mobility of the nodes

Figure V Showing various nodes’ position and respective velocity

Figure VI Showing Base Station and various nodes along with respective cluster head

These are only few interactions with the simulator. The entire simulator is constructed in order to facilitate performance evaluation of different routing protocols.