ANT COLONY OPTIMIZATION FOR ENERGY EFFICIENT ROUTING IN WIRELESS SENSOR NETWORKS

- Guide

P. Sateesh Kumar

Assistant Professor

CSE department, IIT ROORKEE

- By

K. Uday Kanth Reddy

Abstract

Wireless Sensor Networks are spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature etc and to cooperatively pass their data through the network to a main location.(Here throughout the project base station which is the sensor node connected to the computer is the main location).They are generally characterized by specific requirements such as limited energy availability, low memory and reduced processing power. A large number of network routing algorithms have been developed to effectively pass this sensor data from individual nodes to base station. Through this project a new Wireless Sensor Network protocol based on Ant Colony Optimization heuristic is studied and effectively implemented using isense hardware and software platform for Wireless Sensor Netoworks.

KeyWords

Ant Colony Optimization, WSNs, heuristic, phermone, isense

Problem Statement

To find an energy efficient path from source to destination along with minimum no of hops.

Introduction

Wireless Sensor Networks have captured the attention of researchers and students over the past decade because of the diverse applications they support and the flexibility of network deployment options they provide. These advantages, along with the remarkable advances in sensor technology, make WSNs a favourite option for many tracking and monitoring applications. In such applications, sensors nodes can collaboratively monitor the network environment and report real-time information about the monitored phenomenon.

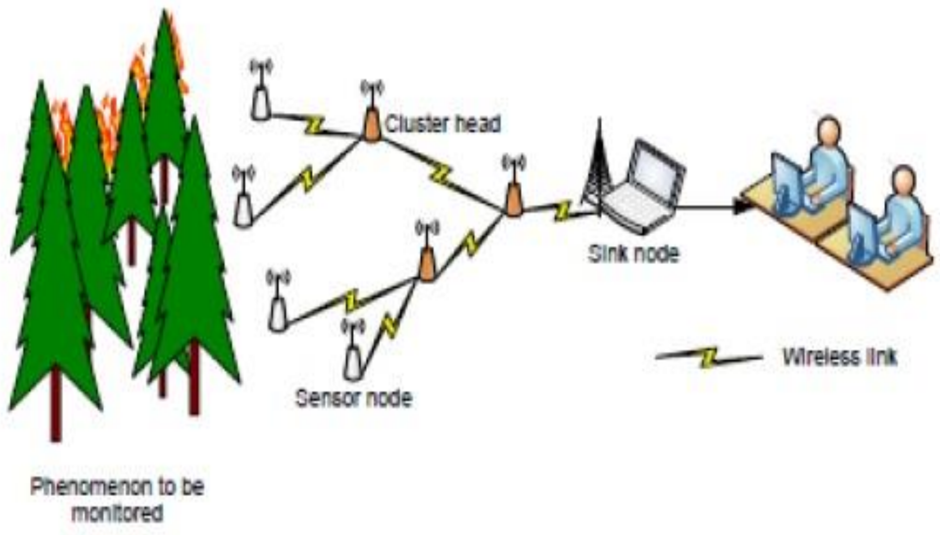


Figure 1.1 Architecture of a Typical WSN

Constraints of WSNS

- Due to the fast development of the microprocessor, sensor and transceiver, there is great applications foreground about WSNs.
- Also since we often use these networks in rough and inaccessible environments such as battlefields, volcanoes, forests and so on, normally there is low possibility to change or recharge the defective or dead nodes. Hence, the main difference between WSNs and other classic wireless networks is that WSNs are hypersensitive and vulnerable to energy.
- The limited energy is the key issue influencing WSNs.

So, routing algorithms have to be designed to maximize the life of WSNs

ACO

- A family of Ant Colony Optimization(ACO) algorithm have been tested in past for checking their efficiency(Here, efficiency is calculated in terms of time till the first node dies) in routing data in WSNs.
- The efficiency found was comparatively high as compared to other routing algorithms.

- Through this project ACO for energy efficient routing has been implemented using special parameters.
- The main objective of the algorithm is to maximize the network lifetime by carefully defining link cost as a function of node remaining energy.

Coalesenses isense

- Coalesenses is a German Company providing solutions for massively distributed systems with a focus on wireless sensor networks (WSNs).
- Coalesenses originates from a university background in this new application area and holds on to the concept of cooperation with public research facilities.
- Their WSNs modules are prefixed with ISENSE.
- Website(http://www.coalesenses.com)
- In this project we have used four different types of modules and an adapter from Coalesenses company.

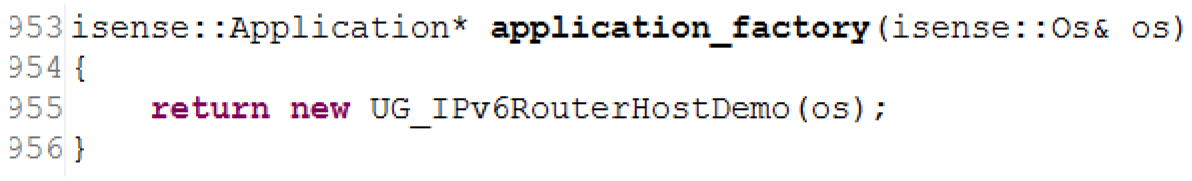
Becoming Acquainted with the softwares

The following softwares have been used in implementing the project

1. Coalesenses ishell
2. Isense sdk
3. Eclipse development environment
4. Cygwin ba-elf2 compiler

Application Basics

Each and every application consists of class UG_IPv6RouterHostDemo which contains all the required functions to be executed and a method called

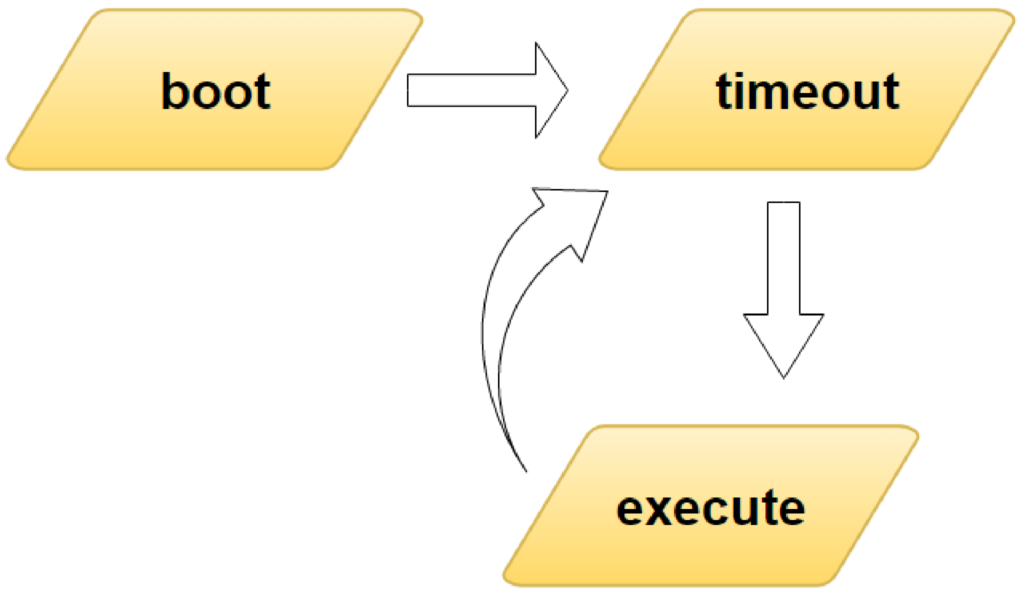


```
isense::Application* application_factory(isense::Os& os)
{
    return new UG_IPv6RouterHostDemo(os);
}
```

This method shall return a new instance of our application to our operating system(operating system of core module. In future I would be referring to the same) and is always called first before any other method is executed.

Control Flow

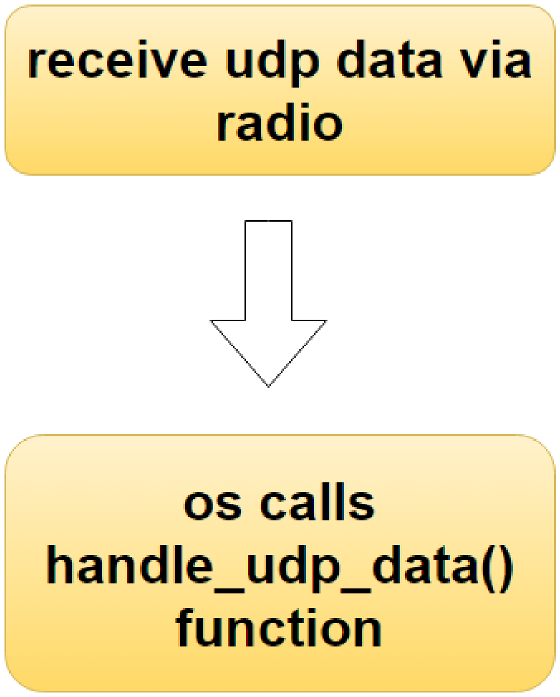
The control flow of every application as follows:-



After an instance of the application has been returned to the operating system the boot method of the application is called in which all the variables and constants related to the application are initialized and the timeout function of Time 60 seconds is called so that the module can wait for the entire network to get initialized before sending ants.

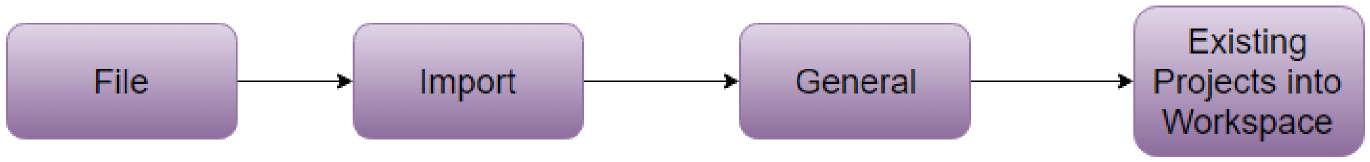
Another control flow which is present in the application is the udp data handling.

Whenever a node receives data through radio it is passed to the udp instance which shall be instantiated while creating the ip instance of the module and this udp instance shall call the handle_udp_data function.


receive udp data via radio
os calls handle_udp_data() function

Creation, Compiling and Uploading

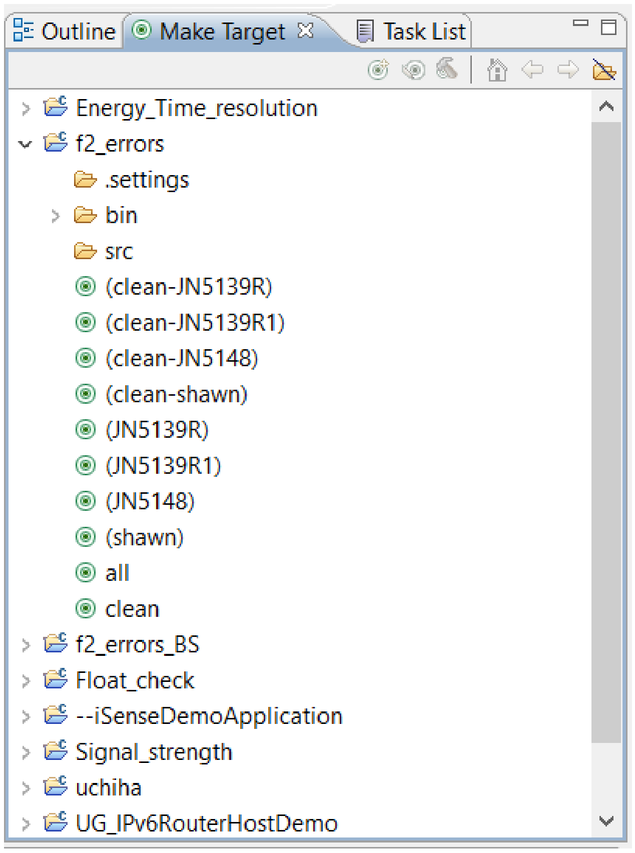
Creating a new project is done in the following way



Then you can browse one of the demo projects available which can be downloaded from the coalesenses website and start making changes as per your requirement.

The entire code has been written in eclipse IDE for C++ and compiled using the "Make Target" option available on to the right. Here is a snapshot of that column

Since the processor of our core module is JN5148 the make target of JN5148 has been used.



```
> Energy_Time_resolution
v f2_errors
    .settings
  > bin
    src
    (clean-JN5139R)
    (clean-JN5139R1)
    (clean-JN5148)
    (clean-shawn)
    (JN5139R)
    (JN5139R1)
    (JN5148)
    (shawn)
    all
    clean
> f2_errors_BS
> Float_check
> --iSenseDemoApplication
> Signal_strength
> uchiha
> UG_IPv6RouterHostDemo
```

Exploring Routing Options

There are several routing options available in isense_sdk and can be found in the following path

"isense_sdk\iSense\src\isense\protocols\routing"

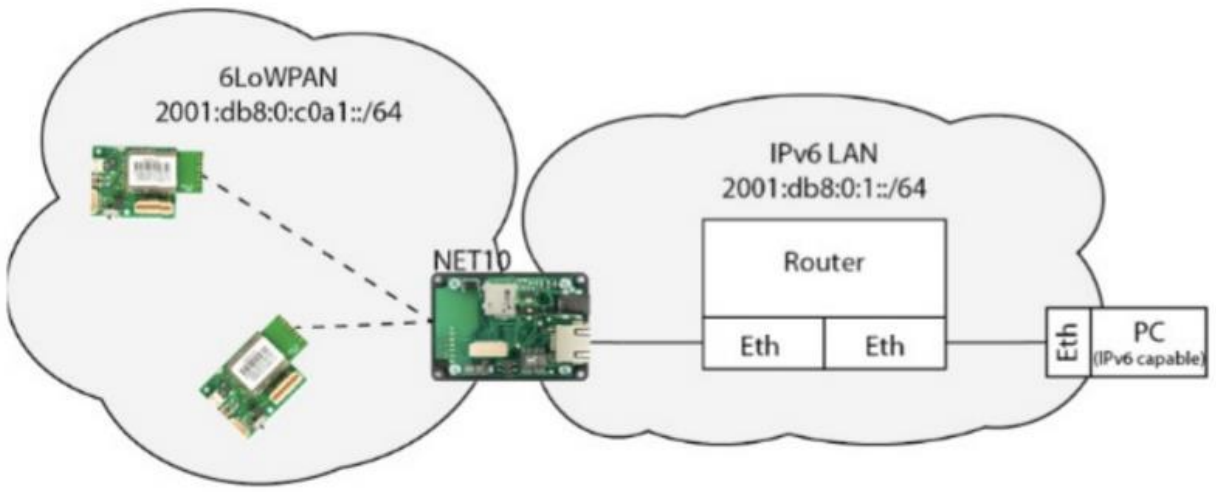
Two of the protocols with which I have experimented was Bidirectional quality routing and Bidirectional Source routing. The Bidirectional quality routing tries to create routes along particularly stable links. Choosing links with low packet loss rate is done using Neighbourhood Monitor and the link metrics it provides .These are forwarded and considered during route creation. Note that this protocol does not necessarily choose the optimal route, it may get stuck in local optima. Resulting routes usually work fine, but are not perfect.

The modules are fitted with a Zigbee radio which offers high data rates upto ranges of 600m while providing hardware AES encryption.

IPv6 Support

Coalesenses offers an IPv4 and IPv6 dual network stack to easily integrate wireless sensor nodes into the internet. It comprises all functionality required for connecting wireless sensor networks with existing Ethernet installations using the internet protocol family. There is 6LoWPAN(IPv6 over Low Power Wireless Personal Area Networks) which includes all the standard function performed by the Ipv6 layer. The IPv6 datagrams are transmitted to the IEEE 802.15.4 link layer radio interface.

The assumed topology by the Coalesenses company for using 6Lowpan is as follows:-



But the non-availability of NET10 ethernet gateway and a Router for dynamically assigning IPv6 address to the sensor nodes posed a difficulty.

This was solved by using Neighbouring detection which would provide Linklocal IP address which was further used in Ant Colony Routing algorithm.

Adopted Communication Scheme

The different layers that have been used for the communication between two nodes are

1) Application layer
2) Udp layer
3) 6LoWPAN
4) Link layer
5) Physical layer

Application layer

The code we write is in the application layer. It is the top most layer in any application. The application layer passes the data to the Udp layer and receives data from the udp layer through udp socket.

Udp layer

The pointers used for accessing Udp layer and udp socket are as follows:-

```
Udp* udp_;
UdpSocket* udpSocket_;
```

The udp layers passes the data received from application layer to the network layer i.e 6LoWPAN layer.

6LoWPAN

Pointer for accessing the 6LoWPAN layer:-

```
SixlowpanNetworkInterface* ip_if0_;
```

It is called as network interface because it acts a medium between Udp layer and Radio

Link Layer

The link layer comprises Radio which uses the IEEE 802.15.4 protocol.

Physical layer

The medium between two nodes is air which is the physical layer.

The instantiation of the above mentioned pointers has been done in boot() method i.e the starting phase of every module.

Setting up

At first an instance of the application layer is created by calling the isense::Application(os) function in the Constructor of the application.

The boot method first creates an instance of IPv6. The IPv6 layer will automatically create an instance of UDP which is stored in pointer udp_.

An UDP socket is created on port 8080 using pointer udp_ for which the application registers itself as UdpDataHandler.

A SixlowpanNetworkInterface is created by passing it a reference to the IEEE 802.15.4 radio, adding to the IPv6 layer and registering it with the link layer interface and a reference is stored in ip_if0_.

Receiving udp data

Whenever an UDP packet on port 8080 is received the handle_udp_data() method of the application is called.

Sending udp data

Sending udp data is required in the execute() method. The send method of the udpSocket_ is used for this purpose. An IPv6 address representing the target address is used in the send method. While creating the 6LoWPAN network interface it is automatically added to the IPv6 layer and is used for sending the udp data to further lower layers and receive data form these layers. This is necessary because while sending the data using LinkLocal address the outgoing interface cannot be determined using the target address. The data is send in the form of character array.

ACO

The first week was used for studying about Computer Netwroks and the next week was used for getting familiarized with isense modules and setting up the environment. From the third week, information regarding ACO was gathered through various research papers, and how to implement this routing algorithm in WSNs.

Understanding ACO

- A lot of species of ants have a trail-laying/trail-following behavior when foraging.
- While moving, individual ants deposit on the ground a volatile chemical substance called pheromone, forming in this way pheromone trails.
- Ants can smell pheromone and, when choosing their way, they tend to choose, in probability, the paths marked by stronger pheromone concentrations.
- In this way they create a sort of attractive potential field, the pheromone trials allows the ants to find their way back to nest.
- Also they can be used by other ants to find the location of the food sources discovered by their nestmates.

Implemented Algorithm

- The proposed protocol includes route discovery and route maintenance process.
- In the route discovery process, the sensor nodes establish all valid paths to the destination node by sending a query packet (forward ant), if the ants find the destination node, then the destination node generates a response packet (backward ant).
- The backward ant goes back to the sending node along the reverse path, and releases pheromone while it returns.

- The concentration of pheromone is in inverse proportion to the distance to the destination node(Here, distance is measured in number of hops) and directly proportional to the average of remaining amount of energy level of the nodes on the path.
- Once all the paths are set up, the source node begins to release the data packets and the packets transmit along the path with the highest pheromone concentration.
- In route maintenance process, the sensor nodes send a certain number of forward ants to the destination node periodically to monitor the quality of the existed.

Transition Probability

- In ant colony optimization, each ant attempts to find the optimal path. In their journey from source node to destination node, ants move from node i to a neighbouring node j with a transition probability

$$P_{ij}[k] = \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{l \in N_i[k]} \tau_{il}^{\alpha}\eta_{il}^{\beta}}, j \in N_i[k],$$

Where *$P_{ij}[k]$* is the probability for ant k to move from node i to node j, *T_{ij}* is the pheromone value deposited on link (I,j), η_{ij} a heuristic value assigned to the link, and (α, β) are weights used to control the importance given to the pheromone and the heuristic values, respectively. Here, $N_{ij}[k]$ is the list of neighbours of node I visited by ant k.

The pheromone value *$T_{ij}[k]$* on a given link depends on the likelihood that the ants pass by the link while constructing the solutions.

The heuristic value η_{ij} on the other hand, depends on the calculated cost of the link. The initial pheromone value is usually set to be equal for all links. The heuristic value of the link (i ,j) is the inverse of the link cost, C_{ij}.

The heuristic variable is calculated as follows.

$$\eta_{ij}(t) = \frac{e_j(t)}{\sum_{s_l \in C(i)} e_l(t)}$$

where ej(t) is the remaining amount of energy of neighbor j for node i. The denominator is the sum of remaining energy levels of all neighbours for node i.

Phermone Enhancement

When the forward ant gets to the destination node, the destination node generates a backward ant, and sends it back along the reverse path.

The backward ant in each visited node release a certain amount of pheromone $\Delta\tau$ and it is given by

$$\Delta\tau = c \times (HOP_{\max} - hop_{count}) \times Eavg_n$$

where c is the variable parameter, HOP_{max} represents the maximum allowed number of hops for forward ants and data packets in the network; hop_{count} represents the number of hops forward ant has taken to reach the destination node.

Phermone Update Rule

Therefore, when node receives the response packet from the destination node by the nth neighbor node, the node will update the pheromone concentration $\tau_{n,d}$

$$\tau_{n,d} = (1-\rho) \times \tau_{n,d} + \Delta\tau$$

where ρ is the pheromone evaporation coefficient, 1- ρ is the pheromone residue factor. The range of pheromone is from [0,1].

The proposed algorithm considers both energy and distance. In order to take into account the movement of destination node the above formula is slightly modified as

$$\tau_{n,d} = (1-\rho)\times\tau_{n,d} + \Delta\tau / \omega\cdot hop_{count}$$

where 'w' is a control factor for controlling the influence of no of hops over pheromone update rule and HOPcount is the no of hops travelled by the backward ant.

In order to limit the maximum and minimum values of pheromone values to the range between 0 and 1 the minimum amount of pheromone for a particular neighbor is 0.005 and the maximum value is 0.9.

In my project temperature data has been used as the phenomenon that is being monitered by Wireless Sensor Networks.

The data and forward ants have been sent at specific intervals and entire network is made to operate at a single synchronized time for coordination between forward ants and data packets by using time synchronization protocol provided in isenseSDK.

ACO Algorithm

ACO for individual nodes (Other than Base Station)

```
for each individual node other than BS do:
    • initialize parameters alpha = 1, beta = 2, rho = 0.2, w = 0.6)
    • detect neighbours using Flooding and Neighbourhood Monitor
    • Receive Energy levels of Neighbouring nodes and initialize pheromone
      values for all neighbouring nodes to 0.5
    • Start Route Discovery Process by sending forward ants in all possible
      paths.
    • Receive Phermone Updates.
    • Send temperature data to Base station.
    • Enter Route Maintenance process
end for
    Route Maintenance process = {
        1. Send forward ants to the Base station at regular intervals by using
           Transition probability Formula.
        2. Receive Neighbouring Energy levels.
        3. Forward Neighbouring Node Ants when received and check for loops
           and TTL of the ant
        4 . Receive Phermone Updates by Backward Ants. }
```

ACO for Base Station

The Base Station has the function of sending Backward Ants which carry the feedback of the path followed by forward ant. The following pseudocode explains the function of Base Station. The following actions are performed when the forward ant is received and handle_udp_data() function is called by the processor.

Receive Forward Ant ={

- Access Memory of Forward Ant.
- Calculate Avg energy level of the path and $\Delta\tau$ for this forward ant.
- Create Backward Ant by using the path travelled contained in the memory of Forward Ant.
- Send Backward Ant back to its source }

Receive Temperature data = {Display Temperature data on screen using ishell}

Implementing ACO in WSN – Challenges And Solutions

The first challenge was how to represent an ant in the given WSNs. In the simulations that are available online the program has access to the entire network and also the ants. So the program shall take care of coordination between ants and the update rule. In practical case Ants have to be forwarded from one node to the other and each node should respond accordingly.

Also the representation of food source which is the destination for which ants are searching a path and the feedback calculation that is usually done by the ants have to implemented in the given WSNs.

Solution:-

Each and every ant was represented as a character array and this character array would carry the path followed by the forward ant till now. Even the backward ant should also carry the feedback of the path followed by the forward ant and the path followed by the forward ant.

Data Structure of Forward ant

Identifier of ant	Source Identifier	Integer part of average energy	Decimal part of average energy	Average energy level of path	No. of hops travelled	Hop identifier 1	Hop identifier 2	…

Data Structure of Backward ant

Identifier of ant	Source identifier	Integer part of $\Delta\tau_k$	Decimal part of $\Delta\tau_k$	No of hops to be travelled	No of hops travelled	Hop Identifier 1	Hop identifier 2	…	…

Identifiers for different data

In order to identify the kind of the data received the following identifiers have been use:-

1) "**200**" for energy data
2) "**202**" for forward ant
3) "**203**" for backward ant

The source identifier shall contain the id of source from which the forward ant has been created.

Note:-

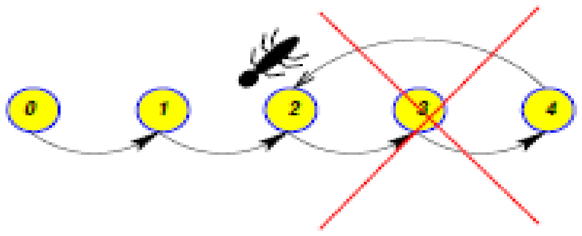
1) The sequence of hop identifiers for backward ant shall be in the opposite direction of forward ant. So the last node before reaching Base station for forward ant shall be the first node for Backward ant.
2) The decimal part is up to two decimal places.
3) No. of hops to be travelled by backward ant is the No. of hops travelled by forward ant.

4) Whenever an ant reaches a node the no.of hops travelled is incremented and its corresponding identifier is added to the list and the average energy level is updated.
5) The maximum number of hops has been limited to 10 as the total size of network is of 30 nodes.

Further Insight

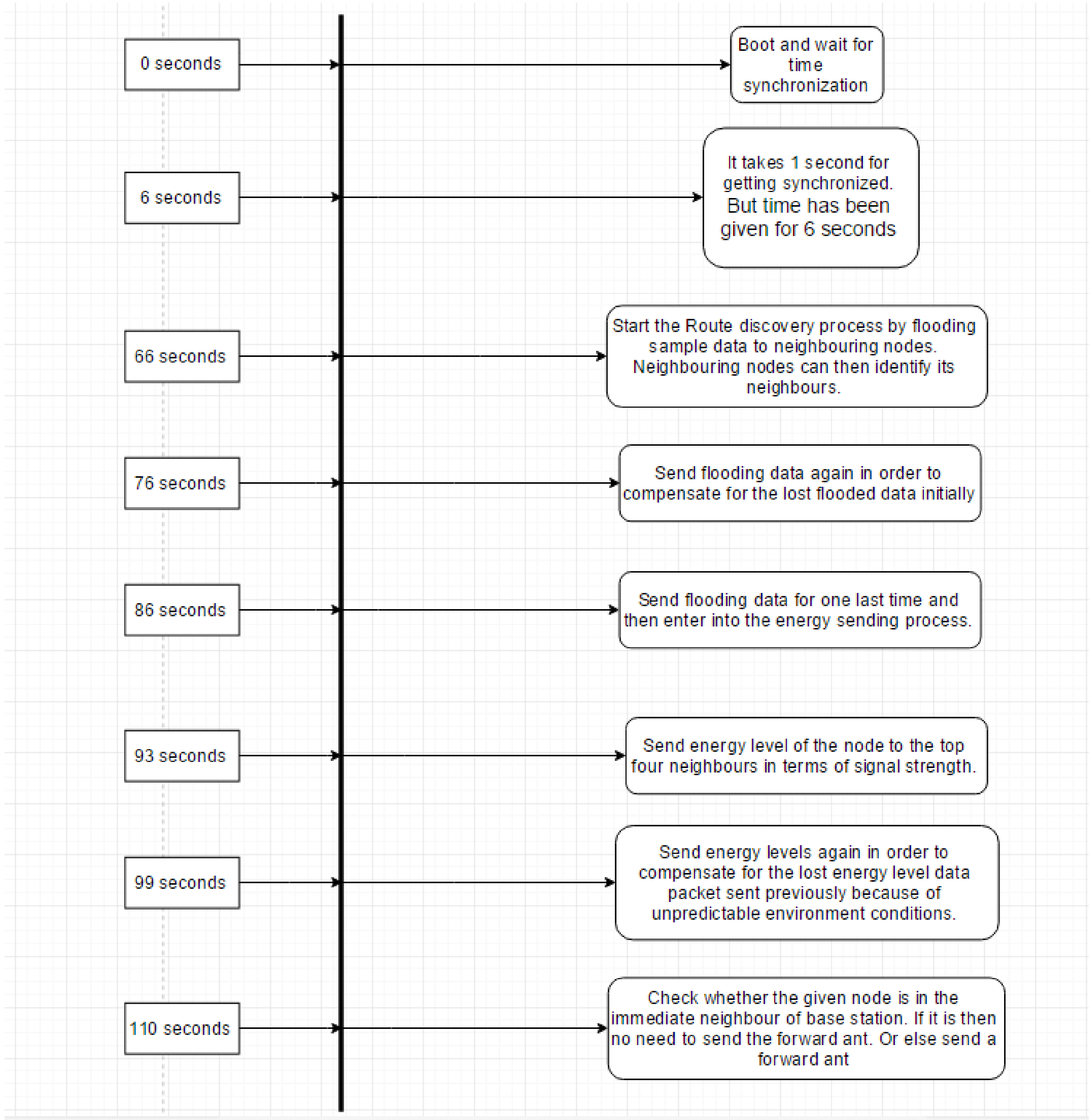
If a cycle is detected, that is, if an ant is forced to return to an already visited node, the nodes composing the cycle are taken out from the ant's internal memory, and all information about them is destroyed. If the ant moved on a cycle for a time interval which is greater than half of its age at the end of the cycle, the ant is destroyed.

(Suppose an ant has reached a node and that node is its sixth hop. If the node is already present in the list of hops covered by the ant and suppose the no of hops between the first time it has reached that node and now is greater than half of the no. of hops i.e 3, then the corresponding ant is destroyed.)



Initially the idea of including time of the sensor nodes was thought to be implemented but that would add additional memory to the ant as now each and every node should be associated with the corresponding time at which it has reached that node.

The following time frame was followed for the successful implementation of WSNs by taking practical problems into consideration.


0 seconds
Boot and wait for time synchronization
6 seconds
It takes 1 second for getting synchronized. But time has been given for 6 seconds
66 seconds
Start the Route discovery process by flooding sample data to neighbouring nodes. Neighbouring nodes can then identify its neighbours.
76 seconds
Send flooding data again in order to compensate for the lost flooded data initially
86 seconds
Send flooding data for one last time and then enter into the energy sending process.
93 seconds
Send energy level of the node to the top four neighbours in terms of signal strength.
99 seconds
Send energy levels again in order to compensate for the lost energy level data packet sent previously because of unpredictable environment conditions.
110 seconds
Check whether the given node is in the immediate neighbour of base station. If it is then no need to send the forward ant. Or else send a forward ant

Depending upon the position of node two types of time frame are followed after following the above time frame

1) If the node is in the top four neighbours of Base station then the Base station shall also be in the top four neighbours of the corresponding node. Then there is no need to send a forward ant as the base station is in the vicinity of node.
2) If the node is not in the top four then the corresponding node shall start sending the forward ants and the above process of flooding and neighbourhood detection is repeated.

Code Insights

Sensor Nodes

The sensor nodes shall maintain the next neighbor in the shortest path from the current node to the Base station.

Each and every sensor has the following variables in order for the shortest path process to work.

1. **float** rho;
2. **float** phermone[6];
3. **float** energy[6];
4. **float** hueristic[6];

As mentioned before these variable shall store the necessary information.

All these variables are private.

There are also two additional variables for maintaining the Route discovery and Route maintenance process .

1. **bool** desti
2. **uint8** f_ND
3. **uint8** normal
4. **uint8** energy_c

1. The purpose of first variable is to keep track of whether the destination is in the top four neighbours of the present node. (In order to limit the no. of neighbours a node can have only the top four neighbours have been considered. This is because the range of range of radio signal is very high and even a node at a distance of 600m can be detected.)
2. f_ND is used to count the no. of times we are sending the flooding data to neighbours.
3. The variable normal is used to indicate that the route discovery process has been done and route maintenance process has been initiated
4. The last variable is used to count the no. of times the energy level has been sent to the neighbouring nodes.

Note:- In the SDK written by isense char ,int , long and long int have been aliased to uint8, uint16, uint16 and uint32

Neighbourhood Detection

All data packets picked up by the core module's radio are registered at the dispatcher. Also a list of addresses of the nodes sending the packet along with information like LQI value, RSSI value, and Last Heard Time are stores in real time by the Neighbourhoood Monitor.

These lists are returned to our application by the 'get_neighbour" function of the class NeighbourhoodMonitor. It is checked at this stage whether the base station is a neighbor node.

Energy level measurement

The Energy level is measured by including the header file "isense/modules/core_module/coremodule.h" in the application.

We can then create a pointer to the Coremodule class by writing

Coremodule* cm_

Then we can access the supply_voltage () function of the coremodule class. This is the voltage that is fed into the regulator of the coremodule. The function of regulator is to control the level of voltage in case of short circuit or any other fault.

Time synchronization

The time synchronization protocol has been used for the smooth functioning of the algorithm and switching from one phase to another.

In order to use time synchronization protocol the following header file "isense/protocols/time_sync/confirm_time.h" has to be included.

Then we can have access to the Time synchronization protocol by creating a pointer in the private section and then defining in the boot method.

```
TimeSync* ts_;
ts_ = new TimeSync(os());
```

There are two types of nodes under time syncronizatio. The master node which launches time synchronization and the remaining nodes which request for time synchronization. Both these function can be done in the following way.

```
ts_->launch_time_sync();
ts_->request_time_sync(ISENSE_RADIO_BROADCAST_ADDRESS);
```

Either one of the above two functions has to be called for a single node. Multiple time synchronization launches in the network has to be avoided.

Note:- ISENSE_RADIO_BROADCAST_ADDRESS indicates that any neighbor in the vicinity of node shall respond.

Base Station

The code for Base station is simple. It shall receive forward ants and shall send backward ants to the source.

Key Challenges while programming and adopted solutions

- Limited Memory Capability:- As it is a characteristic of WSNs of having limited memory capability many times code was optimized in order to satisfy memory constraints.
- No Support for float:- Unfortunately, there was no support for float for the processor JENNIC 5148 which was attached to the core module. Considerable amount of time was invested in solving this issue and soft dynamic link library routines provided by gcc were used in order to solve this issue.
- Initial Setup for project:- The amount of help content provided in web and at the site mentioned was not even close to the minimum help required to get started with the kit. So, large amount of time was invested in familiarizing ourselves with the kit and in the worst case by using hit and trail methods.

As mentioned above the support for float was added in the following way:-

1) add libgcc.a library to the following path “isense_sdk\iSense\lib\jennic\5148_1v4”
2) uncomment “ LIBFILE += $(JENNIC_SDK_PATH)/libgcc.a” command in Makefile.jennic.5148 file.

Results

The network was tested on a wide variety of network topologies with no. of nodes in the range of 30 nodes. In majority of the cases shortest path was found.

In few cases ants got stuck into loops and have to be removed from the network.

Optimal network path with best combination of energy level and no. of hops was found.