DIRECTED DIFFUSION ROUTING PROTOCOL: ATTACKS AND COUNTERMEASURES

BY

RAJKUMAR VIJAYARAMAN

Bachelor of Engineering

Annamalai University

Chidambaram, India

2001

Submitted to the faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 2005

DIRECTED DIFFUSION ROUTING

PROTOCOL: ATTACKS AND

COUNTERMEASURES

Thesis Approved:

_____

Dr. Johnson Thomas, Thesis Advisor.

_____

Dr. John Chandler.

_____

Dr. Venkatesh Sarangan.

_____

Dr. A. Gordon Emslie, Dean of the Graduate College

ACKNOWLEDGEMENTS

I wish to convey my sincere appreciation to my thesis advisor Dr. Johnson Thomas for the guidance and encouragement that he gave me throughout my thesis work. I would also like to extend my appreciation and gratefulness to Dr. John Chandler and Dr. Venkatesh Sarangan for their support and advice.

I would like to thank my mother Mrs. Saroja Vijayaraman my father Mr. Vijayaraman and my brother Mr. Natrajan for their love and support throughout the years.

Finally I would like to thank all my friends who stood beside me with their unfailing and indispensable support.

TABLE OF CONTENTS

CHAPTER ..... PAGE

I INTRODUCTION 1

1.1 Sensor Networks 1

1.2 TinyOS 2

II ROUTING IN SENSOR NETWORKS 4

2.1 Introduction 4

2.2 TinyOS Beaconing 5

2.3 Geographic Routing 5

2.4 Minimum Cost Forwarding 6

2.4 Clustering Based Protocols 6

2.5 Rumor Routing 7

2.6 Directed Diffusion 7

2.6.1 Naming 8

2.6.2 Interests and Gradients 9

2.6.3 Interest Propagation 9

2.6.4 Gradient Establishment 11

2.6.5 Data Propagation 12

2.6.6 Positive Reinforcement 13

2.6.7 Path Truncation Using Negative Reinforcement 14

III ATTACKS ON DIRECTED DIFFUSION 16

3.1 Introduction 16

3.2 Suppression 17

3.3 Cloning 17
3.4 Path Influence 18
3.5 Selective Forwarding and Data Tampering 18

IV KEY ESTABLISHMENT, ASSUMPTION AND DESIGN GOALS 19
4.1 Symmetric Key Algorithm 19
4.1.1 Pair-wise shared key 20
4.1.2 Global Key 20
4.1.3 Individual Key 20
4.2 Assumptions and Design Goals 21
4.2.1 Network and Security assumptions 21
4.2.2 Design Goals 22
4.3 Key Establishment 22
4.3.1 Introduction 22
4.3.2 Establishing Individual Key 23
4.3.3. Establishing Pair-wise shared key 23
4.3.4. Establishing Global Key 24
4.5 Node Addition 25

V PROPOSAL TO ENCHANCE DIRECTED DIFFUSION 27
5.1 Security Analysis 27
5.2 Assumptions 28
5.3 Cloning Attack 28
5.3.1 Proposed solution 30
5.3.1.1 RSSI (Received Signal Strength Indicator). 30
5.4 Flow Suppression 32
5.4.1 Normal operation 32
5.4.2 Attacks using negative reinforcement 33
5.4.3 Proposed Solution 33

5.5 Path Influence 34

5.5.1 Proposed Solution 35
5.5.2 Calculation of Remaining Energy 38
5.6 Selective Forwarding 38
5.6.1 Proposed solution 39
5.7 Data Fusion 41
5.7.1 Single Source Problem 42
5.7.2 Mobile Beacon 43
5.7.3 Proposed Solution 44
5.7.4 Implementation Issues 44
5.8 Effects of Node Inclusion and Exclusion 45
5.8.1 Node Addition 45
5.8.2 Node Leaving 46
5.8.2.1 Negative Reinforcement 47
5.8.2.2 Selective Forwarding 48
5.8.2.3 Path Influence and Cloning Attack 48

VI PERFORMANCE AND SECURITY ANALYSIS 49
6.1 Storage Requirement 49
6.2 Computational Cost 50
6.3 Communication Cost 51
6.4 Packet Size 53
6.5 Security Analysis 55

VII SIMULATION AND RESULTS 56
7.1 Simulation Set-up 56
7.2 Storage Requirement 56
7.3 Communication Cost 58
7.4 Computational Cost 60
7.4.1 Selective Forwarding 60
7.4.2 Negative Reinforcement 62

VIII CONCLUSIONS AND FUTURE WORK 64

REFERENCES **65**

APPENDICES ..... ERROR! BOOKMARK NOT DEFINED.

LIST OF FIGURES

Figure Page

1. Sensor Nodes 2
2. A representative topology constructed using TinyOS beaconing 5
3. Gradient Establishment 11
4. Data Propagation 13
5. Positive Reinforcement 14
6. Alternate Path 15
7. Cloning Attack 29
8. Signal strength measurement as a function of distance 31
9. Negative Reinforcement 32
10. Path Influence 35
11. Selective Forwarding 39
12. One mobile beacon assisting in the localization of a sensor field 43
13. Negative Reinforcement 47
14. Communication Cost 52
15. Memory Requirements 57
16. Power Consumption 59
17. Selective forwarding overhead 61
18. Negative Reinforcement overhead 63

LIST OF TABLES

Table Page

1. Table for storing RSSI and battery power 36
2. Packet with power information 36
3.TinyOS packet format 53
4. TinySec packet format 54
5.Packet with power information 54
6. Simulation Result for 10, 30 and 50 nodes 58
7. Simulation result for Communication Cost 60
8. Simulation result for Selective Forwarding 62

CHAPTER I

INTRODUCTION

1.1 Sensor Networks

Recent advances in miniaturization, low-cost and low-power design have led to active research in large-scale, highly distributed systems of small, wireless, low-power, unattended sensors and actuators [1]. The power of wireless sensor networks lies in the ability to deploy large numbers of tiny nodes that assemble and configure themselves. Usage scenarios for these devices range from real-time tracking, to monitoring of environmental conditions, to ubiquitous computing environments, to *in situ* monitoring of the health of structures or equipment [2]. While often referred to as wireless sensor networks, they can also control actuators that extend control from cyberspace into the physical world [3].

The sensors can be installed at pre-selected locations and data can be collected from specific location with the help of the neighbors. In this paper we target the Berkeley's MICA motes and the TinyOS sensor platform. The MICA sensor node is manufactured by Cross Bow technology and can be programmed using the nesC language. The hardware runs the programs in the TinyOS platform. The specifications of MICA mote are:

- 4MHz Atmega 128L processor.
- 128K bytes flash 4K bytes SRAM 4K bytes EEPROM.
- 916MHz radio transceiver with a maximum data rate of 40Kbits/sec.
- Attached AA (2) battery pack for power supply.
- An analog-to-digital converter (ADC) converts the analog signals from the different sensors on the sensor board to 10-bit digital format.

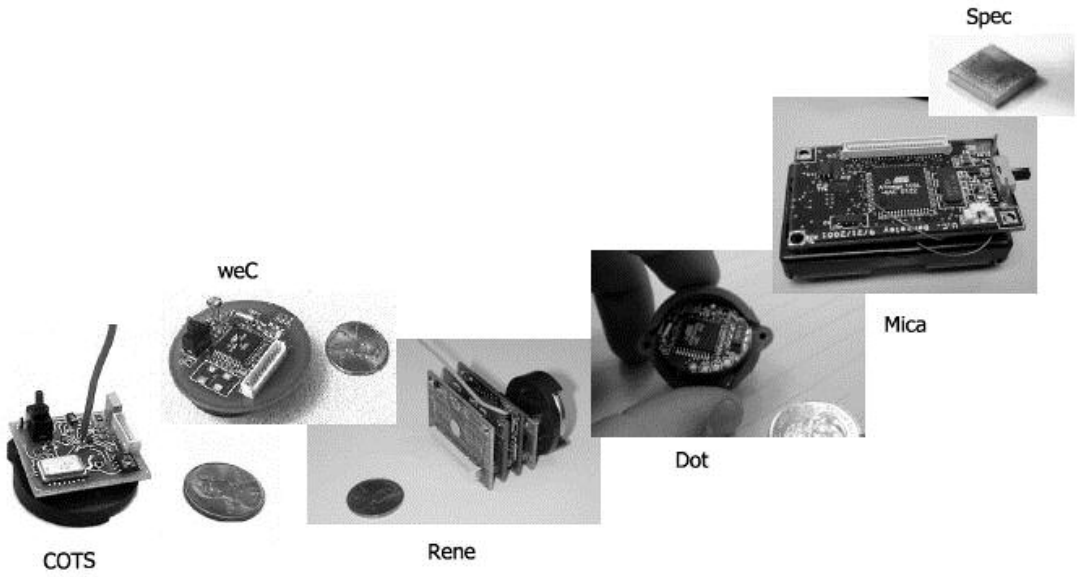

Figure 1: Sensor Nodes

1.2 TinyOS

The demands placed on the software of wireless sensor networks are numerous. It must be efficient in terms of memory, processor, and power so that it meets strict application requirements. It must also be agile enough to allow multiple applications to simultaneously use system resources such as communication, computation and memory

[3]. The extreme constraints of these devices make it impractical to use legacy systems [3]. TinyOS is an operating designed explicitly for networked sensors.

TinyOS executes only one program at a time. It holds the necessary software components in the memory. There are two threads of execution in TinyOS: *tasks and hardware event handlers* [4]. Tasks are functions whose execution can be differed. Once scheduled for execution, tasks run to completion. Hardware event handlers are the one which is invoked by the hardware interrupts. The hardware interrupts can preempt the execution of a task or other hardware event handler. The applications that run in TinyOS are written in nesC language.

CHAPTER II

ROUTING IN SENSOR NETWORKS

2.1 Introduction

A sensor network is formed by hundreds or thousands of low-cost, low-power nodes to monitor events in an area. A node communicates with other nodes that lie within the transmittable range to accomplish the given tasks. Due to the storage and energy constraints the sensor network routing protocols are much simpler than any other network routing protocols. The sensor network is often compared to the ad-hoc networks. Like the sensor networks, the ad-hoc networks also depend on the cooperative nature of the nodes. The main difference between the ad-hoc networks and the sensor network is the computational power of the nodes. The ad-hoc routing protocols are designed for the traditional nodes rather than for the special hardware like the MICA mote. The various routing protocols proposed for the sensor networks are:

1. TinyOS Beaconing.
2. Geographic Routing.
3. Minimum Cost Forwarding.
4. Clustering Based Protocols.
5. Rumor Routing.
6. Directed Diffusion.

2.2 TinyOS Beaconing

The TinyOS beaconing protocol constructs a breadth first spanning tree rooted at a base station [1]. The base station broadcasts the route update message periodically. The nodes which receive this message update the routing information and also mark the base station as its parent. Each node also broadcasts the routing information, the nodes receiving this message marks the node from which it gets the message as its parent. The algorithm continues recursively with each node marking its parent as the first node from which it hears a routing update during the current *time epoch* [1].

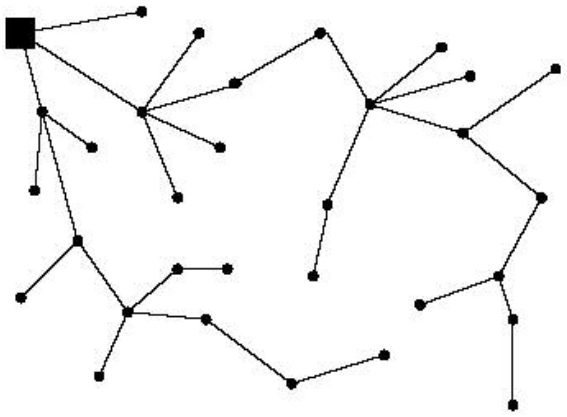

Figure 2: A representative topology constructed using TinyOS beaconing

2.3 Geographic Routing

The Geographic and Energy Aware Routing (GEAR) routing protocol depends on the geographic location of the nodes. The Greedy Perimeter Stateless Routing (GPSR) is a variation of geographic routing protocol. The GPSR uses greedy forwarding at each

hop, forwarding to the neighbor closest to the destination. When holes are encountered where greedy forwarding is impossible, GPSR recovers by routing around the perimeter of the void [1]. One drawback of GPSR is that packets along a single flow will always use the same nodes for the routing of each packet, leading to uneven power consumption in nodes [1].

2.4 Minimum Cost Forwarding

All the nodes in the network maintain a *cost field*. The *cost field* specifies the minimum cost required to reach the base station. To forward a packet to the base station, the node checks the cost field associated with a neighbor and chooses the minimum cost route. The *cost field* can store any metric like hop-count, energy, latency or loss. The advantage of this protocol is that there is no need to maintain the path information. This protocol also has the disadvantage of using the same link for forwarding the packets.

2.4 Clustering Based Protocols

LEACH (*Low-Energy Adaptive Clustering Hierarchy*) leverages clustering to efficiently disseminate queries and gather sensor readings to and from all nodes in the network [1]. LEACH assumes that every node in the network can reach the base station directly. But reaching the base station in one-hop is a high power operation which is inefficient considering the power available in the sensor nodes. LEACH organizes the nodes into clusters with one node acting as a cluster head. The nodes within a cluster send the collected data to its cluster head and the cluster heads of the entire cluster

communicate to aggregate the data. The aggregated data is then forwarded to the base station.

2.5 Rumor Routing

Rumor routing is a probabilistic protocol for matching queries with data events. The query from the base station is flooded in the entire network. The data events in response to the queries are shared by the nodes. The base station does not depend on a single link to receive the events. The disadvantage of this algorithm is the cost associated with the flooding of the queries and the data events. A variation of rumor routing algorithm injects the interest queries only to a small number of nodes. An agent is used to collect the information about the events. The information collected by the agent is shared with the nodes in the network.

2.6 Directed Diffusion

Directed diffusion consists of several elements: interests, data messages, gradients and reinforcements [5]. The base station floods the sensor network with the query about the interested events. The query is called as the interest in the sensor networks. Each interest contains the description about the sensing task. The data messages are the events generated by a single or a group of nodes in response to the query send by the base station. In directed diffusion, query is named using attribute value pairs [5]. The interest queries are disseminated throughout the sensor network as an interest for named data. This dissemination sets up the “gradients” within the network to draw events. A gradient

is direction state created in each node that receives an interest [5]. The node, which generates the events, sends the events back to the base station along multiple gradient paths. The directed diffusion algorithm assumes that each node knows its location once deployed.

2.6.1 Naming

In [5], location tracking task was used to explain the directed diffusion algorithm. The same application is used in this paper to see how query is disseminated in to the network. The query send by the base station is a list of attribute-value pair. For locating a vehicle the task description will be,

```
Type=Wheeled vehicle //Detect Vehicle location
Interval=20 ms       //send events every 20 ms
Duration=10 seconds  //send data till this time
Rectangle= [-100, 100, 200, 400]
```

The data sent in response to this query is also named. For example the sensor that detects the wheeled vehicle might respond like,

```
Type=Wheeled vehicle      //type of vehicle seen
Instance=truck            //instance of this type
Location= [125, 220]      //node location
Intensity= 0.6            //signal amplitude measure
Confidence= 0.85          //confidence in match
Time Stamp= 01:20:40      //Event Generation Time
```

2.6.2 Interests and Gradients

The interest is injected in to the network by the base station.

2.6.3 Interest Propagation

The [5] describes how interests are diffused throughout the network. Using an example task, the authors of [5] explain the propagation of interest in the sensor networks. We will use the same example here. They have considered that a task with specified type, rect, duration of 10 minutes and an interval of 10 ms is instantiated at the base station. The interval parameter specifies an event data rate. So for our example the data rate will be 100 events per second. The base station records the task. The task will be removed from the base station after the time specified in the duration field. For each active task the base station broadcasts this message periodically. The initial message for setting up the gradients and fetching the data will have much larger interval. Intuitively, this initial message is thought of as exploratory; the base station tries to determine if there indeed some nodes in the specified region senses the task specified [5]. The following is query is an exploratory message,

Type=Wheeled vehicle

Interval=1 ms

Rect= [-100, 200, 200, 400]

Time stamp=1:20:30

Expires at=1:30:30

The base station periodically refreshes the message. This is done by simply sending the same interest message with increasing time stamp. This is necessary since the interests are not reliably transmitted in the network. The refresh rate is a protocol design parameter that trades off overhead for increased robustness to lost interests [5].

Each node in the network maintains an interest cache. The interest cache contains the information about the interest it received. Two interests are considered distinct, if atleast one of the attribute values is different. Interest cache does not have information about the base station but the one-hop neighbor from which it received the interest. Thus, the interest rate scales with the number of distinct active interest. There are several fields in the interest cache. A time stamp field indicates the time stamp of the last event received. The interest cache also contains several gradient fields, up to one per neighbor. Each gradient contains the data rate field which contains the data rate requested by the corresponding neighbor, derived from the interval attribute. It also maintains a duration field derived from the time stamp and the expiresAt attributes.

When a node receives an interest, it checks the interest cache to see if the interest already exists. If no matching entry exists in the interest cache, then the node creates one interest entry and stores the information about the interest. This entry has a single gradient towards the neighbor form which it received the interest [5]. For the example taken in [5], the neighbor of a base station will setup an interest entry with a gradient of 1 event per second towards the base station. The node has to distinguish the neighbors in order to send the data at the requested rate. Use of 802.11 MAC address is discussed in

[2] for this purpose. If the entry already exists then the time stamp and expires at field are updated. When a gradient expires, it is removed from its interest entry. After storing the information about the interest, a node will send the interest to all its neighbors.

2.6.4 Gradient Establishment

Figure 3 shows the gradient establishment between the nodes. Every pair of neighboring node establishes a gradient towards each other. This is because; when a node receives an interest message from its neighbor it has no way of knowing whether that interest was in response to the one it sent out earlier, or it is an identical interest from the other side of the neighbor [5]. Such a two way gradients can cause a node to receive one copy of low data rate events from each of its neighbors [5]. But this technique can enable the fast recovery from failed nodes.

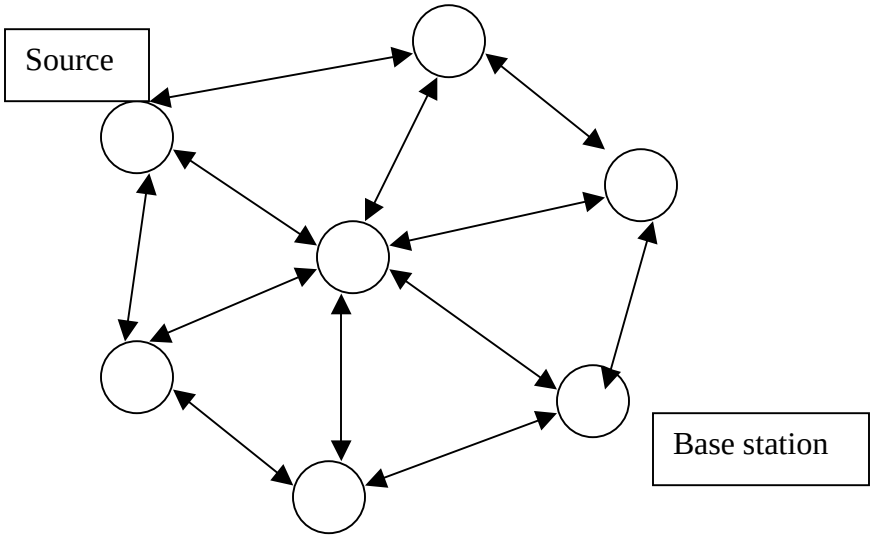

Figure 3: Gradient Establishment

The gradient specifies the data rate and the direction in which to send events. In summary, interest propagation sets up state in the network to pull down the data events from the source node. The rules for interest propagation are application specific [5]. Due to the multi-path transmission of the interest, it is not possible for an adversary to prevent the interest information from reaching the nodes in the network [5].

2.6.5 Data Propagation

The node which lies in the specified rect value, tasks its sensors to begin collecting samples. If the node finds the target then it will search its interest cache for a matching interest entry. If a matching interest is found then the node will look at the data rate parameter for all the gradients and forward the data at the rate specified. In our case this will be 1 event per second initially for all gradients. So, all the neighbors receive a copy of the event. The source node unicast the events, to all the neighbors for which it has a gradient. If the data rates of downstream nodes are different, then the source node interpolates the messages it is sending to the high data rate neighbor. The interpolated message is send to the low data rate neighbor.

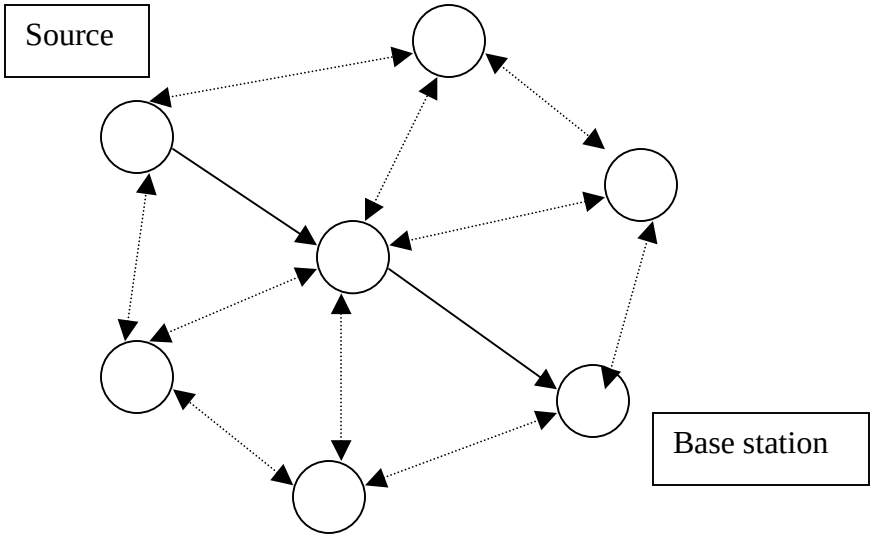

Figure 4: Data Propagation

The node which receives the events from the source, attempts to find a matching entry in its interest cache. If a match does not exist then the data message is dropped silently. If there exists a match, then the node checks the data cache associated with the matching interest entry. The data cache keeps track of recently seen data items. It has several potential uses, one of which is loop prevention. If a received data message has a matching data cache entry, the data message is silently dropped. Otherwise, the received message is added to the data cache and the data message is sent to the node's neighbors [5].

2.6.6 Positive Reinforcement

The data message will eventually reach the base station. The base station reinforces one particular neighbor, and that neighbor, reinforces one of its upstream neighbors. The reinforcement continues till the message reaches the source node. The

reinforcement is nothing but the same interest message with the increase in the data rate. The higher data rate events allow high quality tracking [5]. The local rules for choosing one particular node are application specific. Any node in the network can send a positive reinforcement to its upstream neighbor if that node consistently sends the unseen event to it.

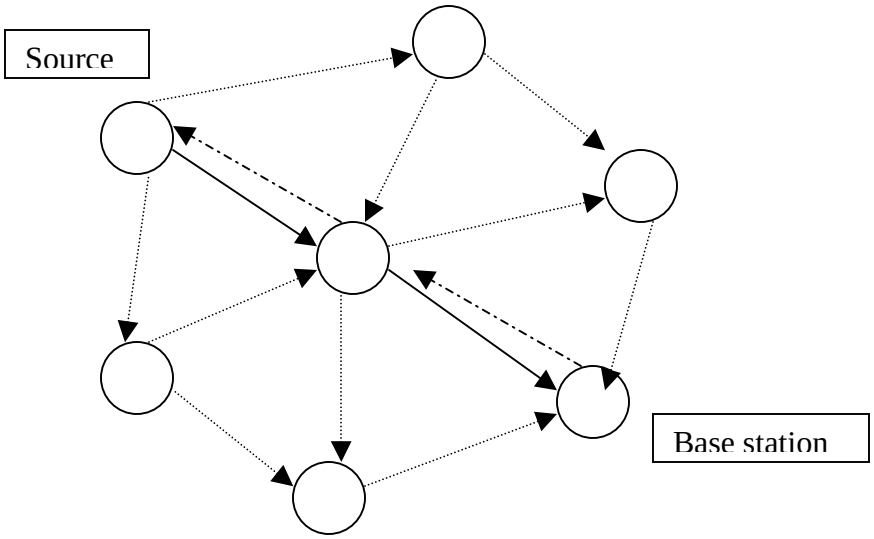

Figure 5: Positive Reinforcement

2.6.7 Path Truncation Using Negative Reinforcement

A node on the data flow path can also be negatively reinforced if that node cannot consistently supply new events to the downstream nodes. An interest message, with the initial exploratory data rate is send to the node which needs to be negatively reinforced. Like positive reinforcement, the negative reinforcement message is also forwarded to the source node. The nodes receiving this message change their data rate of the neighbor. For example, let us consider from the following figure that the data flows in the path A->C->E->G and the link A->C is congested. Now node E will receive the data message from

the node D (since the other link is congested). This prompts node E to negatively reinforce node C and positively reinforce node D.

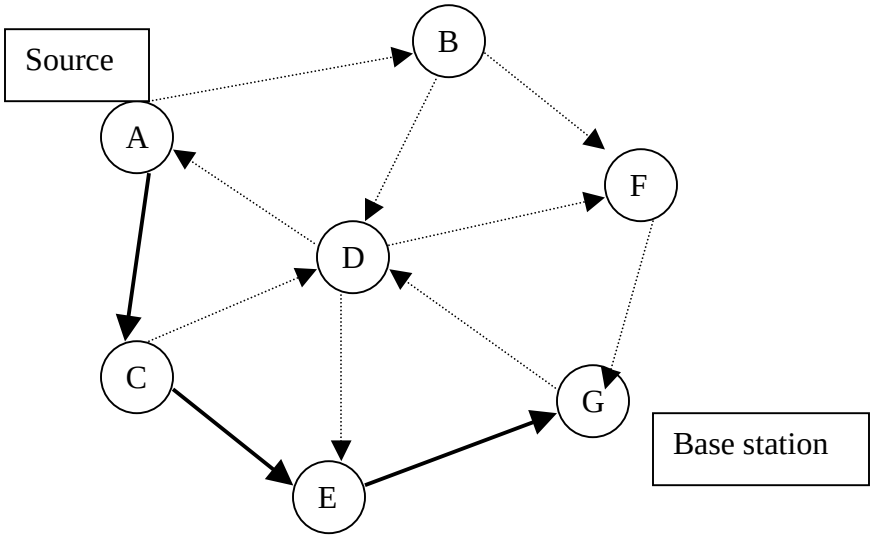

Figure 6: Alternate Path

CHAPTER III

ATTACKS ON DIRECTED DIFFUSION

3.1 Introduction

One aspect of sensor network that complicate the design of a secure routing protocol is the in-network aggregation [1]. At various locations in the network, the nodes need to aggregate the information from other nodes. This help in removing the duplicate messages flowing in the network. End-to-end mechanism is used in the conventional network for message authenticity, integrity and confidentiality. But end-to-end security mechanism is not feasible in sensor network because the communication is mainly between the one-hop neighbors. Since the MICA motes have a very slow processor (4 MHz), it is not possible to use the 128 bit encryption mechanism. The directed diffusion algorithm does not use any keying mechanism to prevent the nodes from outsider attacks. In the following sections we explain the possible attacks on directed diffusion algorithm. The four possible types of attacks possible in directed diffusion routing algorithm are:

1. Suppression
2. Cloning
3. Path influence
4. Selective forwarding and data tampering.

An attacker with laptop-class devices can do more than an attacker with only ordinary sensor nodes. An ordinary sensor node might only be able to jam the radio link in its immediate vicinity, while a laptop-class attacker might be able to jam the entire sensor network using its stronger transmitter. A single laptop-class attacker might be able to eavesdrop on an entire network, while sensor nodes would ordinarily have a limited range [5]. Also, laptop-class attackers might have a high-bandwidth, low-latency communications channel not available to ordinary sensor nodes, allowing such attackers to coordinate their efforts [5].

3.2 Suppression

Flow suppression is an instance of denial of service attack. The easiest way to suppress a flow is to spoof negative reinforcements. The adversary simply sends the negative reinforcement to the node which is delivering data at a high rate. The adversary has to find out the unique id of the downstream node and the interest information. Since there is no keying mechanism to verify the authenticity of the message, the upstream neighbor which received the negative reinforcement further forwards the message to its upstream node.

3.3 Cloning

Each node in the network will be assigned a unique id before deployment. The base station will also have a unique id. The base station can inject the interest information at any node in the network [5]. If the goal of the adversary is to receive the data messages

generated by the source node then the adversary has to announce that it a base station. This is done by receiving the interest information from the authentic base station and modifying the packet information. The adversary will change the source id in the packet to its own id. Now the adversary also receives a copy of the data message from the source node.

3.4 Path Influence

By injecting information about the energy available to do complex calculations and/or about the availability of quality link the adversary can influence the nodes in the data flow path. For example if the adversary is located in the low-data flow path, it can influence all the nodes in the high data flow path by sending an announcement that it has more remaining battery power to do multiple calculations. This attracts the neighboring nodes to send the packets to the adversary. The attacker also can announce the availability of high quality link by sending a powerful signal which makes the high data flow path nodes to send the data to the adversary. Now the data will flow through the adversary. The adversary can also launch this attack by spoofing positive and negative reinforcement.

3.5 Selective Forwarding and Data Tampering

By using the above attack to insert itself onto the path taken by a flow of events, an adversary can gain control of the flow. The adversary can modify the packets or forwards the packets selectively.

CHAPTER IV

KEY ESTABLISHMENT, ASSUMPTION AND DESIGN GOALS

4.1 Symmetric Key Algorithm

In symmetric key algorithm only one key is used for both encryption and decryption. It requires a shared key between the nodes. When compared to the asymmetric key algorithm, the calculation is nearly 1000 times faster [6]. TinySec [7], a security mechanism provided for sensor nodes in TinyOS environment uses the symmetric key algorithm. SPINS [6], another security protocol for sensor network also uses symmetric key algorithm to provide message authentication, integrity and confidentiality. In this paper we are proposing the integration of LEAP [8], a security protocol for the sensor networks, in to directed diffusion. The reason we choose LEAP (Localized Encryption and Authentication protocol) is because it provides security based on the importance of the message. For example the announcement from the base station gets a minimal security when compared to a packet from source node which gets maximum security. Four different keys are used in each node to provide various level of security. In the other two mechanisms (SPINS and TinySec), only one key is used. The four keys used in LEAP algorithm are:

1. Cluster Key
2. Global Key

3. Pair-wise Key and.
4. Individual key

4.1.1 Pair-wise shared key

Each node shares a pair-wise key with all its neighbors. This key is used for the secure communication between a node and one of its neighbors. Most of the communication in sensor networks is one hop, so it is logical to have a pair-wise shared key for all the nodes.

4.1.2 Global Key

All the nodes in the sensor network share a common key with the base station. The base station uses this key to encrypt the interest message and all the nodes in the network uses this key to decrypt the announcements from the base station. The nodes store the interest information in their interest cache and then encrypt the message using the global key to further broadcast it. The communication cost is reduced by using this key, since it is enough to broadcast the interests.

4.1.3 Individual Key

Each node will have a unique key. This key is shared between the base station and a node. The individual key is used for secure communication between a node and the base station. The base station uses this key to verify the messages sent by this node and also for updating the global key of the node.

The LEAP algorithm was built for all the routing protocols. The protocols such as LEACH in which the nodes form many clusters, can use the cluster key. In directed diffusion, all the communications are between one-hop neighbors so there is no need for a cluster key. In this paper we discuss how a reasonable level of security can be provided by using three keys.

4.2 Assumptions and Design Goals

4.2.1 Network and Security assumptions

We assume that the sensor network is static, i.e., sensor nodes are not mobile. The base station, acting as a controller (or key server), is assumed to be a laptop class device and supplied with long-lasting power. We also assume that the base station is equipped with a powerful transmitter and it can send the message to any node in the network in one-hop. The sensor nodes are similar in their computational and communication capabilities and power resources to current generation sensor nodes, e.g. the Berkeley MICA motes. We assume that every node has space for storing up to hundreds of bytes of keying materials. The sensor nodes can be deployed via aerial scattering or by physical installation. However, we assume that the immediate neighboring nodes of any sensor node will not be known in advance. Because wireless communication is not secure, we assume an adversary can eavesdrop on all traffic, inject packets or replay older messages. We also assume that the base station will not be compromised.

4.2.2 Design Goals

The main goal of our algorithm is to design an efficient security mechanism for supporting communications in directed diffusion. The security requirements not only include authentication and confidentiality but also robustness and survivability. In other words, the sensor network should be robust against various security attacks, and if an attack succeeds, its impact should be minimized. For example, the compromise of a single node should not break the security of the entire network.

4.3 Key Establishment

4.3.1 Introduction

The idea of establishing the individual, global and pair wise shared key is taken from the LEAP protocol. The LEAP protocol uses the Pseudo-Random functions to derive the keys. The base station derives one master key K_I and this key is loaded in to all the nodes in the network before deployment. Each node is also assigned a unique id and a global key before deployment. The LEAP protocol uses the following notations.

N- Number of nodes in the network.

u, v – nodes in the network.

$\{f_k\}$ – Family of pseudo random functions

$\{S_k\}$ – Encryption of message S with key k

MAC (k, S) is the message authentication code (MAC) of message S using a symmetric key k.

From a key K a node can derive other keys for various security purposes. For example, a node can use $K_0 = f_K(0)$ for encryption and use $K_1 = f_K(1)$ for authentication [8].

4.3.2 Establishing Individual Key

Every node has an individual key that is only shared with the base station. The individual key K_u for a node u (each node has a unique id) is generated as follows: $K_u = fK_I(u)$. Here f is a pseudo-random function and K_I is a master key. In this scheme the base station might keep only its master key to save the storage needed to keep all the individual keys. When it needs to communicate with an individual node u, it computes K_u on the fly. Due to the computational efficiency of pseudo random functions, the computational overhead is negligible [8].

4.3.3. Establishing Pair-wise shared key

Since the communication in sensor network is always among the neighbors, the pair-wise shared key is used more than any other key. The establishment of pair-wise shared key is described in the LEAP algorithm. We are using the idea described in the LEAP for our case.

The authors of LEAP assume that the nodes in the network can withstand the possible attack by the adversary for atleast a short interval (in seconds). They assume that the time required to establish all the keys in the network (T_{est}) is less then the time required (T_{min}) by the adversary to compromise one or more nodes.

When a node u is deployed, it first initializes a timer to fire after time T_{min}. It then tries to discover its neighbors. It broadcasts a HELLO message which contains its id and waits for each neighbor v to respond with an ACK message including the identity of node v. The ACK from every neighbor v is authenticated using the individual key K_v of node v, which was derived as $K_v = fK_I(v)$. Since node u knows K_I, it can derive K_v and then verify node v's identity [8].

u → *: u. ; /* Node u broadcasts the Hello packet specifying with its id in the packet */

v→u: v, MAC(K_v, u-v) ; /* v replies to u with its id and the calculate MAC. The MAC is calculated using the individual key K_v for the message u-v (the ids of u and v).*/

Node u computes its pair-wise key with v, K_{uv}, as $K_{uv} = fK_v(u)$. Node v can also compute K_{uv} in the same way [8]. K_{uv} serves as their pair-wise key. No message is exchanged between u and v in this step. Note that node u does not have to authenticate itself to node v by sending a special message, because any future messages authenticated with K_{uv} by node u will prove node u's identity [8].

4.3.4. Establishing Global Key

The global key can be established before the deployment of sensors. Since all the nodes are going to share the same key with the base station, the loading of this key can be done before deployment.

4.4 Key Erasure

When the timer T_{min} expires, node u erases K_I (the master key) and all the individual keys K_v's of its neighbors, which it computed in the neighbor discovery phase [8]. Note that node u does not erase its own individual key K_u. Every node keeps its own individual key.

After the above steps, node u will have established a pair-wise shared key with each of its neighbors and erased key K_I. No nodes in the network possess K_I. An adversary may have eavesdropped on all the traffic in this phase, but without K_I it cannot inject erroneous information or decrypt any of the messages.

4.5 Node Addition

If a node does not establish a pair-wise key with a neighbor before T_{min}, it cannot establish a pair-wise key for ever, since all the nodes in the network would have erased the master key K_I. Note that a new node entering the network can establish the pair-wise key with the neighbors [8]. For example let us say that node u is the new node with the master key K_I and node v is already in the network. After deployment, node u will derive its individual key K_u using K_I and then it sends the broadcasts a "Hello" packet. On receiving the "Hello" packet from node u, node v derives the pair-wise key K_{uv} as, $K_{uv} = fK_v(u)$. Node v then calculates the message authentication $MAC(K_v,u\text{-}v)$ and unicasts to node u. Since node u posses the master key K_I it also can calculate the pair-wise key.

The problem with this approach is if an adversary is successful in finding the master key K_I before the time $T_{min,}$ then the adversary can derive the entire pair-wise keys. LEAP solves this problem by introducing the concept of multiple master keys. LEAP algorithm assumes that there are at most m node addition events and that these m events occur in m intervals T_1, T_2, T_3,…T_m, respectively, where these intervals could be of different lengths [8]. The base station generates these m keys (K_1, K_2, K_3,... K_m). A node u deployed in time interval Ti is loaded with the key K_i, from which the node derives the individual key K_i (u) =fK_i(u) [8]. It is also loaded with individual keys K_j(u) = fK_j (u) for all i < j <= m, and it will use the individual key K_j(u) (i < j<= m) for establishing pair-wise keys with nodes deployed in T_j.

When deployed, node u uses K_i to establishpair -wise keys with its neighbors and then erases K_i, as in LEAP's original scheme. For example, a node u deployed in the first time interval T_1 is loaded with K_1 and K_j(u)'s for all 1< j <=m. It derives its individual key K_1(u) and uses K_1 as the initial master key to establish pair-wise keys with its neighbors, and then erases K_1 after T_{min}. Note that the base station should also erase the initial key K_1, since it is no longer needed [8]. When adding a sensor node v in T_2, the network controller loads node *v* with master key K_2, from which the node derives its individual key K_2(v), and m−1 individual keys K_2(v),K_3(v), ...,K_m(v) [8]. Node v uses K_2 to establish pair-wise keys with the neighboring nodes that were deployed in T_1 or T_2, because v can derive the individual keys of its neighbors in T_2. When its timer expires, node v erases K_2.

CHAPTER V

PROPOSAL TO ENCHANCE DIRECTED DIFFUSION

5.1 Security Analysis

With the introduction of keying mechanism in directed diffusion, only the nodes which have the keys can send and receive packets in the network. The outsider attack is completely ruled out.

The size of the keys used in LEAP is 8 bytes [8]. It has been proved that a laptop class adversary can crack the message which was encrypted using the symmetric key algorithm [14]. It is possible that an adversary finds out the keys. If the adversary is successful in finding the key, then the above said attack can still be launched. This is the insider attack, an attack which is very difficult to find. Because of its disadvantages, the use of symmetric key algorithm alone is not enough for providing security in sensor networks. We propose new methods by which we can differentiate between authentic and fake packets. We show how our scheme minimizes the effects of an attack.

5.2 Assumptions

We assume that the sensor network is dense and there will be atleast two neighbors (with enough power) for each node. We also assume that the probability to break more than one pair-wise key of the same node is very minimal.

5.3 Cloning Attack

If the goal of the attacker is to receive a copy of the data messages from the source node, then this type of attack is launched. To launch this attack the adversary need the global key. If the adversary successfully breaks the global key and decrypts the interest message from the base station, then it can forge the message with itself listed as the base station. The neighboring nodes further forwards this message in the network. This sets up gradients along the path. The adversary is yet to achieve the goal. The adversary needs atleast one pair-wise key shared between the nodes in the data flow path to decrypt the data message sent by the source node.

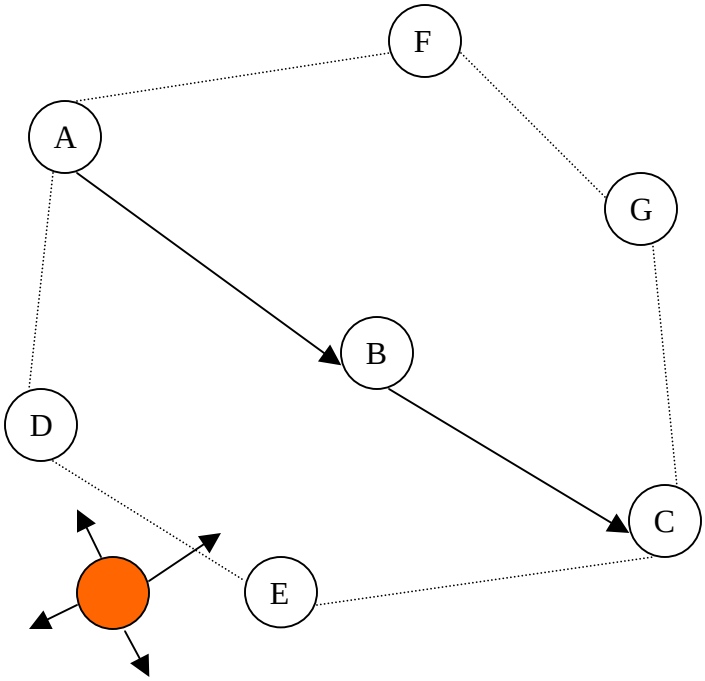

Figure 7: Cloning Attack

In figure 7, let A be the source node and C be the base station. The base station encrypts the interest message using the global key and broadcasts. Let us assume that the adversary has broken the global key. Since the adversary holds the global key it can decrypt the interest information from C and can forge the interest information with itself listed as the base station [1]. The adversary can then broadcast the fake message. The nodes which receive the fake message, decrypts using the global key and stores the information in their interest cache. This will form a new path to pull down the data from the source node. The adversary needs one pair-wise key shared between any nodes in the data flow path to decrypt the data messages from the source node, since data messages are encrypted using the pair-wise shared key.

5.3.1 Proposed solution

If each node in the network maintains the distance information of the base station, then the probability for this type of attack can be reduced. The distance of a node can be calculated using the signal strength of the message it transmitted. Various papers [9, 10, 15, 16, 17, and 18] discuss the possibility of using the RSSI (Received Signal Strength Indicator) values to localize the sensor network. Localization is a scheme by which all the nodes in the sensor network will learns about the location it is situated using one or more mobile nodes.

5.3.1.1 RSSI (Received Signal Strength Indicator).

Definition: A signal that indicates the strength of the incoming (received) signal in a receiver.

Using the signal strength, the distance of the node is found. Papers [9, 11 and 18] uses RSSI alone to find the distance where as paper [12] uses RSSI and acoustic signal in determining the distance of the node from which the signal is received. The advantage of the scheme discussed in [12] is the accuracy when compared to the schemes introduced in [9, 11 and 18] but it needs additional hardware. The MICA mote already comes with the hardware necessary to calculate the RSSI.

The signal strength is directly proportional to the remaining battery power. With the decrease in the battery power the signal strength reduces and the distance increases. This causes the distance estimation with errors but we have assumed that the base station

is equipped with long lasting power, so the signal strength never reduces in this case. The nodes can find out the distance of the base station with reasonable amount of accuracy.

During the initial set up time, the base station should broadcast the “Hello” packets powerful enough to reach all the nodes in the network. Each node receives this message and stores the RSSI value of the base station. Each node has to calculate the RSSI value of any future message received from the base station and compare with the existing value. Even if the adversary has the powerful transmitter, it is very difficult to transmit with the same power in order to fake the nodes. Note that there is a chance that the noise level is high or there are many obstructions in the path. This causes the node to receive a weaker signal and leads to a problem of nodes rejecting the authentic base station’s message. This can be solved by having a MAX value and a MIN value of RSSI. The nodes in the network should accept the packets if the RSSI is with in the MAX and MIN value.

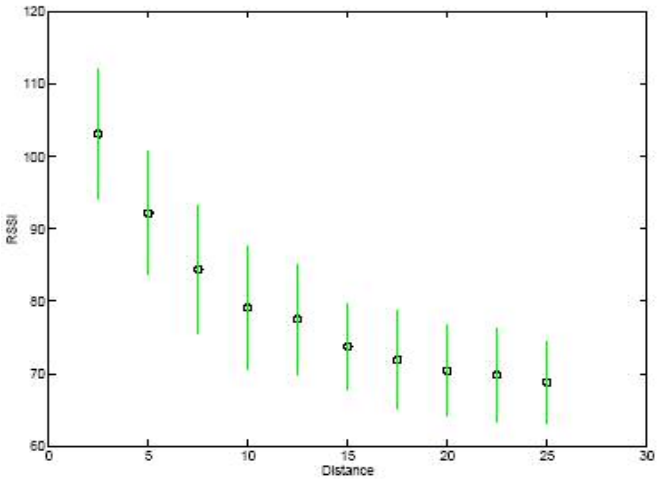

Figure 8: Signal strength measurement as a function of distance

5.4 Flow Suppression

Definition: A data flow path can be suppressed if a node gives a negative reinforcement to its upstream neighbor. A negative reinforcement is nothing but the interest information with initial exploratory data rate.

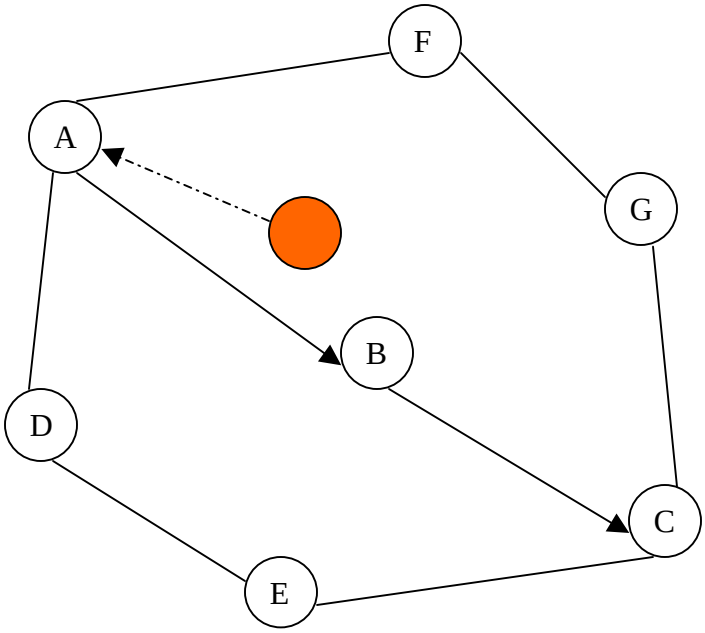

Figure 9: Negative Reinforcement

5.4.1 Normal operation

Let us consider that node A as the source node and node C as the base station from figure 9. Let us assume that the data flow path is A->B->C. The data flow path nodes supply the data at a much faster data rate when compared to the other nodes in the network. If the path between A and B or B and C is congested and if the congestion prevails for long time, then the paths A->D->E or A >F >G gives the required data to the base station at a faster rate. If the nodes E or G supplies the new events consistently to the

base station then the base station will positively reinforce E or G. After positively reinforcing one of E or G, the base station also negatively reinforces B. This will cause the node B to negatively reinforce the source (A) thereby suppressing the data flow.

5.4.2 Attacks using negative reinforcement

If the adversary breaks the pair-wise key shared between the nodes A and B then, it can simply send a negative reinforcement to node A posing as node B. Node B receives this information and changes the data rate value in the interest cache. This is a simple instance of denial of service attack.

5.4.3 Proposed Solution

For negative reinforcement information to be valid and processed further, the node which received the negative reinforcement should also receive the information about negative reinforcement from atleast one more neighbor. For example, if B sends a negative reinforcement to A and if it has to be valid, then the adversary has to compromise atleast one other neighbor of B (breaking the pair-wise keys used between B and one other neighbor). Since we have assumed that it will be difficult for the adversary to break more than one pair-wise shared key of a single neighbor, the negative reinforcement attack can be prevented. Note that we also have assumed that each node will have atleast two neighbors. If node A did not get the confirmation of negative reinforcement information from one other neighbor then it decides that the negative reinforcement information is not authentic and sends the information to the base station for key revocation.

5.5 Path Influence

The adversary can attract all the traffic through itself by announcing the availability of more energy and/or the availability of a high quality link to reach the base station (by sending a powerful signal). All the nodes start sending the packets to this adversary. After receiving the packets, the adversary can choose to forward packets selectively or change the packet information and forward it.

Let us consider the node A as the source and node C as the base station from figure 10. If the adversary breaks the pair-wise key shared by nodes E and B, it might try to attract all the nodes in the network by sending a fake announcement to B .The message can be regarding the availability of high quality path to base station and/or the high remaining battery power. On receiving the information, B further forwards to all its neighbors. The nodes which need to perform long calculation and the nodes which need to send the packet to the base station will be attracted. To prevent this type of attack we propose a new technique using the RSSI value and the remaining energy.

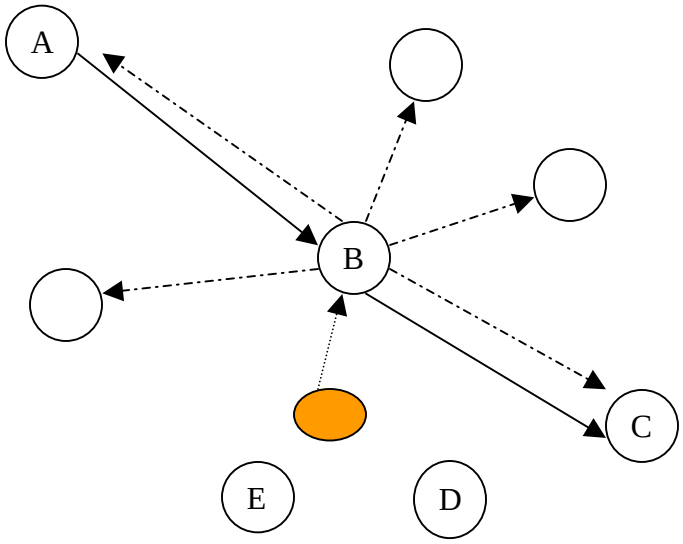

Figure 10: Path Influence

5.5.1 Proposed Solution

We have earlier assumed that the adversary takes T_{min} time to break a pair-wise key of a node and it is larger than the time it takes for all the sensor nodes to establish the pair-wise keys. During this initial setup period each sensor node should construct a hash table with the node id as the unique key. The table should contain remaining battery power and RSSI value for all its neighbors. A hash table is a data structure to store the data and can be searched for an item in O (1) time. The nodes store the data during the pair-wise key setup time. Each node is expected to append its remaining battery power with every packet so that the neighbor nodes can construct the table. Note that because of our assumption that the adversary takes atleast T_{min} time to break one or more keys, the initial packets from the nodes will be authentic and the information about battery will resemble the true value.

Node ID	Energy Remaining	RSSI

Table1: Table for storing RSSI and battery power

Let us assume that the adversary has broken the pair-wise shared key between any two nodes. To attract a neighbor, the adversary has to announce that the node has a high remaining power. Now if the adversary tries to send a fake message about a node's energy availability, the receiving node will find out by comparing the battery power remaining for that node from the table. The battery value of a node will never increase; even the sleeping nodes will consume some battery power, so this type of attack can be completely eliminated by using our technique.

Dest (2)	AM (1)	Len (1)	Src (2)	Ctr (2)	Power (1)	Data (0-29)	MAC (4)

Table 2: Packet with power information

If the adversary announces the availability of high quality link to a neighbor, the receiving node will find out by comparing the table value with the received signal strength value. Note that each node keeps track of the RSSI value of all its neighbors. The RSSI value is expected to decrease with the decrease in remaining battery power. So any increase in the RSSI value from a node is not possible. By comparing the value of signal strength from a node and the value of RSSI exist in the table, a node can differentiate the

authentic and fake message. The node can then send the information to the base station for key revocation.

The remaining power of each node is indirectly proportional to the data rate. If the data rate requested by a neighbor is high, then the battery power of the node will reduce at a faster rate than the node which requested minimum data rate. Note that it is not true that if the data rate of one neighbor is low, then the battery power of the node will reduce slowly, because the neighbor node might be sending data at a higher to some other neighbor. Any announcement of having more power from the high data rate nodes can be compared by the received neighbor. For example if the data rate is 100 events / second, then the nodes on the data flow path should be receiving and sending 100 messages (except the source and base station where source only transmits and the base station only receives). This consumes more battery power. If the node on the data flow path calculates the remaining battery power of a neighbor based on the data rate, then the fake messages can be found out. Note that this calculation is necessary only when a neighbor sends an announcement regarding the battery power.

There is one problem with the above approach, knowing the battery power of the immediate neighbor alone is not enough. An adversary might try to send an announcement using other nodes id. For example, in the figure 9 let us consider that the adversary knows the pair-wise key used between E and B. The attacker might send a message about the power availability to node B faking its identity as E. In the message content, the adversary might announce that node E received the information from node D.

If such a message was sent, then the node B cannot find out whether the message received is a fake or authentic. Our above solution won't work for this condition, so we are introducing a new technique to prevent this attack.

If each node has the knowledge of the remaining battery power of all other nodes in the network, then the comparisons can be made for the farther nodes too. If nodes in the network share the information regarding the battery power and RSSI of their neighbors then this type of attack can be prevented. While forwarding the data message each node has to attach the remaining battery power value of all its neighbors in the packet. On receiving this message, each node stores the information. Note that no additional transmission and receiving is necessary in this case. The overhead of this approach is the need for extra memory space in each node to store the information about two hop neighbor nodes information.

5.5.2 Calculation of Remaining Energy

The ADC7 of the Mica mote gives the battery voltage [13]. For our simulation we can use the two components provided by nesC language to calculate the remaining power, computeRates(..) and PowerMonQuery. Using the data sheets provided for the MICA mote, power required for all the operations can be calculated.

5.6 Selective Forwarding

The selective forwarding is a type of attack in which the attacker, after receiving the data messages from the upstream neighbor does not forward all the messages. The

adversary can modify the data message or inject its own data message to the downstream nodes. In the figure below, let us consider that the data flows from A to B to C. If an attacker breaks the pair-wise key shared between B and A, then the adversary can modify the data message and then send to B. B further forwards the data message to the base station.

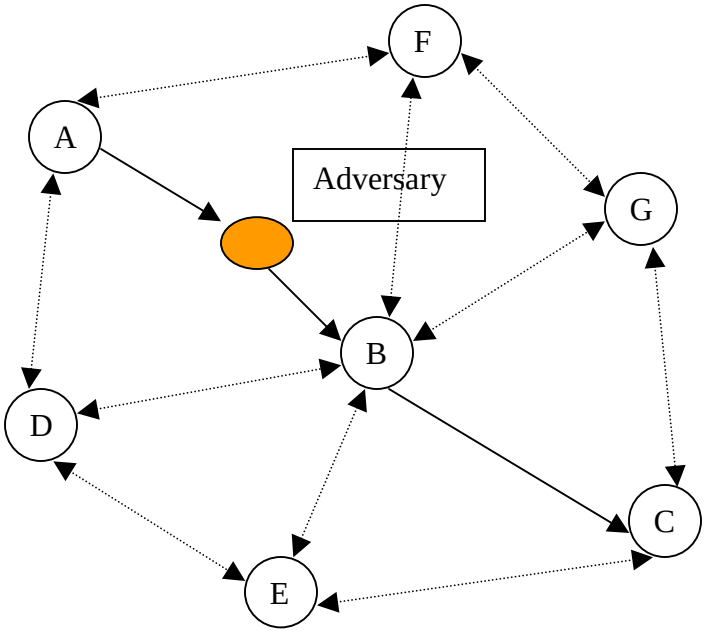

Figure 11: Selective Forwarding

5.6.1 Proposed solution

In directed diffusion each node has a gradient towards each of its neighbor's. Some nodes receive data messages at higher rate and others receive data message at the initial data rate. The important thing to remember here is the availability of alternate paths. The data messages received in the alternate paths can be used to find out whether the nodes in the higher data flow path are acting normally or not.

In the figure 11 the dashed lines represent the low data rate paths. The data rate of path A->B->C is different from other paths (in our example it is 100 events /second); the low data rate paths receive data at 1 event/second. Let us assume that the adversary breaks the pair-wise shared key used between A and B. Now the adversary gets the access to the data messages from the source node. It can modify the data messages and send to B or selectively forward it to B. After receiving the modified packet the node B forwards the packet to the base station. Node B also sends the data messages to neighbors D, E, F, and G at initial data rate. The source node A also sends the data messages to its neighbors other than B (F and D). The data messages received from the source are stored in the data cache of each node (a node drops a data message if that data message was already seen by the node). So node D will have data message from source A and node B in it data cache. By implementing a simple check mechanism to compare the two data messages, we can find out the selective forwarding attack. Note that the same comparison will be performed in all the neighbors of the node B.

The data message comparison is application dependent. Let us take an example to explain this. Consider the chemical lab monitoring application. In this application the base station might spread interest information about the possible gas leak at specific location. In this application the base station expects either “YES” or “NO” as the message from the source node. Let us assume that maximum number of events from the source node contains “NO” and the rest of the events are “YES”. In this case the source node sends a “YES” to the nodes with the low data rate (Even if there is only one “YES”). The important thing to remember is the low data rate paths receive the

approximation of the 100 events, not the 100th event. If the adversary modifies the "YES" data message from the source to a "NO" and forwards it, then the nodes can find out the modification. Different techniques are used in other applications to find out whether the data is same or not. The confidence rate can be used for matching two events. The source node compares the stored wave form of an event with the sensed event and finds out the confidence value.

In section 2.6.5, we discussed about the data cache. In the data cache each node maintains the last seen event (data message) from the neighbor. If a data comparison method is developed to check the data message received from a node and the data message present in the data cache, a node can also find out the fake data message. But each application needs a new data comparison technique. The basic idea is that there will be a relation between the two events of the same interest.

5.7 Data Fusion

Our above solution was based on the assumption that the data messages derived by the source node are authentic. If the source node was compromised (by breaking all the neighbor's keys) then the adversary might derive its own data messages. The probability to break all the pair-wise keys of a node is very less, but if that ever happens then the whole purpose of setting up the sensor nodes will become useless. More over there is no way of finding whether the source node is compromised or not. To solve this problem Wenliang Du and Jing Deng proposed **a Witness-based approach for data fusion assurance in wireless sensor networks** [13]. In that paper, not only the source node

activates it's sensors to collect data, even some of its neighbors (chosen probabilistically) also collects the data. Note that not all the neighbors of a node can detect the event detected by the source node since some nodes may lie outside the radio range. The nodes which are collecting the events are called as witness nodes. These witness nodes does not forward the data like source node, it simply collects and stores the data. The base station will query these nodes for verifying the data send by the source node. In [13], m out of n witness nodes is queried by the base station for verification. The witness nodes forward the encrypted message to the nodes and they further forward till the verification message reaches the base station. Using this technique the problem of compromised source node can be solved. By implementing this approach, the directed diffusion algorithm can be improved. The ideas proposed in [13] are based on the assumption that some of the neighbors of the source node can sense the same events with equal lesser confidence value.

5.7.1 Single Source Problem

Since the nodes are spread in the area of interest from a plane, there is a chance that some nodes do not have more than one neighbor. If that node becomes a source node and serves the data to the base station, the above said witness nodes method cannot be implemented. The adversary has a very good chance of breaking the key, and sending the false data to the base station. In order to prevent this type of attack we propose the implementation of concepts discussed in [19].

5.7.2 Mobile Beacon

The paper [19] discusses how one mobile node can be used to localize the network. The mobile node which contains the GPS equipment traverses through the entire network to find and set the GPS location for each node. The mobile beacon can be a human operator, an unmanned vehicle deployed with the sensor network, or in the case of a deployment from a plane, the plane itself. The method presented in the paper [19] is radio-frequency (RF) based: the received signal strength indicator (RSSI) is used for ranging. The advantage of the RSSI ranging is its ubiquitous availability in practically all available receivers on the market.

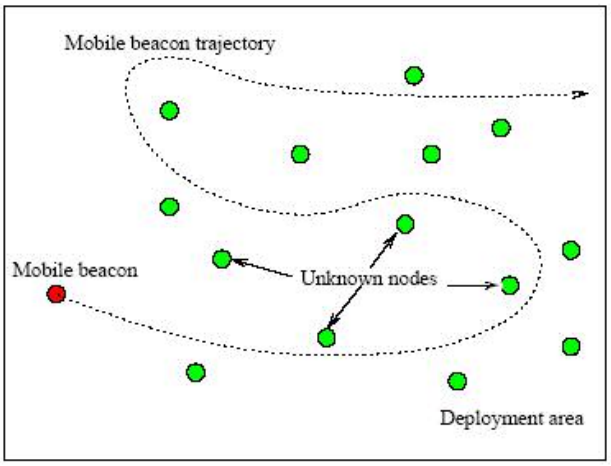

Figure 12: One mobile beacon assisting in the localization of a sensor field.

Figure 12 depicts a sensor network deployed over a geographical area. After deployment, a mobile beacon traverses the sensor network while broadcasting beacon packets. A beacon packet contains the coordinates of the beacon. Any node receiving the beacon packet will be able to infer that it must be somewhere around the mobile beacon with a certain probability. This information constrains the possible locations of a node.

The RSSI is measured for each beacon that is received. Corresponding to the RSSI measurement and the position of the beacon (*xB; yB*) (included in the beacon packet), each node receiving the beacon constructs a constraint on its position estimate:

Constraint(*x; y*) = *PDFRSSI* (*d* ((*x, y*), (*xB, yB*)) (*x, y*) *[*(*x*min, *x*max) * (*y*min, *ymax*)] (1)

where *PDFRSSI* is the probability distribution function of the distance corresponding to the RSSI of the beacon packet, *d*(*A;B*) is the Euclidean distance between points *A* and *B*, and *x*min*; x*max*; y*min and *y*max are the bounding coordinates of the deployment area.

5.7.3 Proposed Solution

If there is only one node delivering the sensed values to the base station then, the base station can instruct the mobile node to reach the region where the single source is situated. Now the mobile node also can sense the events. The mobile node now acts as the witness node and collects the data and does not forward the data like the source node. The base station sends the query to the mobile node at random time interval to verify the source node's data messages.

5.7.4 Implementation Issues

When the mobile node enters the network, it has to set up the pair-wise keys with the nodes which are currently its neighbors so that it can communicate with the nodes. This is similar to the node addition and leaving discussed in the LEAP algorithm. The mobile node has to carry all the m Master keys, where m is number of node addition events. With the addition of new nodes the security of the nodes increases.

5.8 Effects of Node Inclusion and Exclusion

5.8.1 Node Addition

If a new node has to be added in the network then it should be loaded with the global key , the master key K_i and $K_j(u)$ individual keys where j is 1< j <=m. The procedure for calculating the individual key of the node is same as discussed above. The base station can calculate the individual key of the newly added node (since it has the master key), using the individual key of the added node, the base station unicasts the "Hello" packet. The added node receives this message and calculates the signal strength value for the base station and stores it (so that it can authenticate any future message from base station).

The new node added in the network will broadcast the "Hello" packet with its id specified in the packet. The node which receives this packet replies by broadcasting "Hello" packets. Let us consider that the nodes u_1 to u_N are present in the network and node v is the new node. Now node v has to set up pair-wise key with all its neighbors' u_1 to u_M where M <=n. Since the nodes u_1 to u_M already have the individual key of v, each can calculate the pair-wise shared key with node v. The procedure is same as the initial pair-wise key set-up of LEAP. The nodes u_1 to u_M add node v in their neighbor list.

In directed diffusion algorithm, the problem of adding a new node in the network is not addressed. This leads to path loss and excessive delay in the network once the nodes lose their power. LEAP provides a mechanism to add the nodes in the network and establish pair-wise shared keys with existing nodes (discussed above). But after addition,

the added node needs to know the interest information to build the gradients in the network. This problem can be solved if the added node's neighbors pass the interest information to the new node. This will cause the added node v to receive the data messages from its neighbors. Note that the node v also builds the table for storing the RSSI and the battery power remaining of its neighbors from the message it received from them while setting up the pair-wise keys.

The addition of nodes in the network gives more security. Since the solutions proposed by us depend on the assumption that the sensor network is dense, addition of more nodes means the adversary has more nodes to compromise.

5.8.2 Node Leaving

The directed diffusion addresses the problem of node failure in the network. It uses the negative reinforcement to truncate a path if a node does not respond with in a specified amount of time. Each node will maintain the remaining battery power information of all its neighbors. When the battery power of a node reaches a threshold level, then the neighbors of the node removes the id of the node from their list. This will cause the base station to select another path to get the data from the source node. The four type of attack discussed above depends on the assumption that there will be atleast two neighbors for each node (with enough power). The above discussed solution for node leaving will still work if the assumption is true. Let us take an example for each type of attack and prove that this is true.

5.8.2.1 Negative Reinforcement

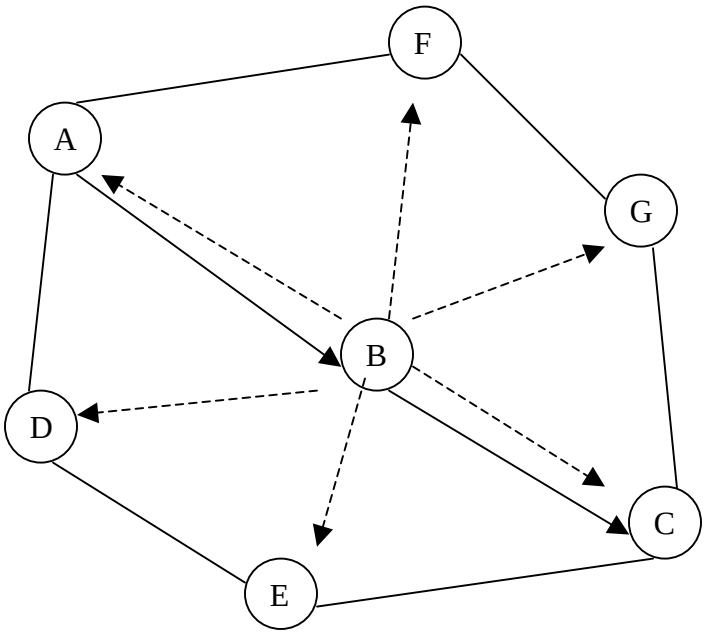

Figure 13: Negative Reinforcement

Let us consider that the node C is the base station and node is the source. Let us also consider that the link AB is congested and B wants to send a negative reinforcement to A, to do this node B has to send a negative reinforcement to A and the negative reinforcement information to all its neighbors. Even if one or more nodes did not have enough power to forward the information to the source node, the availability of multiple paths ensures that atleast one other neighbor gives the information to the source. In the figure if both D and F fail at the same time then the negative reinforcement information does not reach the source node, but then our assumption that each node will have atleast two nodes with enough power to perform operation will not be satisfied.

5.8.2.2 Selective Forwarding

The selective forwarding attack will be still found if our assumption that each node will have atleast two neighbors. The availability of multiple nodes to check the data from the source node ensures that the nodes in the data flow path sends the original data received from the source node to the base station.

5.8.2.3 Path Influence and Cloning Attack

The solutions for these two attacks are not affected by the number of neighbor nodes. Since the solutions for these attacks is not dependent on other nodes. If a forged message is sent to a node then it will compare the value with the value present in the table. Thus the solutions for these two attacks are not affected by the loss of one or more nodes.

CHAPTER VI

PERFORMANCE AND SECURITY ANALYSIS

6.1 Storage Requirement

In this section we will see the overheads involved if we implement the LEAP algorithm and our proposed methods in the existing directed diffusion algorithm. In our scheme, each node has to store an individual key, a group key and *d* pair-wise keys, where d is the number of one-hop neighbors. The number of bytes required depends on the density of the sensor networks. The total number of keys a node has to store is d+2 (d is the one hop neighbors). Each key is 8 bytes in length [1]. So the total bytes required for a node will be, (d+2)*8 bytes. Although memory is a very scarce resource for the current generation of sensor nodes (4 KB SRAM in a Berkeley Mica Mote) [2 leap], for a reasonable degree *d*, storage is not an issue in our scheme.

The nodes in the network need to store the RSSI value and the remaining battery value of all its one and two-hop neighbors. The RSSI and the remaining battery power take a byte each to store. So, the total bytes required for a node becomes, (d+2)*8 + (D + d)*2 bytes. Here D is the 2^{nd} hop neighbor and d is the one-hop neighbor.

Since there is no cluster key in our implementation, considerable amount of storage space in each node is saved when compared to LEAP (d per node). In LEAP each node has to have a cluster key. The cluster head of a cluster has to store the keys of all other cluster heads in the network. With the introduction of multiple Master keys the storage requirement increases. Depending upon the number of node additions, the memory required to store the keys increases.

6.2 Computational Cost

The pair-wise key and individual key set-up is done after the deployment. Each node has to calculate d pair-wise keys and one individual key. If there are N number of nodes in the sensor network then the number of key calculations will be,

$$\sum d_i + N \text{ where } i = 1 \text{ to } N.$$

The number of encryption and decryption is an important factor in determining the efficiency of our scheme. The base station encrypts the interest message using the global key and broadcast the message. The nodes after receiving the interest message, decrypts and stores the interest in the interest cache. This is followed by encrypting the message using the global key and broadcasting further. Each node will broadcast the interest message and neighbors decrypt the message. So if there are N numbers of nodes, then the number of encryption and decryption will be,

$$\sum d_i - S_j \text{ where S is the source node and } j=1\ldots M.\ M<=N \text{ and } i=1 \text{ to } N$$

Besides the pair-wise key and individual key, each node also has to calculate the individual key of all its neighbors for verification. So the equation becomes,

$\sum 2d_i - S_j$ where S is the source node and j=1…M. M<=N and i=1 to N

The computation cost to establish the keys in the network will be less for our proposed version when we compare with LEAP. Nodes don't have to calculate the cluster keys in our case. Our case requires some additional comparison for keeping the network secure. All the packets from the base station will be compared with the stored value of battery power remaining and RSSI value. This cost goes up when the number of interest message goes up.

6.3 Communication Cost

Since the interests are broadcasted, each node has to transmit only once (15 mA [4]). But each node will receive the same interest from all of its neighbors (d*12 mA). The same applies for the data messages. So with the number of interests the communication cost increases. For I interest, the transmission and receiving cost for broadcasting will be,

I*summation of d_j + I where j =1 to N.

For receiving and transmitting data messages, the communication cost will depend on the number of paths reinforced positively and the data rate requested by the downstream nodes.

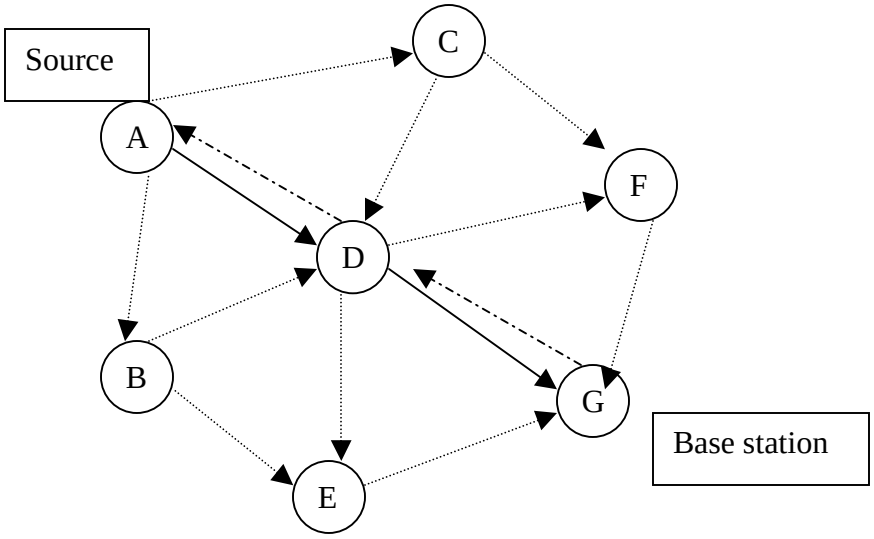

Figure: 14: Communication Cost

In the above figure, let A be the source node and G be the base station. Let us also assume that the base station positively reinforces node D. Node D reinforces node A. All the other nodes in the network receive the data at the initial data rate. If the data rate is 100 events per second in the data flow path (A->D->G) and 1 event per second in all other paths then the number of transmission and receiving will be,

(2*100)*15 mA+(2*100)*12 mA+9*15 mA+9* 12 mA.

Using the above we can derive a generic equation for the power usage.

Let d number of neighbors of a node which is in the non data flow path and D be the number of nodes involved in the data flow path. The power required in the MAX data time will be,

((D-1)*MAX_DATA_RATE)*15 mA (for transmission) +

(D-1)*MAX_DATA_RATE)*12 mA (for receiving) +

$\sum d_i$ *15 mA (non data flow nodes transmission) +

$\sum d_i$ *12 mA (non data flow path receiving).

The communication cost for sending the queries and receiving the data message is same in our scheme and directed diffusion. Only when a node wants to send a negative reinforcement, it has to multicast the information about sending the negative reinforcement to all other neighbors. But a negative reinforcement message is not expected often in the network, so the overhead occurred by a few negative reinforcement can be compromised for increased security.

6.4 Packet Size

The following is the packet used by the TinySec, the keying mechanism recently proposed for the sensor networks. The packet format of TinyOS is also given for comparison purpose. For our purpose we will use the TinySec packet format with additional information like GPS quordinates and the remaining battery power.

Dest	AM	Len	Grp	Data	CRC
(2)	(1)	(1)	(1)	(0-29)	(2)

Table 3.TinyOS packet format

est (2)	AM (1)	Len (1)	Src (2)	Ctr (2)	Data (0-29)	MAC (4)

Table 4: TinySec packet format

The nodes in our network have to include the remaining battery power. So the packet format will be like the following,

Dest (2)	AM (1)	Len (1)	Src (2)	Ctr (2)	Power (1)	Data (0-29)	MAC (4)

Table 5:Packet with power information.

The TinyOS packet format in the figure 1 does not have the source node information; this leaves the entire network vulnerable to the outsider attack. Any node can inject a packet with little effort. To detect transmission errors, TinyOS senders compute a 16-bit cycle redundancy check (CRC) over the packet. The receiver recomputes the CRC during reception and verifies it with the received CRC field. If they are equal, the receiver accepts the packet and rejects it otherwise [1- TinySec]. But this CRC does not provide any security from attacks. Since we have included the MAC field, CRC is not needed. Active message types are similar to port numbers in TCP/IP. The AM type specifies the appropriate handler function, to extract and interpret the message on the receiver. The TinyOS packet format contains a group field to prevent different sensor

networks from interfering with each other. It can be thought of as a kind of weak access control mechanism for non-malicious environments [1- TinySec]. The Ctr field is used for specifying the packet numbers. Our implementation needs one extra byte from the TinySec packet format (not from the directed diffusion).

6.5 Security Analysis

Though our algorithm requires more memory space and more energy for computation, it defends the sensor nodes against the possible attacks. With the introduction of LEAP algorithm, the outsider attack has been completely eliminated. The LEAP algorithm prevents the sinkhole and wormhole attacks. The solution presented by us will provide more security to the network even if some of the nodes in the network are compromised. We have introduced the use of mobile nodes so that there is always more than one source node available for the base station to verify the results.

CHAPTER VII

SIMULATION AND RESULTS

7.1 Simulation Set-up

C++ code was written to simulate the sensor networks of sizes 10, 30 and 50. The performance of our proposed algorithm relies on the density of the network. The x and y coordinate for each node was generated randomly and stored in a C++ structure variable. The node with id 1 is assumed as the base station and the last node is set as the source node. The distance between every pair of nodes in the network was found in order to establish multiple paths between the base station and the source nodes. The random placement of the nodes in the constant area is necessary since our proposed algorithm's performance depends on the density of the nodes.

7.2 Storage Requirement

The original directed diffusion algorithm does not use any keying mechanism therefore the memory required in each node is very minimal. In our algorithm each node has to store one individual key, one group key and d number of pair-wise keys, where d is the is the number of neighbors of a node. The following graph was plotted for the storage requirement for the sensor network of size 10, 30 and 50. The nodes were placed at random location and the simulation was run for 30 times each for 10, 30 and 50 nodes. From the figure 15, we can infer that the memory require to store the keys increases with

the increase in the number of nodes. The table 6 shows the result of 30 simulations. The result for 30 nodes varies from 6594 to 4650; the reason for this variation is because of the varying number of neighbors for the nodes. Each node has to store an individual key, a group key, *d* pair-wise keys, where d is the number of one-hop neighbors and the RSSI value and remaining battery power of one-hop and two-hop neighbors.

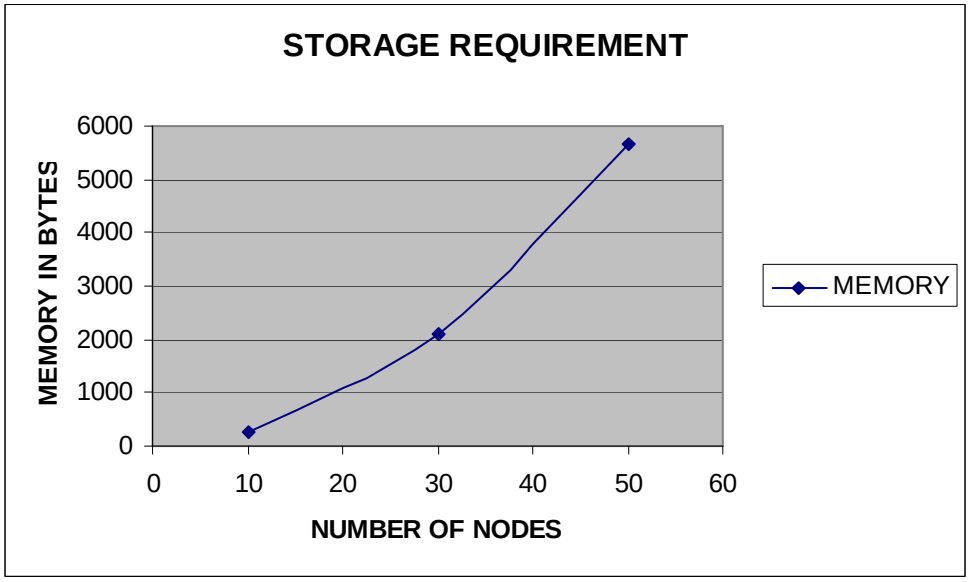

Figure: 15 Memory Requirements

50	30	10
6422	2190	354
5494	2318	328
5538	1990	126
5374	1734	296
6010	1734	226
6010	2340	254
5714	2106	386
6282	1618	252
5750	1618	182
5750	2844	294
4650	2844	294
4650	1712	238

5822	2062	268
5826	2288	268
5388	2288	340
5388	1722	350
6102	1722	310
5932	2458	210
5096	2238	210
5096	2238	370
6522	2228	218
5498	2146	328
5642	2146	328
5642	2178	306
5698	2178	168
5538	1966	208
5538	1796	208
6594	1796	298
5634	2038	296
5754	2070	296
170354	62606	8210
5678.467	2086.867	273.6667

Table 6: Simulation Result for 10, 30 and 50 nodes

7.3 Communication Cost

In our proposed algorithm each node has to find and store the id's of all the neighbor nodes before communicating with them. The initial set-up of sensor network is different in our algorithm then the original directed diffusion. In order to establish the pair-wise key with all its neighbors, each node has to send a "hello" packet. The other communication overhead introduced in our algorithm is the "hello" packet sent by the base station to all the nodes in the network so that the nodes in the network calculate and store the base station's RSSI value. The following graph shows the communication overhead caused by our algorithm and the original communication cost of the directed diffusion algorithm.

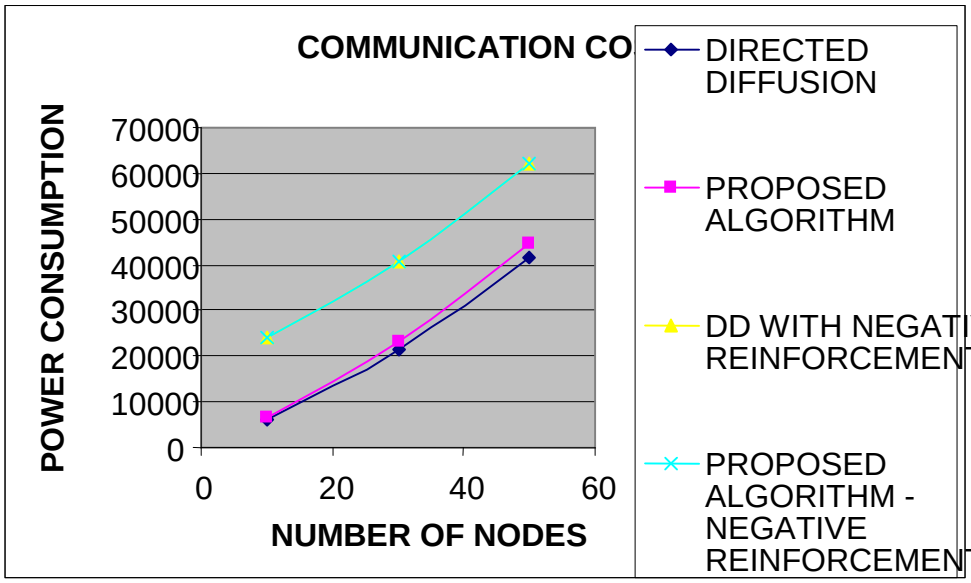

Figure 16: Power Consumption

Form the figure 16 we can see that the difference between the directed diffusion and our proposed algorithm increases with the increase in the network size. The above graph also shows the overhead caused by our proposed solution for the negative reinforcement. Even if there is an authentic negative reinforcement, there will be communication and computation overheads in our proposed algorithm. If a node sends a negative reinforcement to an upstream neighbor, then it has to send the negative reinforcement information to all its neighbors, which is not done in the directed diffusion algorithm. As we can see from the above graph, the increase in the network size and the density causes more transmission and reception in the network. Thus the overhead increases with the network size.

10	6033	6422	24163	24257
30	21339	23056	40528	40620
50	41360	44730	62112	62204

Table 7: Simulation result for Communication Cost

7.4 Computational Cost

Our solutions to all the four type of attacks require some computational power. Our proposed solution to the selective forwarding algorithm causes more overhead, since all the packets forwarded in the high data flow path has to be compared by all other nodes. The solutions to other three types of attacks require the nodes to compare only when the need arises whereas in case of selective forwarding, the nodes have to compare all the time the interest is active. Besides the comparison overheads, the other computational energy required is for the encryption and decryption of the messages transmitted in the network.

7.4.1 Selective Forwarding

When the nodes in the high data flow path sends the data to the base station, the low data flow path nodes have to compare the data forwarded by the nodes in the high data flow nodes. The computation overhead for implementing selective forwarding depends upon the number of nodes in the low data flow path. The following graph shows that the power consumption increases with the number of nods.

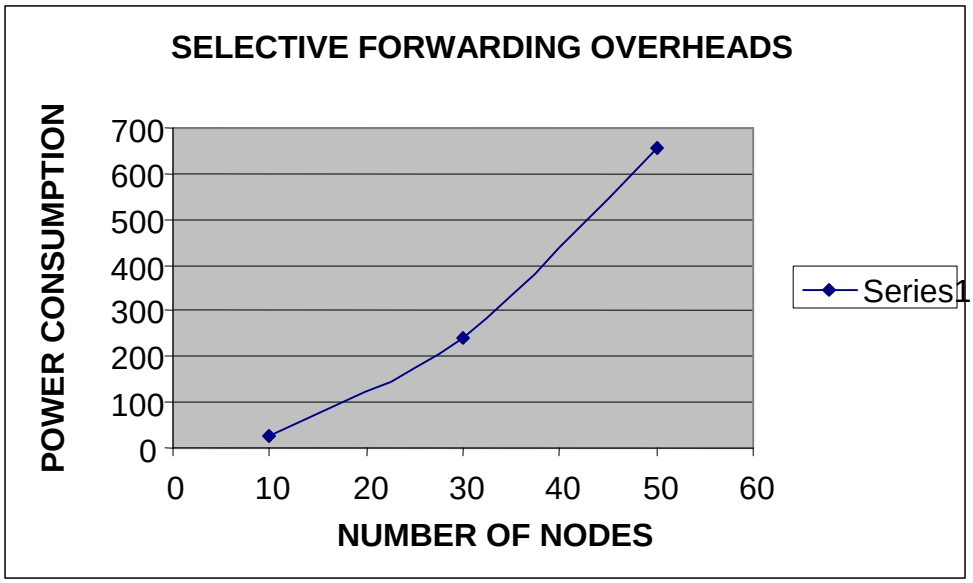

Figure 17: Selective forwarding overhead

10	30	50
55	224	763
2	200	695
24	235	596
28	308	424
13	222	623
13	277	629
39	321	737
28	206	445
41	272	645
28	286	717
41	246	576
24	239	746
41	281	706
30	213	710
17	190	715
17	112	547
35	184	787
17	297	746
30	162	649
30	211	649
15	187	719
35	294	598
13	310	646
30	213	745
30	213	701
24	294	690

28	248	657
28	270	616
37	213	598
37	257	598
830	7185	19673
27.66667	239.5	655.7667

Table 8: Simulation result for Selective Forwarding

7.4.2 Negative Reinforcement

When the base station or any node in the network send a negative reinforcement, the upstream node which receives this negative reinforcement, process only after comparing the same information from all its neighbors. The node which received the negative reinforcement information will forward the negative reinforcement to the upstream node only after receiving the negative reinforcement information atleast from two neighbors. The computation overhead caused by this type of attack is minimal and also this overhead occurs only when there is a negative reinforcement passed by some nodes in the network. The negative reinforcement is not expected frequently in a sensor network.

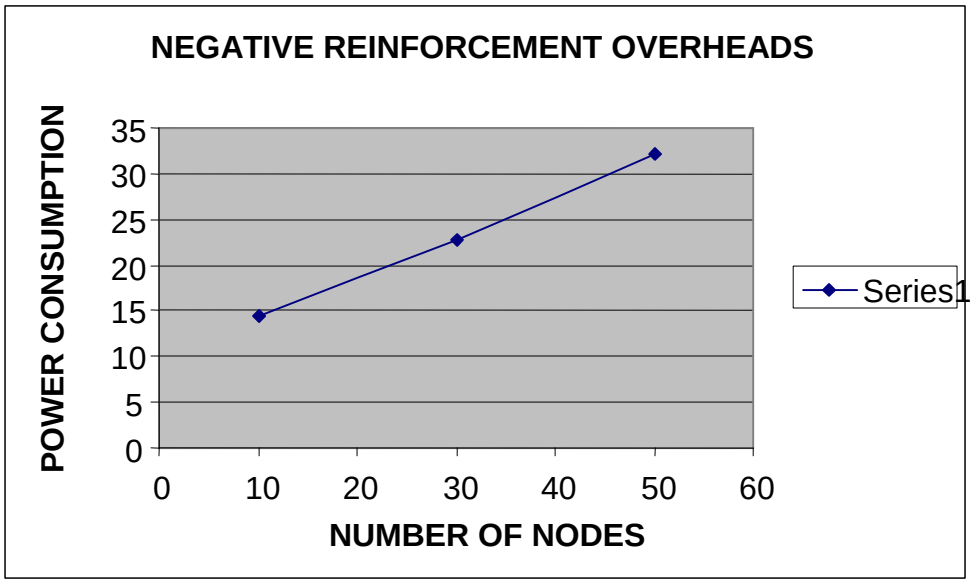

Figure 18: Negative Reinforcement overhead

CHAPTER VIII

CONCLUSIONS AND FUTURE WORK

Our simulation results show that the storage space required increases with the increase in the number of nodes in the network. But when compared to the LEAP algorithm, our algorithm requires less memory space since our algorithm uses only three types of keys. In the case of negative reinforcement, our proposed algorithm differs from directed diffusion only by few transmissions. The density of the network determines the storage space required for each node: if there are more neighbors for a node then the node has to store more keys, thus increasing the memory requirement. Our proposed algorithm can be used for applications which require message authentication and message confidentiality. We have assumed that the network is static; our proposed algorithm can be improved so that it handles the mobile nodes too. Each packet contains the information about remaining battery power information and RSSI value of a node. This approach can be modified so that the packet size is reduced. In directed diffusion a node transmits the interest information even to the node which originally sent it. This work can be extended by analyzing the duplicate interest problem and providing a solution. The packet size is also an important factor in determining the efficiency of our algorithm. Further work can be done to reduce the size of the packet.

REFERENCES

[1]
Karlof, C. and D. Wagner (2003). "Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures." Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols Volume: **1,** Issue 1, Page: 293--315.

[2]
Asada, G., T. Dong, et al. (1998). "Wireless integrated network sensors: Low power systems on a chip." 24th IEEE European Solid-State Circuits Conference Volume: 9, Issue: 2, Page: 9-18.

[3]
Hill, J., R. Szewczyk, et al. (2000). "System Architecture Directions for Networked Sensors." Proceedings of the ninth international conference on Architectural support for programming languages and operating systems Volume: **34,** Issue: 5, Page: 93-104.

[4]
TinyOS Documentation http://tinyos.net/tinyos-1.x/doc/

[5]
Intanagonwiwat, C., R. Govindan, et al. (2003). "Directed Diffusion for Wireless Sensor Networking." Proceedings of the 1st ACM international workshop on wireless sensor networks and applications Volume: **11**, Issue: 1, Page: 2-16.

[6]
Perrig, A., R. Szewczyk, et al. (2001). "SPINS: Security Protocols for Sensor Networks." Proceedings of Seventh Annual International Conference on Mobile Computing and Networks MOBICOM 2001: Volume 4, Issue 7, Page: 189-199.

[7]
Karlof, C., N. Sastry, et al. (2004). "TinySec: A Link Layer Security Architecture for Wireless Sensor Networks." Proceedings of the 2nd ACM International workshop on wireless sensor networks and applications Volume: **4,** Issue: 1, Page: 22-29.

[8]
Zhu, S., S. Setia, et al. (2003). "LEAP: efficient security mechanisms for large-scale distributed sensor networks." Proceedings of the 2nd ACM International workshop on wireless sensor networks and applications Volume 2, Issue 3, Page: 62-72.

[9]
Hightower, J., R. Want, et al. (2000). "SpotON: An Indoor 3D Location Sensing Technology Based on RF Signal Strength." Proceedings of Sixth Annual International Conference on Mobile Computing and Network, CSE 2000-02-02, University of Washington Volume: **1,** Issue: 1, Page: 1-13.

[10]
Pethe, A. G., G. Krishnakumar, et al. (2004). "Rank Based Localization Protocol for Sensor Networks." Proceedings of 8th Annual International Conference on Mobile Computing and Network, Michigan Technological University: Page: 1-18.

[11]
Langendoen, K. and N. Reijers (2003). "Distributed localization in wireless sensor networks: a quantitative comparison." Computer Networks: The International Journal of Computer and Telecommunications Networking Volume: **43,** Issue: 4, Page: 499-518.

[12]
Whitehouse, K. and D. Culler (2002). "Calibration as Parameter Estimation in Sensor Networks." Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications Volume: 4, Issue 3, Page: 59-67.

[13]
Crossbow Technology www.xbow.com

[14]
Zhang, T., S. Pande, et al. (2002). "Tamper Resistance a Cautionary Note." Proceedings of the 2003 ACM SIGPLAN conference on Language, compiler, and tool for embedded systems Volume 6, Issue 3, Page: 209-219.

[15]
Park, S., A. Savvides, et al. (2001). "Simulating networks of wireless sensors." Proceedings of the 2001 Winter Simulation Conference. Volume: 4, Issue: 1, Page: 44-61.

[16]
Christopher and Y.-H. Chau (2003). "Inexpensive Wireless telemetry for animal tracking." A Design Project Report Presented to the Engineering Division of the Graduate School Of Cornell University.

[17]
Pathirana, P. N., A. V. Savkin, et al. (2004). "Node localization using mobile robots in delay-tolerant sensor networks." IEEE Transactions on Mobile Computing Volume: **4,** Issue: 1, Page: 1-13.

[18]
Hightower, J., C. Vakili, et al. (2001). "Design and calibration of the spot on ad-hoc location sensing system." Proceedings of the 10th annual international conference on mobile computing and networking Volume:**7,** Issue: 1, Page: 1-18.

[19]
Ramadurai, M. L. S. a. V. (2004). "Localization of Wireless Sensor Networks with a Mobile Beacon." IEEE Convention of Electrical and Electronics Engineers. Volume: **11,** Issue: 2, Page: 17-31.

[20]
Welsh, V.S.M.H.B.-r.C.G.W.A.M. (2004), “Simulating the power consumption of large-scale sensor network applications”. Proceedings of the 2nd international conference on Embedded networked sensor systems,. Volume:**1 ,** Issue: 6, Page: 188-200.

APPENDICES

VITA

Rajkumar Vijayaraman

Candidate for the Degree of

Master of Science

Thesis: DIRECTED DIFFUSION ROUTING PROTOCOL: ATTACKS AND COUNTERMEASURES

Major Field: Computer Science

Biographical:

Personal Data: Born in Tirupathur, Tamil Nadu, India on January 28, 1979, the son of N. Vijayaraman and Mrs. Saroja Vijayaraman.

Education: Received Bachelor of Engineering in computer Science and Engineering from Annamalai University, Chidambaram, India in May 2001. Completed the requirements for the Master of Science degree with a major in Computer Science at Oklahoma State University in May, 2005.

Professional Experience: August 2004 – December 2004: ColdFusion developer, Dr. Donald French, Zoology Department, Oklahoma State University, Stillwater.