International Journal of Electronics and Communication Engineering (IJECE)
ISSN(P):2278-9901; ISSN(E): 2278-991X;
Vol. 5, Issue 1, Dec - Jan 2015, 1-18
© IASET

International Academy of Science, Engineering and Technology
IASET Connecting Researchers; Nurturing Innovations

EFFICIENT ROUTING PROTOCOL FOR MAINTAINING BETTER ENERGY MANAGEMENT IN SENSOR NETWORKS

P. SATHYARAJ[1], S. RUKMANI DEVI[2] & D. KALPANA[3]

[1]Assistant Professor, RMK College of Engineering and Technology, Kavaraipettai, Tamil Nadu, India

[2]Professor, RMD Engineering College, Kavaraipettai, Tamil Nadu, India

[3]Assistant Professor, RMD Engineering College, Kavaraipettai, Tamil Nadu, India

ABSTRACT

Underwater wireless sensor networks (UWSN) similar to the terrestrial sensor networks have different challenges such as limited bandwidth, low battery power, defective underwater Channels and high variable propagation delay. A crucial problem in UWSN is finding an efficient route between a source and a sink. Consequently, great efforts have been made for Designing efficient protocols while considering the unique characteristics of underwater Communication. Several routing protocols are proposed for this issue and can be classified into geographic and non-geographic routing protocols. In this paper we focus on the cluster Head routing protocols to find the minimum distance. We introduce a review and comparison of different algorithms proposed recently in the literature. We are also presented a novel Taxonomy of these routing in which the protocols are classified into two categories (intra Cluster and inter cluster) according to their forwarding strategies.

KEYWORDS: Efficient Routing Protocol for Maintaining Better Energy Management in Sensor Networks

INTRODUCTION

A wireless Sensor Network (WSN) consists of hundreds or thousands of sensor nodes and a small number of data collection devices. The sensor nodes have the form of low-cost, low-power, small-size devices, and are designed to carry out a range of

Sensing applications, including environmental monitoring, military surveillance, fire detection, animal tracking, and so on. The sensor nodes gather the information of interest locally and then forward the sensed information over a wireless medium to a remote data collection device (sink), where it is fused and analyzed in order to determine the global status of the sensed area.

In many WSN applications, the sensor nodes are required to know their locations with a high degree of precision, such as tracking of goods, forest fire detection, and etc. For example, in forest fire tracking, the moving perimeter of the fire can only be traced if the locations of the sensors are accurately known. Accordingly, many sensor localization methods have been proposed for WSNs. Broadly speaking, these methods can be categorized as either range-based or range-free. In range-based schemes, the sensor locations are calculated from the node-to-node distances or inter-node angles. Conversely, in range-free schemes, the sensor locations are determined by radio connectivity constraint. Range based schemes are typically more accurate than range-free schemes. However, they require the use of infrared, X-ray or ultrasound techniques to calculate the inter-node distance and/or angle.

A key feature of such networks is that each network consists of a large number of un tethered and unattended sensor nodes. These nodes often have very limited and non-replenishable energy resources, which makes energy an important design issue for these networks. Routing is another very challenging design issue for WSNs. A properly designed routing protocol should not only ensure high message delivery ratio and low energy consumption for message delivery, but also balance the entire sensor network energy consumption, and thereby extend the sensor network lifetime.

Motivated by the fact that WSNs routing is often geography based, we propose a geography-based secure and efficient Resource Conscious Secure routing (RCS) protocol for WSNs without relying on flooding. RCS allows messages to be transmitted using two routing strategies, random walking and deterministic routing, in the same framework. The distribution of these two strategies is determined by the specific security requirements. This scenario is analogous to delivering US Mail through USPS: express mails cost more than regular mails; however, mails can be delivered faster. The protocol also provides a secure message delivery option to maximize the message delivery ratio under adversarial attacks. In addition, we also give quantitative secure analysis on the proposed routing protocol based on the criteria proposed.

LITERATURE REVIEW

Network Partition

The network is evenly divided into small grids. Each grid has a relative location based on the grid information. The node in each grid with the highest energy level is selected as the head node or message forwarding. In addition, each node in the grid will maintain its own attributes, including location information, remaining energy level of its grid, as well as the attributes of its adjacent neighboring grids. The information maintained by each sensor node will be updated periodically. We assume that the sensor nodes in its direct neighboring grids are all within its direct communication range. We also assume that the whole network is fully connected through multi-hop communications. In addition, through the maintained energy levels of its adjacent neighboring grids, it can be used to detect and filter out the compromised nodes for active routing selection.

Shortest Path Routing

The shortest path routing also called deterministic routing, in this routing the next hop grid is selected from the neighbor grid list based on the relative locations of the grid. The grid that is closest to the sink node is selected for message forwarding and also we are considered energy level of the selected node. The selected nodes have the highest energy level when compared with other node’s energy levels. In this routing we are using cryptographic technique for message security. The deterministic shortest path routing guarantees that the messages are sent from the source node to the sink node.

Secure Message Forwarding

This routing is also called random walking, in this routing the next hop grid randomly selected from neighbor grid list for message forwarding. The routing path becomes more dynamic and unpredictable. In this way, it is more difficult for the adversary to capture the message or to jam the traffic. Therefore, the delivery ratio can be increased in a hostile environment. Using this routing we can avoid the jamming.

Message Encryption & Decryption

In this module we are generating a key for secure message forwarding. Using this secret key we can encrypt and decrypt the message. Encryption is the conversion of electronic data into another form, called ciphertext, which cannot be easily understood by anyone except authorized parties. Encryption is the process of encoding messages or information in such a way that only authorized parties can read it.Encryption does not of itself prevent interception, but denies the message content to the interceptor. In an encryption scheme, the message or information, referred to as plaintext, is encrypted using an encryption algorithm, generating cipher text that can only be read if decrypted. The reverse process of encryption is called decryption. The decryption is process of converting a cipher text message to plaintext message.

System Specification

Hardware Requirements

System	**:** Pentium IV 2.4 GHz.
Hard Disk	**:** 40 GB.
Floppy Drive	**:** 1.44 Mb.
Monitor	**:** 15 VGA Colour.
Mouse	**:** Logitech.
Ram	**:** 512 Mb

Software Requirements

Operating System	**:** Ubuntu12.04
Simulation Tool	**:** NS 2.35
Font End	**:** OTCL
Back End	**:** C++

Highlights of Different Methodology

Table 1: Different Methodologies Used

S. No	Title of the Paper & Year	Pros	Cons
1	Geography informed Energy Conservation for Ad Hoc Routing & 2001	Reduce the Energy consumption	They not considered about data security
2	Quantitative Measurement and Design of Source-Location Privacy Schemes for Wireless Sensor Networks & 2012	Provide the secure for Source location	They are not considered energy and provide the secure for only the source location not for destination
3	Providing Hop-by-Hop Authentication and Source Privacy in Wireless Sensor Networks & 2012	Provide security for hop by hop and also provide the source privacy	They are not considered about energy and routing strategies

Existing System

In Geographic and energy aware routing (GEAR), the sink node disseminates requests with geographic attributes to the target region instead of using flooding. Each node forwards messages to its neighboring nodes based on estimated cost and learning cost. Source-location privacy is provided through broadcasting that mixes valid messages with dummy messages. The transmission of dummy messages not only consumes significant amount of sensor energy, but also increases the network collisions and decreases the packet delivery ratio. In phantom routing protocol, each message is routed from the actual source to a phantom source along a designed directed walk through either sector based approach or hop-based approach. The direction/sector information is stored in the header of the message. In this way, the phantom source can be away from the actual source. Unfortunately, once the message is captured on the random walk path, the adversaries are able to get the direction/sector information stored in the header of the message.

Disadvantages

- More energy consumption
- Increase the network collision
- Reduce the packet delivery ratio
- Cannot provide the full secure for packets

Proposed System

We propose a secure and efficient Resource Conscious Routing (RCS) protocol that can address energy balance and routing security concurrently in WSNs. In RCS routing protocol, each sensor node needs to maintain the energy levels of its immediate adjacent neighboring grids in addition to their relative locations. Using this information, each sensor node can create varying filters based on the expected design tradeoff between security and efficiency. The quantitative security analysis demonstrates the proposed algorithm can protect the source location information from the adversaries. In this project, we will focus on two routing strategies for message forwarding: shortest path message forwarding, and secure message forwarding through random walking to create routing path unpredictability for source privacy and jamming prevention.

Advantages

- Reduce the energy consumption
- Provide the more secure for packet and also routing
- Increase the message delivery ratio
- Reduce the time delay

Problem Statement

In existing work, source-location privacy is provided through broad casting that mixes valid messages with dummy messages. The transmission of dummy messages not only consumes significant amount of sensor energy, but also increases the network collisions and decreases the packet delivery ratio quantitatively measure source-location information leakage for routing-based schemes through source location disclosure index (SDI) and source-location space index (SSI). To the best of our knowledge, none of these schemes have considered privacy from a cost-aware perspective. So, in this project we propose a resource conscious routing (RCS) protocol. Now we overcome the collision and packet delivery problem from the existing system used by the RCS routing protocol

Block Diagram

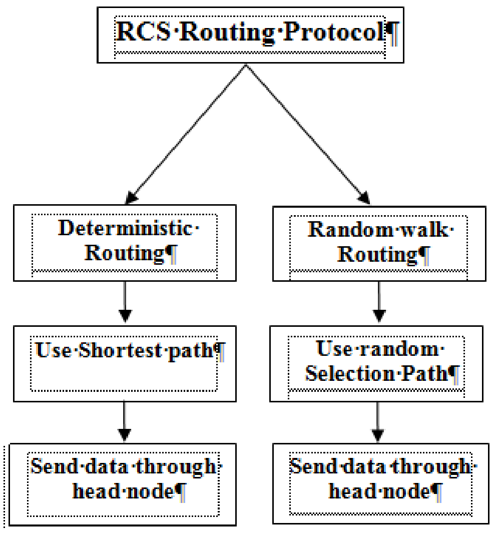

RCS Routing Protocol
Deterministic Routing
Random walk Routing
Use Shortest path
Use random Selection Path
Send data through head node
Send data through head node

Figure 1: Classification of RCS Routing Protocol

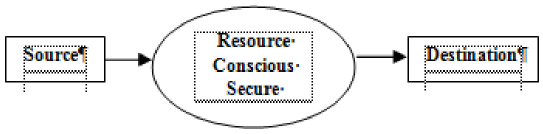


Figure 2: Level 0

Flow Chart

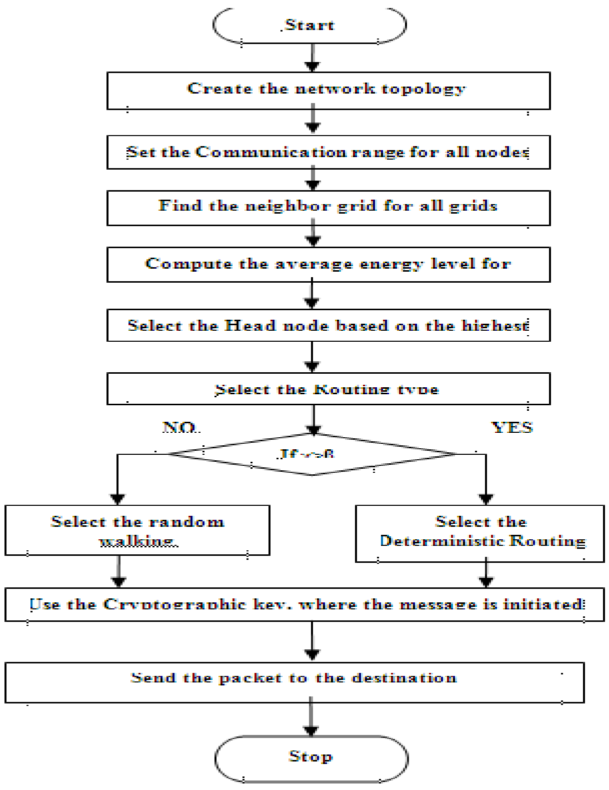

Start
Create the network topology
Set the Communication range for all nodes
Find the neighbor grid for all grids
Compute the average energy level for
Select the Head node based on the highest
Select the Routing type
NO
YES
If r>=R
Select the random walking
Select the Deterministic Routing
Use the Cryptographic key, where the message is initiated
Send the packet to the destination
Stop

Figure 3: Flow Chart for the Router

System Architecture

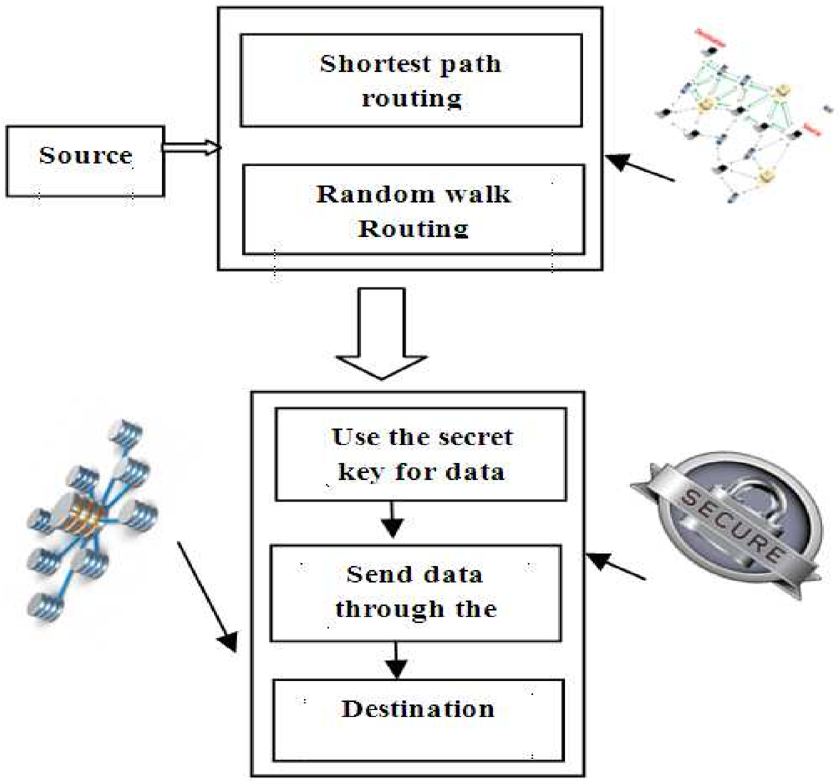


Figure 4: System Architecture Diagram

Wireless Sensor Network

The concept of wireless sensor networks is based on a simple equation:

Sensing + CPU + Radio = Thousands of potential applications

As soon as people understand the capabilities of a wireless sensor network, hundreds of applications spring to mind. It seems like a straightforward combination of modern technology. However, actually combining sensors, radios, and CPU's into an effective wireless sensor network requires a detailed understanding of the both capabilities and limitations of each of the underlying hardware components, as well as a detailed understanding of modern networking technologies and distributed systems theory. Each individual node must be designed to provide the set of primitives necessary to synthesize the interconnected web that will emerge as they are deployed, while meeting strict requirements of size, cost and power consumption. A core challenge is to map the overall system requirements down to individual device capabilities, requirements and actions. To make the wireless sensor network vision a reality, architecture must be developed that synthesizes the envisioned applications out of the underlying hardware capabilities. To develop this system architecture we work from the high level application requirements down through the low-level hardware requirements. In this process we first attempt to understand the set of target applications. To limit the number of applications that we must consider, we focus on a set of application classes that we believe are representative of a large fraction of the potential usage scenarios. We use this set of application classes to explore the system-level requirements that are placed on the overall architecture. From these system-level requirements we can then drill down into the individual node-level requirements.

Application

The three application classes are: environmental data collection, security monitoring, and sensor node tracking. We believe that the majority of wireless sensor network deployments will fall into one of these class templates.

Environmental Data Collection

A canonical environmental data collection application is one where a research scientist wants to collect several sensor readings from a set of points in an environment over a period of time in order to detect trends and interdependencies. This scientist would want to collect data from hundreds of points spread throughout the area and then analyze the data offline. The scientist would be interested in collecting data over several months or years in order to look for long-term and seasonal trends. For the data to be meaningful it would have to be collected at regular intervals and the nodes would remain at known locations.

At the network level, the environmental data collection application is characterized by having a large number of nodes continually sensing and transmitting data back to a set of base stations that store the data using traditional methods. These networks generally require very low data rates and extremely long lifetimes. In typical usage scenario, the nodes will be evenly distributed over an outdoor environment. This distance between adjacent nodes will be minimal yet the distance across the entire network will be significant.

After deployment, the nodes must first discover the topology of the network and estimate optimal routing strategies. The routing strategy can then be used to route data to a central collection points. In environmental monitoring applications, it is not essential that the nodes develop the optimal routing strategies on their own. Instead, it may be possible to calculate the optimal routing topology outside of the network and then communicate the necessary information to the nodes as required. This is possible because the physical topology of the network is relatively constant. While the time variant nature of RF communication may cause connectivity between two nodes to be intermittent, the overall topology of the network will be relatively stable.

Environmental data collection applications typically use tree-based routing topologies where each routing tree is rooted at high-capability nodes that sink data. Data is periodically transmitted from child node to parent node up the tree-structure until it reaches the sink. With tree-based data collection each node is responsible for forwarding the data of all its descendants. Nodes with a large number of descendants transmit significantly more data than leaf nodes. These nodes can quickly become energy bottlenecks. Once the network is configured, each node periodically samples its sensors and transmits its data up the routing tree and back to the base station. For many scenarios, the interval between these transmissions can be on the order of minutes. Typical reporting periods are expected to be between 1 and 15 minutes; while it is possible for networks to have significantly higher reporting rates. The typical environment parameters being monitored, such as temperature, light intensity, and humidity, does not change quickly enough to require higher reporting rates.

In addition to large sample intervals, environmental monitoring applications do not have strict latency requirements. Data samples can be delayed inside the network for moderate periods of time without significantly affecting application performance. In general the data is collected for future analysis, not for real-time operation.

In order to meet lifetime requirements, each communication event must be precisely scheduled. The senor nodes will remain dormant a majority of the time; they will only wake to transmit or receive data. If the precise schedule is not met, the communication events will fail. As the network ages, it is expected that nodes will fail over time. Periodically the network will have to reconfigure to handle node/link failure or to redistribute network load. Additionally, as the researchers learn more about the environment they study, they may want to go in and insert additional sensing points. In both cases, the reconfigurations are relatively infrequent and will not represent a significant amount of the overall system energy usage.

The most important characteristics of the environmental monitoring requirements are long lifetime, precise synchronization, low data rates and relatively static topologies. Additionally it is not essential that the data be transmitted in real-time back to the central collection point. The data transmissions can be delayed inside the network as necessary in order to improve network efficiency.

Security Monitoring

Our second class of sensor network application is security monitoring. Security monitoring networks are composed of nodes that are placed at fixed locations throughout an environment that continually monitor one or more sensors to detect an anomaly. A key difference between security monitoring and environmental monitoring is that security networks are not actually collecting any data. This has a significant impact on the optimal network architecture. Each node has to frequently check the status of its sensors but it only has to transmit a data report when there is a security violation.

The immediate and reliable communication of alarm messages is the primary system requirement. These are “report by exception” networks. Additionally, it is essential that it is confirmed that each node is still present and functioning. If a node were to be disabled or fail, it would represent a security violation that should be reported. For security monitoring applications, the network must be configured so that nodes are responsible for confirming the status of each other. One approach is to have each node be assigned to peer that will report if a node is not functioning. The optimal topology of a security monitoring network will look quite different from that of a data collection network. In a collection tree, each node must transmit the data of all of its decedents. Because of this, it is optimal to have a short, wide tree. In contrast, with a security network the optimal configuration would be to have a linear topology that forms a Hamiltonian cycle of the network. The power consumption of each node is only proportional to the number of children it has.

In a linear network, each node would have only one child. This would evenly distribute the energy consumption of the network. The accepted norm for security systems today is that each sensor should be checked approximately once per hour. Combined with the ability to evenly distribute the load of checking nodes, the energy cost of performing this check becomes minimal. A majority of the energy consumption in a security network is spent on meeting the strict latency requirements associated with the signaling the alarm when a security violation occurs.

Once detected, a security violation must be communicated to the base station immediately. The latency of the data communication across the network to the base station has a critical impact on application performance. Users demand that alarm situations be reported within seconds of detection. This means that network nodes must be able to respond quickly to requests from their neighbors to forward data. In security networks reducing the latency of an alarm transmission is significantly more important than reducing the energy cost of the transmissions. This is because alarm events are expected to be rare. In a fire security system alarms would almost never be signaled. In the event that one does occur a significant amount of energy could be dedicated to the transmission. Reducing the transmission latency leads to higher energy consumption because routing nodes must monitor the radio channel more frequently.

In security networks, a vast majority of the energy will be spend on confirming the functionality of neighboring nodes and in being prepared to instantly forward alarm announcements. Actual data transmission will consume a small fraction of the network energy.

Lifetime

Critical to any wireless sensor network deployment is the expected lifetime. The goal of both the environmental monitoring and security application scenarios is to have nodes placed out in the field, unattended, for months or years. The primary limiting factor for the lifetime of a sensor network is the energy supply. Each node must be designed to manage its local supply of energy in order to maximize total network lifetime. In many deployments it is not the average node lifetime that is important, but rather the minimum node lifetime. In the case of wireless security systems, every node must last for multiple years.

A single node failure would create vulnerability in the security systems. In some situations it may be possible to exploit external power, perhaps by tapping into building power with some or all nodes. However, one of the major benefits to wireless systems is the ease of installation. Requiring power to be supplied externally to all nodes largely negates this advantage. A compromise is to have a handful of special nodes that are wired into the building’s power infrastructure. In most application scenarios, a majority of the nodes will have to be self powered. They will either have to contain enough stored energy to last for years, or they will have to be able to scavenge energy from the environment through devices, such as solar cells or piezoelectric generators. Both of these options demand that that the average energy consumption of the nodes be as low as possible. The most significant factor in determining lifetime of a given energy supply is radio power consumption.

In a wireless sensor node the radio consumes a vast majority of the system energy. This power consumption can be reduced through decreasing the transmission output power or through decreasing the radio duty cycle. Both of these alternatives involve sacrificing other system metrics.

Localization in Sensor Networks

Advances in technology have made it possible to build ad hoc sensor networks using inexpensive nodes consisting of a low power processor, a modest amount of memory, a wireless network transceiver and a sensor board; a typical node is comparable in size to 2 AA batteries. Many novel applications are emerging: habitat monitoring, smart building failure detection and reporting, and target tracking. In these applications it is necessary to accurately orient the nodes with respect to a global coordinate system in order to report data that is geographically meaningful. Furthermore, basic middle ware services such as routing often rely on location information (e.g., geographic routing). Ad hoc sensor networks present novel tradeoffs in system design. On the one hand, the low cost of the nodes facilitates massive scale and highly parallel Computation.

On the other hand, each node is likely to have limited power, limited reliability, and only local communication with a modest number of neighbors. These application contexts and potential massive scale make it unrealistic to rely on careful placement or uniform arrangement of sensors. Rather than use globally accessible beacons or expensive GPS to localize each sensor, we would like the sensors to self-organize a coordinate system.

No specific algorithm is a clear favorite across the spectrum. For example, some algorithms rely on prepositioned nodes while others are able to do without. Other algorithms require expensive hardware capabilities. Some algorithms need away of performing off-line computation, while other algorithms are able to do all their calculations on the sensor nodes themselves. Localization is still an new and exciting field, with new algorithms, hardware, and applications being developed at a feverish pace; it is hard to say what techniques and hardware will be prevalent in the end.

Localization Hardware

The localization problem gives rise to two important hardware problems. The first problem of defining a coordinate system. The second which is the more technically challenging, is the problem of calculating the distance between sensors (the ranging problem).

Anchor/Beacon Nodes

The goal of localization is to determine the physical coordinates of a group of sensor nodes. These coordinates can be global, meaning they are aligned with some externally meaningful system like GPS, or relative, meaning that they are an arbitrary “rigid transformation” (rotation, reflection, translation) away from the global coordinate system.

Beacon nodes (also frequently called anchor nodes) are a necessary prerequisite to localizing a network in a global coordinate system. Beacon nodes are simply ordinary sensor nodes that know their global coordinates a priori. This knowledge could be hard coded, or acquired through some additional hardware like a GPS receiver. At a minimum, three non-collinear beacon nodes are required to define a global coordinate system in two dimensions. If three dimensional coordinates are required, there at least four non-coplanar beacons must be present. Beacon nodes can be used in several ways. Some algorithms (e.g. MDSMAP,) localize nodes in an arbitrary relative coordinate system, and then use a few beacon nodes to determine a rigid transformation of the relative coordinates into global coordinates (see appendix B). Other algorithms (e.g. APIT) use beacons throughout, using the positions of several beacons to “bootstrap” the global positions of non-beacon nodes. Beacon placement can often have a significant impact on localization. Many groups have found that localization accuracy improves if beacons are placed in a convex hull around the network. Locating additional beacons in the center of the network is also helpful. In any event, there is considerable evidence that real improvements in localization can be obtained by planning beacon layout in the network.

The advantage of using beacons is obvious: the presence of several pre localized nodes can greatly simplify the task of assigning coordinates to ordinary nodes. However, beacon nodes have inherent disadvantages. GPS receivers are expensive. They also cannot typically be used indoors, and can also be confused by tall buildings or other environmental obstacles. GPS receivers also consume significant battery power, which can be a problem for power-constrained sensor nodes. The alternative to GPS is pre-programming nodes with their locations, which can be impractical (for instance when deploying 10,000 nodes with 500 beacons) or even impossible (for instance when deploying nodes from an aircraft).

In short, beacons are necessary for localization, but their use does not come without cost. It will focus on hardware methods of computing distance measurements between nearby sensor nodes (i.e. ranging).

Received Signal Strength Indication (RSSI)

In wireless sensor networks, every sensor has a radio. The question is: how can the radio help localize the network? There are two important techniques for using radio information to compute ranges. The other, Received Signal

Strength Indication (RSSI), is covered below.

In theory, the energy of a radio signal diminishes with the square of the distance from the signal’s source. As a result, a node listening to a radio transmission should be able to use the strength of the received signal to calculate its distance from the transmitter. RSSI suggests an elegant solution to the hardware ranging problem: all sensor nodes are likely to have radios – why not use them to compute ranges for localization? In practice, however, RSSI ranging measurements contain noise on the order of several meters. This noise occurs because radio propagation tends to be highly non-uniform in real environments. For instance, radio propagates differently over asphalt than over grass. Physical obstacles such as walls, furniture, etc. reflect and absorb radio waves. As a result, distance predictions using signal strength have been unable to demonstrate the precision obtained by other ranging methods such as time difference of arrival.

However, RSSI has garnered new interest recently. More careful physical analysis of radio propagation may allow better use of RSSI data, as might better calibration of sensor radios. Whitehouse did an extensive analysis of radio signal strength, which he was able to parlay into noticeable improvements in localization.

SOFTWARE CONFIGURED

Network Simulator

Simulation is a very important modern technology. It can be applied to different science, engineering, or other application fields for different purposes. Computer assisted simulation can model hypothetical and real-life objects or activities on a computer so that it can be studied to see how the system function. Different variables can be used to predict the behavior of the system. Computer simulation can be used to assist the modeling and analysis in many natural systems. Typical application areas include physics, chemistry, biology, and human-involved systems in economics, finance or even social science. Other important applications are in the engineering such as civil engineering, structural engineering, mechanical engineering, and computer engineering. Application of simulation technology into networking area such as network traffic simulation, however, is relatively new.

For network simulation, more specifically, it means that the computer assisted simulation technologies are being applied in the simulation of networking algorithms or systems by using software engineering. The application field is narrower than general simulation and it is natural that more specific requirements will be placed on network simulations. For example, the network simulations may put more emphasis on the performance or validity of a distributed protocol or algorithm rather than the visual or real-time visibility features of the simulations. Moreover, since network technologies is keeping developing very fast and so many different organizations participate in the whole process and they have different technologies or products running on different software on the Internet. That is why the network simulations always require open platforms which should be scalable enough to include different efforts and different packages in the simulations of the whole network. Internet has also a characteristic that it is structured with a uniformed network stack (TCP/IP) that all the different layers technologies can be implemented differently but with a uniformed interface with their neighbored layers. Thus the network simulation tools have to be able to incorporate this feature and allow different future new packages to be included and run transparently without harming existing components or packages. Thus the negative impact of some packages will have no or little impact to the other modules or packages.

Network simulators are used by people from different areas such as academic researchers, industrial developers, and Quality Assurance (QA) to design, simulate, verify, and analyze the performance of different networks protocols. They can also be used to evaluate the effect of the different parameters on the protocols being studied. Generally a network simulator will comprise of a wide range of networking technologies and protocols and help users to build complex networks from basic building blocks like clusters of nodes and links. With their help, one can design different network topologies using various types of nodes such as end-hosts, hubs, network bridges, routers, optical link-layer devices, and mobile units.

Simulation and Emulation

In the research area of computer and communications networks, simulation is a useful technique since the behavior of a network can be modeled by calculating the interaction between the different network components (they can be end-host or network entities such as routers, physical links or packets) using mathematical formulas. They can also be modeled by actually or virtually capturing and playing back experimental observations from a real production networks. After we get the observation data from simulation experiments, the behavior of the network and protocols supported can then be observed and analyzed in a series of offline test experiments. All kinds of environmental attributes can also be modified in a controlled manner to assess how the network can behave under different parameters combinations or different configuration conditions. Another characteristic of network simulation that worth noticing is that the simulation program can be used together with different applications and services in order to observe end-to-end or other point-to-point performance in the networks.

Network emulation, however, means that network under planning is simulated in order to assess its performance or to predict the impact of possible changes, or optimizations. The major difference lying between them is that a network emulator means that end-systems such as computers can be attached to the emulator and will act exactly as they are attached to a real network. The major point is that the network emulator's job is to emulate the network which connects end-hosts, but not the end-hosts themselves. Typical network emulation tools include NS2 which is a popular network simulator that can also be used as a limited-functionality emulator. In contrast, a typical network emulator such as WAN Sim is a simple bridged WAN emulator that utilizes some Linux functionality.

Type of Network Simulators

Different types of network simulators can be categorized and explained based on some criteria such as if they are commercial or free, or if they are simple ones or complex ones.

Commercial and Open Source Simulators

Some of the network simulators are commercial which means that they would not provide the source code of its software or the affiliated packages to the general users for free. All the user s have to pay to get the license to use their software or pay to order specific packages for their own specific usage requirements. One typical example is the OPNET. Commercial simulator has its advantage and disadvantage. The advantage is that it generally has complete and up-to-date documentations and they can be consistently maintained by some specialized staff in that company. However, the open source network simulator is disadvantageous in this aspect, and generally there are not enough specialized people working on the documentation. This problem can be serious when the different versions come with many new things and it will become difficult to trace or understand the previous codes without appropriate documentations.

On the contrary, the open source network simulator has the advantage that everything is very open and everyone or organization can contribute to it and find bugs in it. The interface is also open for future improvement. It can also be very flexible and reflect the most new recent developments of new technologies in a faster way than commercial network simulators. We can see that some advantages of commercial network simulators, however, are the disadvantage for the open source network simulators. Lack of enough systematic and complete documentations and lack of version control supports can lead to some serious problem s and can limit the applicability and life-time of the open source network simulators. Typical open source network simulators include NS2, NS3. We will introduce and analyze them in great detail in the following sections.

Table 2: Network Simulators

Type	Network Simulators Name
Commercial	OPNET, QualNet
Open source	NS2, NS3, OMNeT++, SSFNet, J-Sim

Network Simulator 2 (NS2)

NS2 is one of the most popular open source network simulators. The original NS is a discrete event simulator targeted at networking research. In this section, we will give a brief introduction to the NS2 system.

NS2 Simulation Concept

NS2 simulation consists of two major phases.

Phase I: Network Configuration Phase

In this phase, NS2 constructs a network and sets up an initial chain of events. The initial chain of events consists of events which are scheduled to occur at certain times (e.g., starts FTP (File Transfer Protocol) traffic at 1 second.).These events are called at-events. This phase corresponds to every line in a Tcl simulation script before executing instproc run{} of the Simulator object.

Phase II: Simulation Phase

This part corresponds to a single line, which invokes instproc Simulator::run {}. Ironically, this single line contributes to most (e.g., 99%) of the simulation. In this part, NS2 moves along the chain of events and executes each event chronologically. Here, the instproc Simulator::run{} starts the simulation by dispatching the first event in the chain of events. In NS2, “dispatching an event” or “firing an event” means “taking actions corresponding to that event”. An action is, for example, starting FTP traffic or creating another event and inserting the created event into the chain of events. In Fig. 1, at 0.9 s, Event1 creates Event5 which will be dispatched at 3.7 s, and inserts Event5 after Event2. After dispatching an event, NS2 moves down the chain and dispatches the next event. This process repeats until the last event corresponding to instprochalt{} of OTCL class Simulator is dispatched, signifying the end of simulation.

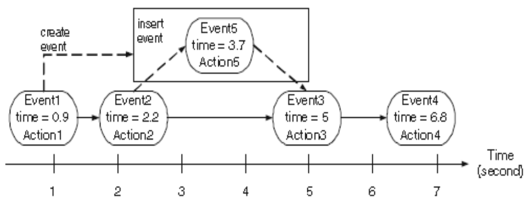

create event
insert event
Event5 time = 3.7 Action5
Event1 time = 0.9 Action1
Event2 time = 2.2 Action2
Event3 time = 5 Action3
Event4 time = 6.8 Action4
Time (second)
1
2
3
4
5
6
7

Figure 5: sample Chain of Events in a Discrete-Event Simulation

Tcl Programming Language

Tcl (pronounced Tickle) is a general purpose programming language originally intended to be embedded in other applications as a configuration and extension language. The success of one its most important embeddings, the Tk toolkit for the X Windows System, has resulted in Tcl and Tk together being most heavily used for building graphical user interfaces (GUIs). It is also heavily used as a scripting language like Awk, Perl or Rexx, and (as Expect) is used to script interactive applications (e.g., to automate telnet logins to various information vendors).

Here in Library Systems at The University of Chicago we already use Tcl:

- As a general-purpose scripting language to write UNIX applications.
- To automate login scripts for telnet connections.
- As the programming language for the Reserve System.

In addition, a number of our UNIX applications are written in Tcl (see below), and Tcl is the programming language of the University of Chicago BSDAC Phoenix Project.

Tcl is a Successful Language

The Tcl/Tk developer community now numbers in the tens of thousands and there are thousands of Tcl applications in existence or under development. The application areas for Tcl and Tk cover virtually the entire spectrum of graphical and engineering applications, including computer-aided design, software development, testing, instrument control, scientific visualization, and multimedia. Tcl and Tk are being used by hundreds of companies, large and small, as well as universities and research laboratories.

The companies using Tcl include SCO, Digital, Cray, AT&T, and Sun Microsystems. John Ousterhout, the designer of Tcl/Tk, has recently moved to Sun Microsystems from UC Berkeley to head up a group dedicated to Tcl/Tk; he has made some comments on Tcl's future at Sun. There is an annual international Tcl conference which has been held twice so far, and a safe variant of Tcl has been proposed as the language for Enabled Mail in the Internet Multimedia Mail standard (RFC 1341).

Scripting Language

Tcl is a powerful scripting language that runs under UNIX, Linux, VMS, DOS/Windows, OS/2, and MacOS (at least). It provides all the usual high-level programming features that we've come to expect from languages like the Unix shell, Awk, Perl, or Rexx, etc.

Embeddability

Tcl is a small language designed to be embedded in other applications (C programs for example) as a configuration and extension language. This minimizes the number of languages that users need to learn in order to configure their applications, and makes these applications programmable with no extra effort. In addition, Tcl is a complete and well-designed programming language, whereas many existing configuration languages were designed (to be kind) in an ad hoc manner.

Extensibility

Tcl is specially designed to make it extremely easy to extend the language by the addition of new primitives in C. These new primitives are truly first-class citizens of the language, sharing the same error handling and memory management as the original primitives. This has led to many useful extensions, such as DBMS access (extensions exist for Oracle, Sybase, Ingres, Postgres and many other DBMS's), SNMP, Motif, etc. In addition, this allows Tcl programs to be optimized by moving time critical code into C.

Equivalence of Data and Programs

Like Lisp, Tcl uses the same representation for data and for programs. This means that Tcl programs or scripts can be manipulated as data: stored in variables, written to and later read from files or databases, passed from one Tcl program to another across the Internet, etc. In addition, it means that the programmer can create new Tcl control structures or error handling routines as easily as writing a simple function.

Automatic Memory Management

All Tcl data structures are dynamically allocated and fully variable in size. The programmer never needs to allocate memory or specify maximum sizes.

Event-Driven Programming

Tcl supports event-driven programming (required for GUI programming) with the ability to associate Tcl code with any variable or array element (the code is executed automatically whenever the variable is read or written).

IPC between Multiple Tcl Applications

Tcl supports the passing of Tcl code as messages between Tcl applications running on different machines across the Internet. This allows client-server protocols which incorporate the full power of the Tcl language, and makes it possible to write very tightly-integrated applications.

Program Development Tools

Tcl has a powerful symbolic debugger, timing and profiling tools, and language sensitive editing modes for Emacs, a WYSIWYG GUI builder (xf), a real-time application monitor (tkinspect), etc.

Freely Redistributable

The source code for the Tcl language system is freely copyable for any purpose, so it can be used in commercial applications as well as academic applications and freeware.

Tcl Extended

Tcl adds primitives to the core Tcl distribution. Some of the primitives include: UNIX system calls, time and date parsing, Awk-like pattern matching in files, keyed lists (structs), profiling commands, TCP/IP commands, etc.

TheTk Toolkit

Tk is a Tcl embedding of Ousterhout'sTk toolkit for the X Window System. With Tk, the Tcl programmer can create GUI applications in a fraction of the time and code required if programming in a low-level language like C. There is also a very full-featured GUI builder for Tk called xf which allows you to build live interfaces without writing any code.

The Expect Scripting Language

Expect extends Tcl to allow scripts to control interactive applications which can't otherwise be programmed. It is similar to the scripting languages found in software terminal emulators like Crosstalk and Kermit, but isn't restricted to scripting access to communications ports.

Tcl Distributed Programming (Tcl-DP)**:** Tcl-DP adds a native (i.e., Tcl-based) remote procedure call interface to Tcl that makes it possible to write tightly-coupled networked applications. Using Tcl-DP, an application running on a machine elsewhere on the Internet can respond automatically to the modification of a variable in a local application.

Group Kit for Groupware**:** Group kit is a groupware toolkit designed to make it "easy" to develop multi-user applications for real-time distributed desk-top conferencing. The philosophy is that groupware development should be only slightly more difficult to code than single-user programs.

Safe-Tcl

Designed primarily for Enabled Mail applications, in which electronic mail messages can contain "live" code to be executed automatically when received, Safe-Tcl is a Tcl interpreter that doesn't allow any "unsafe" commands to be executed -- commands that might delete files or otherwise destroy valuable data. It can be used in any distributed application.

SNAPSHOTS FOR SIMULATION AND DATA TRANSMISSION

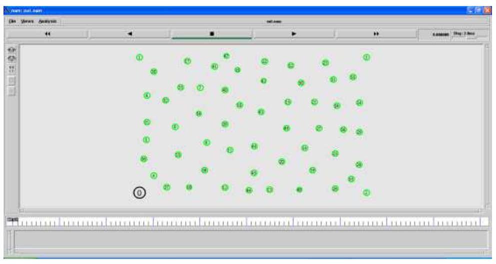
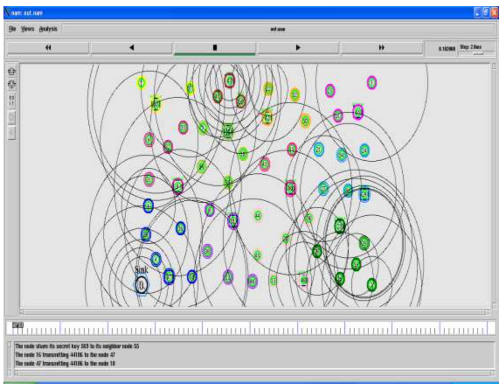
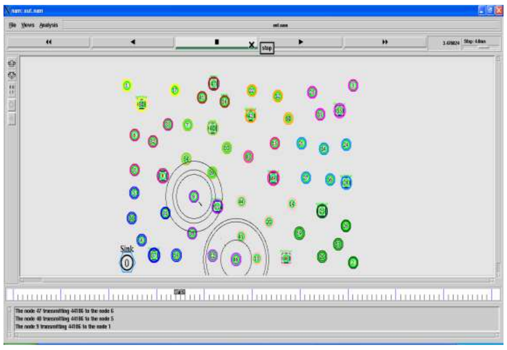


Figure 6: Representation of nodes



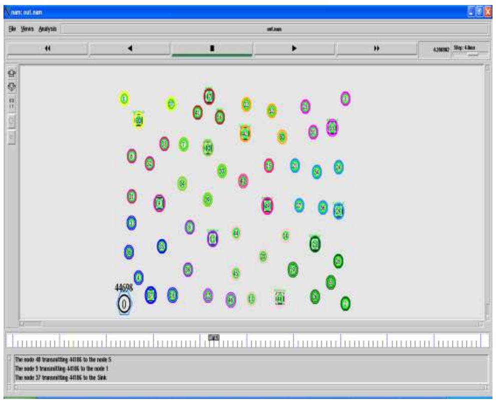
```
The node share its secret key 583 to its neighbor node 55
The node 16 transmitting 4410k to the node 47
The node 47 transmitting 4410k to the node 18
```

Figure 7: Finding the Shortest Path



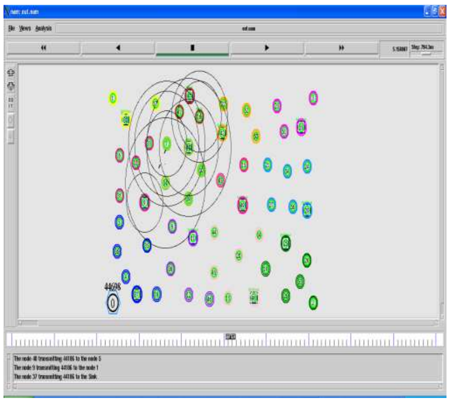
The node 47 transmitting 44186 to the node 6
The node 40 transmitting 44186 to the node 5
The node 9 transmitting 44186 to the node 1

Figure 8: Synchronization



The node 40 transmitting 44186 to the node 5
The node 9 transmitting 44186 to the node 1
The node 37 transmitting 44186 to the Sink

Figure 9: Data Send through the Source



The node 40 transmitting 44186 to the node 5
The node 9 transmitting 44186 to the node 1
The node 37 transmitting 44186 to the Sink

Figure 10: Data Transmission

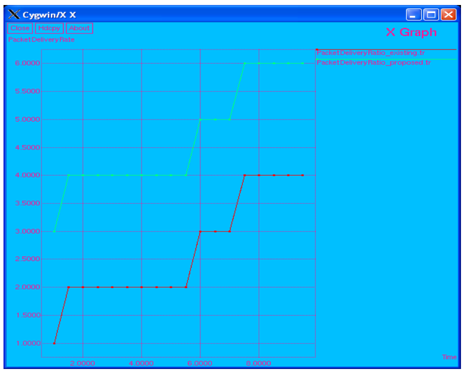


Figure 11: X Graph for Packet Delivery Ratio

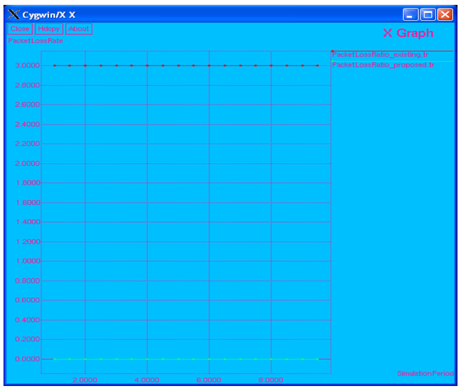


Figure 12: X Graph for Packet Loss Ratio

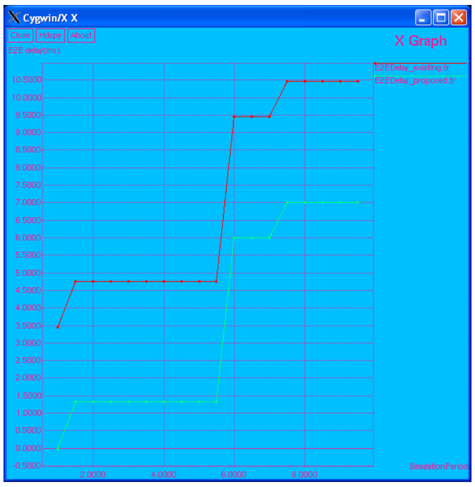


Figure 13: X Graph for Delay Timing

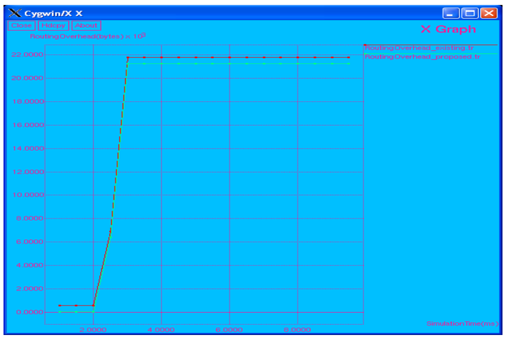


Figure 14: X Graph for Routing Overhead

CONCLUSIONS

In this project, we presented a secure and efficient Resource Conscious Routing (RCS) protocol for WSNs to balance the energy consumption and increase network lifetime. RCS has the flexibility to support multiple routing strategies in message forwarding to extend the lifetime while increasing routing security. We also proposed a non-uniform energy deployment scheme to maximize the sensor network lifetime. In experimental results, we can increase the lifetime and the number of messages that can be delivered under the non-uniform energy deployment.

FURTHER ENHANCEMENT

In our project, we send the data through the grid head. Every time if we are sending the data through the same grid head, there is a chance that head node will become a dead node. So in future we are using dynamic clustering method. In dynamic Clustering method, each sensor node determines its residual energy based upon consumed energy so far used in detecting events and transmitting its information.

This residual energy value determines whether the node should be considered as CHN candidate or not. The algorithm depends on calculating the residual energy of the sensor node and its distance from the base station if it is selected as CHN. The Non cluster head node (NCHN) detects CHN in its neighbor on the basis of multiple operations of WSNs using multiple rounds. The advantage of this approach is to provide enough flexibility to each NCHN to choose nearest CHN to reduce the energy consumption. We determine the residual energy of each node in WSN on basis of a mathematical model. Let us assume that there is single-hop communication is used among sensor nodes to detect events and to transmit the information.

REFERENCES

1. B. Karp and H. T. Kung, “GPSR: Greedy perimeter stateless routing for wireless networks,” in MobiCom’2000, New York, NY, USA, 2000, pp.243 – 254.
2. J. Li, J. Jannotti, D. S. J. D. C. David, R. Karger, and R. Morris, “A scalable location service fo geographic ad hoc routing,” in MobiCom’2000. ACM, 2000, pp. 120 – 130.
3. Y. Yu, R. Govindan, and D. Estrin, “Geographical and energy-awarerouting: A recursive data dissemination protocol for wireless sensor networks,”UCLA Computer Science Department Technical Report, UCLACSD,May 2001.
4. N. Bulusu, J. Heinemann, and D. Estrin, “Gps-less low cost outdoor localization for very small devices,” Computer science department, University of Southern California, Tech. Rep. Technical report00-729, April 2000.
5. A. Savvides, C.-C. Han, and M. B. Srivastava, “Dynamic fine-grained localization in ad-hoc networks of sensors,” in Proceedings of the Seventh ACM Annual International Conference on Mobile Computing and Networking (Mobi Com), July 2001, pp. 166–179.
6. P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, “Routing with guaranteed delivery in adhoc wireless networks,” in 3rd Int. Workshop on Discrete Algorithms and methods for mobile computing and communications,1999, pp. 48–55.
7. T. Melodia, D. Pompili, and I. Akyildiz, “Optimal local topology knowledge for energy efficient geographical routing in sensor networks,” in Proc. IEEE INFOCOM, vol. 3, March 2004, pp. 1705 –1716 vol.3.
8. Y. Li, Y. Yang, and X. Lu, “Rules of designing routing metrics for greedy, face, and combined greedy-face routing,” Mobile Computing,IEEE Transactions on, vol. 9, no. 4, pp. 582–595, April 2010.
9. R. Shah and J. Rabaey, “Energy aware routing for low energy ad hoc sensor networks,” in Wireless Communications and Networking Conference,2002. WCNC2002. 2002 IEEE, vol. 1, 17-21 March 2002, pp. 350–355 vol.1.