DYNAMIC SENSOR ROUTING

Mentor: Sanjay P Josh

Team Members:

Chandrika B.H. (01FB15ECS075)

Amogh K.M. (01FB15EEC026)

Sagar V. Belavadi (01FB15EEC190)

Sukruthi S. Rao (01FB14EEC205)

Date: 22/7/2016

Microsoft Mobile Innovation Lab Summer Internship

June-July 2016

ABSTRACT

Wireless Sensor Networks (WSNs henceforth) are used in places where human involvement is not safe. After installation, human interference must thus be minimal- whether the battery runs out or whether a link between two nodes is broken. Both the mentioned problems must be dealt with robustly by the network itself. A common choice to counter the latter problem is to implement the network in a star topology. But this topology is not always feasible because situations may necessitate a node to be placed out of the range of the coordinator (called "Base Station"). Routers are used to assist the communication between the sensor nodes that act as the source of data packets and the Base Station which is the final sink to all data packets of the network. Each node is also a router in our project. It is the goal of this project to implement a WSN that will robustly handle failure of individual nodes. With this goal in mind, we have implemented dynamic routing. Dynamic routing means that the path taken by the data packet between the source sensor node and the base station is not pre-defined. Instead, each data packet chooses its own path on the basis of selection of certain metric at each router. We have developed two ways to make sure this happen. Both the methods use the same approach but the implementations vary. In addition to accomplishing this main goal, we have countered the problem of battery drain by fixing the wake-sleep cycles for the R.F. transceiver. It comes 'ON' only during a fixed period of time when all the packet routing happens. It then turns 'OFF' till the next such time it should send its own data packet or route other data packets.

CONTENTS

NOTE: All references have been included as foot notes in the respective pages.

INTRODUCTION

WSNs are one of the most exciting and challenging research domains. They are gaining rapid world-wide attention mainly because of their potentially low cost solutions to a variety of real-world challenges. Many other favouring factors of WSN's use are self-organizing, self-healing, having dynamic network topology to cope with node malfunctioning and failures, mobility of deployed nodes, unattended operation, ability to withstand bad environmental conditions, scalability at the time of deployment and after deployment, as well as easy use. It is because of these factors that they have a great potential to be deployed in wide mission-critical applications such as military monitoring, health care and civilian applications.

Since WSNs are generally deployed in areas that are far too unsafe for repeated human intervention, the network must be robust in the face of node failure. And one of the common causes of node failure is that the scarce battery resources may eventually drain out. To counter this problem of node failure, sleep-awake cycles are generally implemented and dynamic routing is practiced.

Dynamic routing is the practice of having the network intelligently work out the path that the data packet generated must take in order to reach the destination.

Energy aware routing is also a commonly used approach in dynamic routing. In this method, the metric used to calculate the best path is the amount of available battery resources in that route.

All of this being done and said, everything boils down to the implantation of these ideas. Often simulations of these sensor nodes are done to develop models and algorithms. We on the other hand are using a real WSN kit that has been provided by the Microsoft Mobile Innovation Lab to ensure reliability and realistic accuracy to the codes implementing the algorithms developed.

PROBLEM DEFINITION

Develop and implement multi hop dynamic routing algorithms in wireless sensor networks with actual sensor nodes.

OBJECTIVES

The objectives of the said project are as described:

1) Implement a star network of sensor nodes which communicates wirelessly (802.15.4 IEEE standard 2003)

2) Implement a static multi- hop tree network of the same sensor nodes.

3) Implement a network that is valid in any topology whose features are as given below:
   - Dynamic routing: The data packet routed through the network should be able to choose its own path on the basis of certain information contained in each router/node it passes through.

   - Energy optimization: Each node must save the power in its batteries as totally draining the battery causes the node to permanently fail (The batteries can't be replaced as human involvement is dangerous in certain regions). Thus, wake-sleep cycles are implemented for the RF transceiver of nodes. All nodes come on at the same time and transmit their data packets, route other data packets it receives and then goes to sleep till the next such time it should do those tasks. For this, time synchronization between each node is essential. But, we haven't been able to implement this in the current project.

   - Packet collision avoidance: If all the data packets are being sent in the same time interval, there is a high chance of packet collision. This must necessarily be avoided.

   - Node Failure: The relevant network members should be able to identify the failure of a node and correct the "information" it contains and propagate it; that information which enables each packet to decide which route to take.

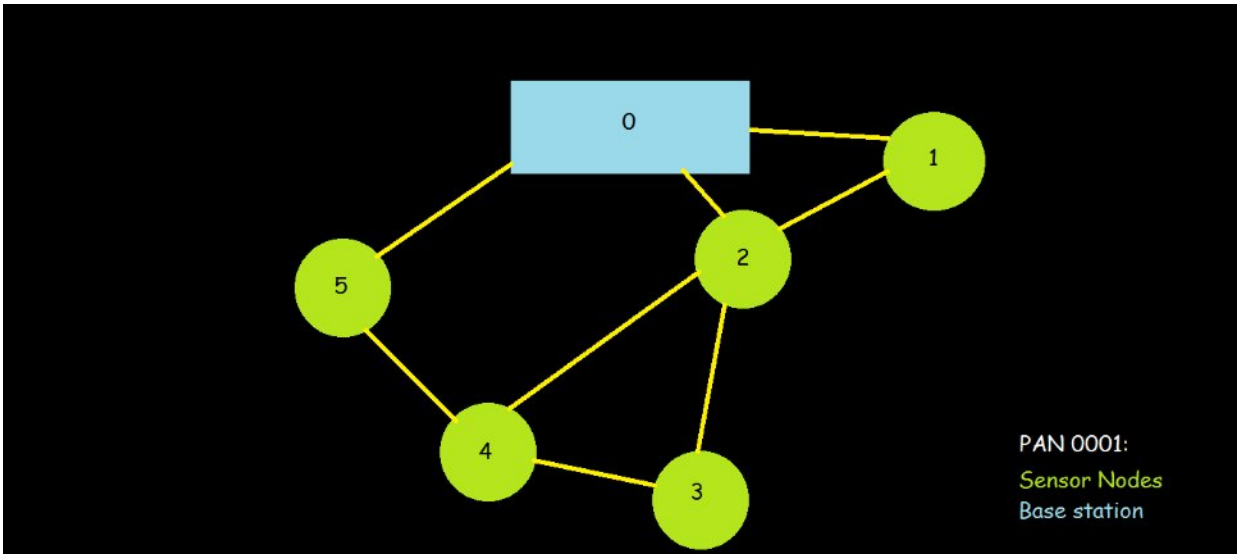
   - New Node Addition: If a new node is added to the network, the node should be identified by the relevant members of the network.

Currently, we have developed algorithms that implement dynamic routing, packet collision avoidance and node failure.

DESIGN

GENERAL DESIGN OF WIRELESS SENSOR NETWORKS:

Diagram 1 depicts a random network topology (incidentally, mesh). The sensor nodes are the data sources. The base station is the data sink. The nodes are able to receive and transmit data. The sensors are connected in a centralized network. Each sensor node also acts as a router that forwards or directs data packets that it receives from other nodes.



(Diagram 1)

OPTIMIZATION PARAMETERS:

Consider NODE 3 from diagram 1. It is clear from the diagram that NODE 3 is not within the radio range of the base station and that it has only two neighbours NODE 4 and NODE 2. So, if it should send its data to the base station, it must send it through NODE 4 or NODE 2. Will NODE 3 send its data through 4 or through 2? On the basis of what criteria will NODE 3 choose the node to which it will forward its data?

The above observation forms the basis to understanding the various parameters of optimisation. Put simply, the answer to the question “On the basis of what criteria will NODE 3 choose the node to which it will forward its data?” is referred to as the “optimization parameter”. Measuring and quantifying this optimisation parameter to assign it to the routes or nodes is essential to the working of sensor network. When quantified and assigned to

routes/nodes, this value is called the “cost” or “weight” associated with the respective path/node.

Some common optimisation parameters include – RSSI (Received Signal Strength Indicator), LQI (Link Quality Index), Hop count to destination and Node battery levels. If the optimisation parameter is the node battery levels, routing is referred to as “Energy Aware Routing”.

It is probably apt to mention that there needn’t be any optimisation parameter in the network! There exists a preposterous, inefficient and pre-historic method in which each node aimlessly and greedily broadcasts its data packet in the hope that it miraculously reaches the destination; instead of routing it by performing a series of intelligent unicasts. This method is called “flooding”. But this method has received the sincere condemnation (that it deserves) by all the wireless sensor network enthusiasts as it consumes a lot of the precious and scarce battery resource.

DESIGN OF OUR WIRELESS SENSOR NETWORK:

Our wireless sensor network has no specific topology. Routing occurs in three phases: Node discovery, data forwarding, node failure detection and path correction. We have used the hops count as the metric of cost calculation.

IMPLEMENTATION – Methods and materials

RESOURCES:

Hardware resources used:

The hardware components that have been used are described below:

1) The PIC24FJ256GA106 microcontroller.
2) The RF24J40MA RF transceiver.
3) The 11AA02E48- standard EEPROM with EUI-48
4) The MCP9700/9700A thermistor as temperature sensor
5) The FXLS8471Q linear accelerometer
6) PICKit 3 was the debugging device used.

The above components were interfaced with each other and enclosed in a Hammond 1593Q enclosure and was made available to us by the Microsoft Mobile Innovation Lab.

Software resources used:

The firmware for the sensor nodes was written by us using the following software:

1) MPLAB X IDE was used to write the firmware
2) MPLAB X driver software
3) XC16 compiler
4) Putty Serial Monitor was used to read the data that was transferred from the base station to the laptop via UART-USB cable.
5) The existing firmware for the star topology network on the sensor nodes and the base station was made available to us by the Microsoft Mobile Innovation Lab. This enabled us to understand how the PIC microcontroller is to be programmed so that it can interact with the peripherals.

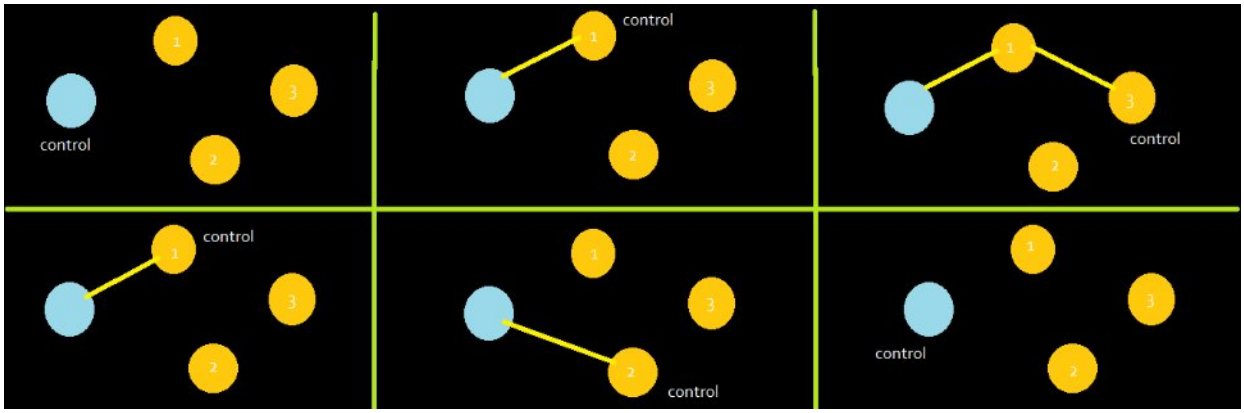
ALGORITHMS DEVELOPED FOR PATH DISCOVERY:

a) The Layered Approach:

In this algorithm, we view the network to consist multiple layers where each layer has nodes that are a particular number of hops away from the base station. The base station is layer 0 and the nodes in its radio range form layer 1. Do note that the word 'control' henceforth means that a node has the right to perform neighbour node discovery and also pass on this right to one of the neighbours.

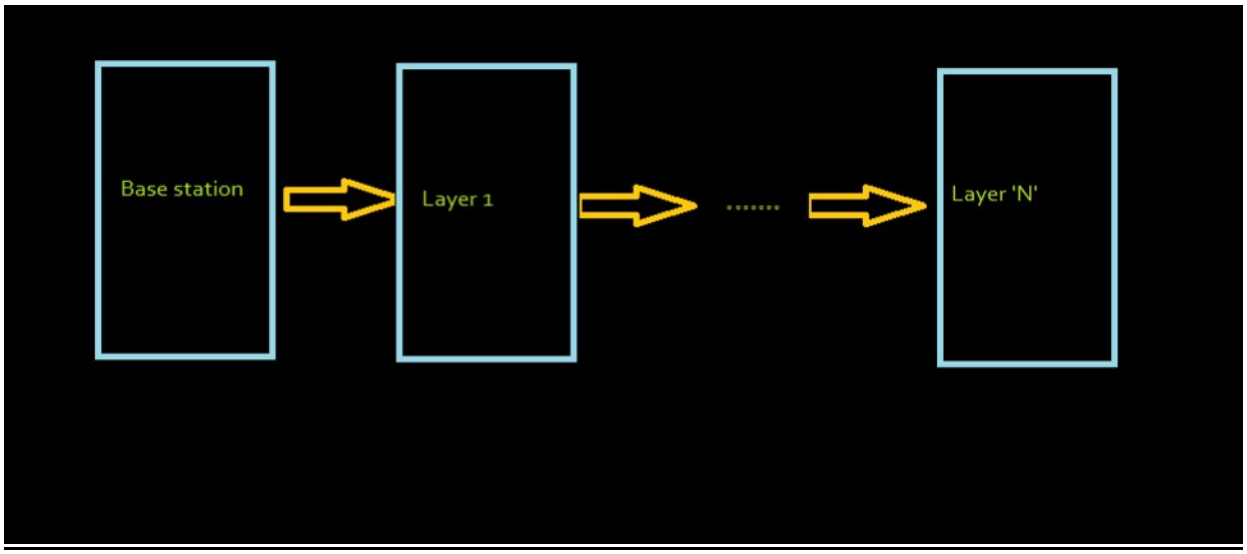
The neighbour discovery begins with the base station in possession of the control and the layer number of all nodes set to 0xFFFF. The base station contains a list of all the nodes in the network and it begins to discover each node on this list with a node discovery packet. Any node that receives from the base station is within its range and hence it sets its layer number to 1 and also responds. This response is in the form of a broadcast which can be heard by all the neighbours of this node. All its neighbours that hear this reply to base check if their layer number is more than the one sent in the reply, if so, they change their layer number to one more than the layer number in the reply message. This ensures all nodes 2 layers above know they belong there. After identifying all its neighbours, the base passes the control to one of them, with a list of all nodes in the network. When a node receives control, it removes itself from the list it received and begins to ping all the nodes on the list and forms its neighbour table with nodes of lowest layer at the top and then passes the control to one of the neighbours. If this neighbour happens to be from the same or lower layer, it rejects the control. This ensures sequential flow of control. A node returns control to the node which gave it control after all of its neighbours have either returned or rejected the control. Once the base receives control back from all its neighbours, the neighbour discovery has concluded and it sends out a packet which is circulated throughout the network indicating this and asking all nodes to begin sending their data. Nodes send sensor data and forward other packets they receive to the node highest on the table and hence most efficient path is ensured.

This algorithm has been illustrated below.



(figure 2)

Figure 3 in the next page shows a block diagram of control flow between the layers.



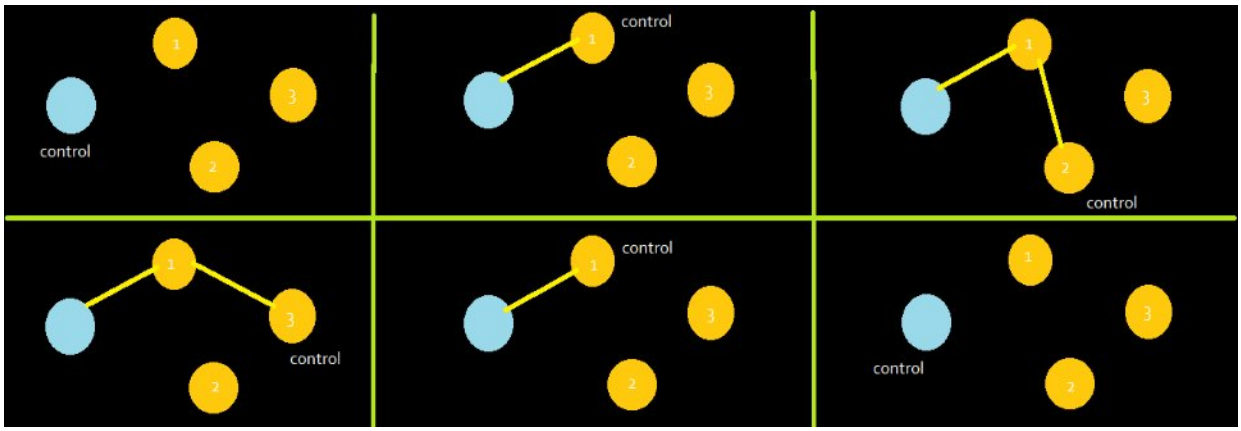
(Figure 3)

b) <u>The Recursive Approach:</u>

In this approach, control is continuously passed from one node to another till the time a maximum number of hops is reached. When a node has control, it initiates the process of discovering its own neighbours. Control is passed to a node as soon as some other node discovers it in its neighbourhood and the node that passed control begins to wait till the control is returned to it. Once control returns back, it continues to discover more nodes; which if discovered, it passes control to it and waits till control is returned. While packets called Node Discovery Packets are forwarded from one node to another in order to pass the control, the previous node gets to know about the return of control by receiving a packet called the Success Packet which on being received sets the wait flag to zero. The main difference between the two approaches; if put in merely one sentence, we could say that

one passes control from one layer to another while the other approach involves control passing from one path to the other.

This algorithm has been represented below in figure 4. The maximum hops are 2 in the figure shown. The nodes shown return the control back as soon as a node receives control which has hoped twice.



(figure 4)

Then, the control is passed to node 2 from the base station and a similar process occurs. When the neighbour tables of the nodes are forwarded to the base station and displayed on the serial monitor, it appears as shown in figure 5.

```
Welcome to PAN 0001
0001 03

0000 01
0002 02
0003 03
---------------
0003 02

0001 02
0003 02
---------------
0002 03

0000 01
0001 02
0003 03
```

(figure 5)

The first field in the above figure indicates the node ID and the second field indicates the number of hops to the base station through the node it is indicated against.

FILES INCLUDED IN THE C CODE

The microcontroller was programmed in the C language. Each project has a set of files that work together to make the sensor nodes and the base work. They are mentioned here:

1) Header files board.h, config.h, fxls8471q.h, ieee802154.h, mrf24j40.h are used to declare important macros pertaining to the respective components interfaced with the microcontroller.
2) main.c file that contains the executed code. It contains all the important methods for node discovery, static/dynamic routing, data transfer, data receive, sleep-wake methods, etc.,

IMPORTANT FUNCTIONS AND METHODS IN THE CODE

Sl no	Function name	Input parameters	Return value	Functionality of method
1	SPI1Init()	void	void	Initialises SPI1 clock and other parameters essential for SPI communication.
2	SPI1SendByte()	Unsigned char	Unsigned char	Sends a byte of data through the SPI1 pin to the RF module.
3	SPI1ReadByte()	Void	Unsigned char	Sends a dummy byte of data through the SPI1 pin to the RF module.
4	BoardInit()	Void	Void	Initialises the microcontroller port lines, timer, interrupt system
5	ADCConvert()	Unsigned char	Unsigned int	It converts the analog inputs to digital inputs using the ADC converter.
6	ConvertADCValueToTemperture()	Unsigned int	Unsigned char	Converts thermistor's voltage values to temperature.
7	SetTimer1()	Unsigned int	void	Sets a timer interrupt after some time depending on the input.
8	FXLS8471Read()	Unsigned char	Unsigned char	SPI function to read from FXLS8471.
9	FXLS8471Write()	Unsigned char, unsigned	Void	SPI function to write into FXLS8471.

		char		
10	FXLSReset()	Void	Void	Resets the accelerometer
11	FXLSInit()	Void	Unsigned char	Configures the accelerometer sensor
12	FXLS8471ReadAccel()	SRAWDATA *	Void	Performs a block read of the status and acceleration data and places the bytes read into the structures of type
13	EUID48Init()	Void	Void	Functions to read EUID48 from the 11AA02E48 IC
14	_T1Interrupt	Void	Void	Timer1 Interrupt
15	_CNInterrupt	Void	Void	Change notification interrupt
16	_RTCCInterrupt	Void	Void	RTCC Interrupt
17	MRF24J40_ReadShortRAMAddr()	Unsigned char	Unsigned char	Reads from RF module's short address memory space[1] register.
18	MRF24J40_WriteShortRAMAddr()	Unsigned char, unsigned char	void	Write into RF module's short address memory space.
19	MRF24J40_ReadLongRAMAddr()	Unsigned int	Unsigned char	Reads from RF module's Long address memory space[2] register.
20	MRF24J40_WriteLongRAMAddr()	Unsigned int, unsigned char	void	Writes into RF module's Long address memory space register.
21	MRF24J40_Init()	Void	Void	Initializes the RF module
22	MRF24J40_Configure_NonBeaconNetwork()	Void	Void	Configures the network to pass no beacons around
23	MRF24J40_GoToSleep()	Void	Void	Puts the RF module to sleep
24	MRF24J40_ExitSleep()	Void	Void	Wakes the RF module from sleep by resetting the PHY and MAC layers.
25	MRF24J40_Tx()	Volatile unsigned char*, unsigned	Void	Has the RF module transmit a message from the microcontroller to other nodes.

[1] Microchip's MRF24J40 datasheet
[2] Microchip's MRF24J40 datasheet

		char		
26	MRF24J40_SetChannel()	unsigned char	Void	Sets the RF channel which is used for data transmission.
27	PHYInit	Void	Void	Initializes the Physical layer.
28	MACInit	Void	Void	Initializes the MAC layer
29	MCPS_DATA_REQUEST()	Unsigned char, unsigned int, unsigned char, unsigned char, unsigned int, unsigned char, unsigned char, unsigned char*, unsigned char, unsigned char	Unsigned char	Takes all the input parameters to make a data packet for transmission. Adds the MAC header to and the MSDU. Then, it transmits this packet.
30	AcquireData()	Void	Void	Acquires the data from the surroundings and stores it
31	FormPeriodicDataPacket()	Void	Void	Forms a data packet when ever called. Responsible for making the MSDU part of the data packet.
32	FormNodeDiscoveryPacket()	Void	Void	Forms the MSDU part of the node discovery packet
33	FormSuccessPacket()	Void	Void	Forms the MSDU part of the node discovery packet
34	receivePacket()	Void	Void	When a physical packet is received, the Interrupt Service Routing calls on this function to receive the packet from the receive buffer of the RF module.

35	NeighbourUpdate()	Int	struct Ninfo	Updates the information of the neighbour it receives through a Node discovery packet.
36	deleteEntry()	Int	Unsigned char	Deletes the structure variable in it's array at the position indicated by the parameter.
37	CheckNeighbourtable()	Void	Struct Ninfo	Checks for redundancies in the table before updating it and updates if necessary.

There are more functions in the different codes and the ones listed above are only the most important ones and the ones that have been called in some part of the code.

RESULTS, DISCUSSIONS AND CONCULSIONS

RESULTS AND CONCLUSIONS:

A dynamically routing network was formed in accordance to the algorithms described. Node failure was gracefully handled. Failure of few nodes didn't disturb the entire network.

POSSIBLE FUTURE ENHANCEMENTS:

1) Time synchronisation to save batteries is an essential feature which must be added in order to preserve the scarce energy resources available to the sensor nodes. The RF module must undergo alternate sleep-awake cycles.
2) The algorithms can be optimised further.
3) The time and space complexities of each of the algorithms can be computed mathematically and compared for future reference.
4) New node addition is a necessary feature of a truly dynamic network. This must be added.

APPENDIX

All the codes and implementations of the different algorithms can be found on this link:

Dynamic Routing In WSNs - MMIL