A LIGHTWEIGHT KEY DISTRIBUTION MECHANISM FOR WIRELESS SENSOR NETWORKS

By

LYNSEY ELIZABETH COMPTON-DRAKE

A thesis submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER ENGINEERING

WASHINGTON STATE UNIVERSITY
School of Electrical Engineering and Computer Science
May 2009

To the Faculty of Washington State University

The members of the Committee appointed to examine the thesis of STACEY MARIE COBB find it satisfactory and recommend that it be accepted.

Sirisha Medidi, Chair

Muralidhar Medidi

Jack R Hagemeister

ACKNOWLEDGEMENT

I would like to thank the members of my committee for their guidance and support, and the SWAN lab research group for their knowledge and assistance in all things related to ns-2. Additionally, I would like to thank Dr. Tracy Camp, Brad Williams and Brian Barritt for their work in implementing SBA for the ns-2 simulator and their willingness to share.

A LIGHTWEIGHT KEY DISTRIBUTION MECHANISM FOR WIRELESS SENSOR NETWORKS

Abstract

by Lynsey Elizabeth Compton-Drake, M.S.
Washington State University
May 2009

Chair: Sirisha Medidi

Security is of critical importance for many potential applications of wireless sensor networks. In order to maintain secure communication throughout the network, it is of vital importance to maintain encryption key freshness by regularly distributing new keys to all nodes. Distribution of group keys used to encrypt broadcast communication is expensive, as it is generally achieved via flooding, which taxes the limited battery life available to each node. We propose LKDT, a lightweight encryption key distribution tree building mechanism to provide a framework by which to distribute keys while reducing power consumption and broadcast coverage overlap. LKDT ensures reliable key distribution by using negative acknowledgment based feedback when a key does not reach its destination. Additionally, LKDT can configure itself quickly, allowing the base station to begin updating keys shortly after deployment.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Dedication

This thesis is dedicated to my parents, Rob, Jack, and Chris.

Without your help and support I could not have achieved so much.

CHAPTER ONE

INTRODUCTION

Over the past decades, advances in electronics miniaturization and wireless data transmission have given rise to wireless sensor networks (WSNs). These networks are comprised of hundreds or thousands of small, battery operated sensor platforms which collectively monitor for changes in the area in which they are deployed. With the wide variety of sensor types available (e.g. light, temperature, motion, sound), WSNs are suited to a vast array of applications. Wildlife habitat monitoring, wildfire detection, battlefield surveillance, and border monitoring are just a sampling of the potential uses. In many applications, especially those relating to military and homeland security uses, sensed data may be used to prevent injury and save lives. This requires that the WSN be secured to prevent reading or altering of the data as it is transmitted.

Wireless security is a vast and complicated area of research. There have been many different approaches to improving security [1, 2, 3]

In [2], Zhu *et al* identify the four types of communication that occur and therefor keys that are needed in WSNs: individual, pair-wise, cluster, and group. Individual keys are used to secure communication between the base station and a single node. Similarly, pair-wise keys are needed for communication between two non-base station nodes. Cluster keys are used to communicate with all of a node's one-hop neighbors and group keys secure data being broadcast to all nodes within the network. We focus on the physical distribution of group keys, which we will refer to as broadcast keys.

To date, there has been no work done in the area of broadcast key distribution. Distribution of broadcast keys is a broadcast distribution problem. Significant work has been done in this area [4, 5, 6, 7, 8, 9], and though many proposed solutions show significant improvements in power consumption and network coverage, no guarantee of delivery is required. This is not acceptable for key distribution where a missed key could potentially compromise the security of the network.

Transmissions from the base station to all other nodes in the network are infrequent when compared to traffic originating at the nodes (sensed data). With this in mind, we need a mechanism to distribute base station broadcasts which reduces power consumption thereby extending network lifetime, provides reliable delivery of data, can be configured quickly so as to allow for re-keying soon after network turn-on, and has low overhead in terms of construction costs.

We propose the use of a Light-weight Key Distribution Tree (LKDT) to control the dissemination of group encryption keys in WSNs. LKDT is based upon the fusion of a maximal leaf tree and a minimum power broadcast tree. LKDT is a fully distributed mechanism in which nodes use only one- and two-hop neighborhood information to self-identify themselves as *leaf* or *forwarder* nodes. Only forwarder nodes are permitted to broadcast encryption key packets. LKDT seeks to strike a balance between minimizing the number of forwarders and ensuring that all nodes in the network receive each key. To accomplish this we use a simple scoring system to compare node coverage areas.

This thesis is organized as follows. Chapter 2 reviews background and related work. In Chapter 3 we describe our key distribution mechanism. Chapter 4 details the performance analysis of LKDT. In Chapter 5 we conclude with a brief review and thoughts on future work related to LKDT and encryption key distribution management.

CHAPTER TWO

BACKGROUND AND RELATED WORK

2.1 Broadcast Distribution

Broadcast distribution refers to the process of delivering a message to all nodes in the network. Primary goal underlying most broadcast distribution mechanisms is the desire for energy efficiency to maximize network lifetime. This is accomplished by controlling the set of nodes which will transmit the message. There have been many broadcast distribution protocols proposed; they can be broadly categorized into four types[10]:

1. Flooding
2. Probability Based Methods
3. Area Based Methods
4. Neighborhood Knowledge

2.1.1 Flooding

Flooding is perhaps the most straight-forward means of broadcast distribution. To begin, the source node sends out a packet to all nodes within its range. Each node that receives the packet is required to rebroadcast said packet to all of its neighbors exactly once. This requires that each node maintain a record of packets that it has sent. Flooding has no overhead, making it the fastest and least expensive in terms of configuration and associated power usage. It also provides a large amount redundancy, leading to good delivery reliability in very high mobility networks, according to Ho *et al*[4]. Unfortunately, the same redundancy which makes flooding a good choice for high mobility networks leads to significant contention and collisions in low mobility or static WSNs. These problems associated with flooding have been termed the *broadcast storm* problem by Ni *et al*[5].

2.1.2 Probability Based Methods

In [5], Ni *et al* show that an inverse relationship exists between the number of copies of a packet that a node receives and the probability that the node will reach an uncovered area by rebroadcasting. This forms the basis for their Counter Based scheme. In this scheme, upon receipt of a previously unseen packet, a node does not rebroadcast immediately, but instead sets a timer to expire after some amount of time. During the time preceding the timer's expiration, the node counts each copy of the packet that it receives. Once the timer expires, the node compares the packet count to a predetermined threshold value C. Only if the count is less than C, will the node rebroadcast the packet. The probability of a node rebroadcasting a packet is directly related to the density of its one-hop neighborhood. Thus, in a sparse network, many of the nodes will rebroadcast, while in a dense network, relatively few nodes will be allowed to broadcast.

The Probabilistic scheme[5] is based on flooding. Here, upon receiving a packet, a node will rebroadcast it with a probability of P. When this method is employed in a dense network where nodes often have similar coverage areas to those of their neighbors, node and network resources can be saved without sacrificing delivery reliability. In a sparse network, the Probabilistic scheme will result in low delivery reliability due to the lack of shared coverage areas. When P is 1, the Probabilistic scheme is equivalent to flooding.

Pleisch *et al* propose MISTRAL[6], which uses a simple probabilistic forwarding mechanism such as that of Ni *et al* and adds a variation on Forward Error Correction[11] to compensate for packets that are not rebroadcast. Data from packets that are not rebroadcast are XORed together to create the payload for a "compensation packet" which will be sent out periodically. A node receiving a compensation packet can extract a missing data packet provided the node has successfully received all but one data packet as listed in the compensation packet's header. This mechanism is most appropriate for networks in which streams of data that are not time sensitive are being broadcast.

2.1.3 Area Based Methods

Area based methods use distance between nodes to determine whether to rebroadcast. Similar to their Counter Based scheme, Ni *et al*'s Distance Based scheme compares the distance between nodes to a threshold value, D, when determining whether to rebroadcast. Upon receiving a broadcast message, a node determines the distance, d_min, between the sender and itself and starts a timer. If a new copy of the message arrives with a calculated distance d, the smaller of the two values becomes the new d_min. If at any time before the timer expires, it is found that $d_min < D$, the timer is canceled and the node will not rebroadcast. If the timer expires with $d_min \geq D$, the node will rebroadcast. While redundancy is reduced in this scheme and the following Location Based scheme by optimizing coverage area, there is no way to know whether the area covered by a rebroadcasting node will actually reach any new nodes.

In the Location Based Scheme [5], nodes are required to be localized, either through the use of Global Positioning Systems on nodes or a localization protocol [12, 13, 14]. This scheme uses a timer similar to the Distance Based and Counter Based schemes. For a node to determine whether or not it will rebroadcast a message, it keeps track of which of its neighbors have sent copies of the message, and of their locations. With this knowledge, the node will calculate what percentage, p, of its potential transmission area has not been covered by previous senders. If at any time $p < A$, where A is some threshold value ($0 < A < .61$), then the node will not rebroadcast and will ignore all future copies of the message. If, when the timer expires, $p \geq A$, the node will rebroadcast.

2.1.4 Neighborhood Knowledge

Neighborhood knowledge based methods require that each node in a WSN knows its one-hop (and possibly two-hop) neighbors. This aids nodes in determining whether any of its neighbors would benefit from rebroadcasting. In most cases, the one-hop neighborhood is determined using "Hello" packets. Lim and Kim[7] propose two neighborhood knowledge based methods: Flooding with Self-Pruning and Flooding with Dominant Pruning.

The Self Pruning method [7] requires that each node that broadcasts or rebroadcasts a message sends a list of its one-hop neighbors as part of the transmission. When any node receives the broadcast packet, it compares the list of the sender's neighbors to its own one-hop neighborhood. If the node cannot reach any new nodes, it will not rebroadcast. This process is repeated with each copy of the received message, until either the node finds that it should not rebroadcast, or the broadcast timer expires. Since Self Pruning uses only one-hop neighborhood information, it realizes performance gains only toward the perimeter of the network where nodes have fewer one-hop neighbors. The decreasing number of neighbors within transmission range increases the likelihood that a node will not rebroadcast. While any decrease in rebroadcasting nodes is beneficial to network lifetime, the decrease achieved by Self Pruning is far from sufficient for most broadcast applications.

WSNs employing the Dominant Pruning method[7] require that all nodes know the one-hop neighborhood of each of their neighbors. Using this information a node decides which of its neighbors has permission to rebroadcast the packet it is about to send and includes the list as part of the message. Upon receiving a broadcast packet, each node must check to see if it is included in the list. If it is, the node will use a Greedy Set Cover algorithm to determine the next nodes for the rebroadcast list such that all of its two-hop neighbors are covered. The node will refrain from rebroadcasting if it does not find its address in the rebroadcast list. The Scalable Broadcast Algorithm (SBA)[8] also requires two-hop neighborhood knowledge. It makes use of a Random Assessment Delay (RAD)[10] in determining when to send. When a node receives a broadcast packet for the first time, it determines which of its one-hop neighbors have not been covered by the sender's transmission.

If the node's immediate neighborhood has not been fully covered, it initializes a RAD and waits for the RAD to expire before rebroadcasting the packet. If another copy of the packet arrives at any time before the RAD expires, the node will reevaluate its potential coverage area by removing the new sender's covered nodes. This is repeated until such time as the RAD expires and the packet is rebroadcast, or the nodes potential coverage area becomes empty and the RAD is canceled. Peng *et al* suggest that the RAD time should be dynamically determined by scaling the base RAD time according to the ratio, $\frac{d_N max}{d_m e}$, where $d_N max$ is the maximum neighbor degree of the node's 1-hop neighbors and $d_m e$ is the degree of the current node. Using this scaling factor, nodes with more one-hop neighbors will be able to rebroadcast sooner than those nodes with fewer one-hop neighbors.

LENWB, the Lightweight and Efficient Network-Wide Broadcast protocol[15], utilizes two-hop neighborhood information in combination with knowledge of a broadcaster's coverage area to determine whether it should rebroadcast. Upon receipt of a packet, the node determines which of the nodes in the sender's coverage area have a higher priority to rebroadcast, that is, have a higher degree than it does. If the neighbors with higher degrees will cover all of the lower priority neighbors, the node will not broadcast. If they will not cover all of the lower priority neighbors, the node rebroadcasts.

Similar to Dominant Pruning, in Multipoint Relaying[9] a node selects which of its neighbors will be allowed to rebroadcast a packet. In this method, a broadcasting node must first determine which of its two hop neighbors can only be reached by one 1-hop neighbor; these 1-hop neighbors are added to the list of MultiPoint Relays (MPRs). The node then determines which nodes in its two-hop neighborhood will not be reached by the current list of MPRs. From this list, the node selects the 1-hop neighbor which covers the largest number of uncovered nodes and adds it to the MPR list. This process of calculating coverage and adding to the MPR list continues until all two-hop neighbors are covered. The MPR list is added to periodic "hello" messages. When a node receives a broadcast packet, it looks up the sender in its MPR table to determine whether it is an MPR for that node and is supposed to rebroadcast.

CHAPTER THREE

THE LIGHTWEIGHT KEY DISTRIBUTION TREE

The topic of the physical distribution of keys in a WSN is approached with three goals in mind: to develop a mechanism that is energy efficient over the long-term, to provide reliable service to all nodes, and to ensure that the mechanism is lightweight, requiring limited communication and processing power thereby allowing for rapid set-up of the broadcast structure.

In order to provide the most energy efficient key distribution mechanism possible, the network must consume as little power as possible to reach all the nodes. In an ideal network, key distribution would happen such that the key is forwarded by the absolute minimum number of nodes required, while at the same time each node receives the key exactly once. This would result in the absolute minimum amount of energy consumption. In a wired network, one might choose to implement a maximal leaf tree (MLT) type topology. We encounter two primary problems with applying a MLT-based approach to WSN broadcast structures. First, constructing a MLT has been shown to be NP-hard[16]. Additionally, the broadcast nature of WSNs makes achieving this goal impossible; even when using a minimal number of nodes to forward packets out from the source, many nodes in the network will receive duplicate copies of the packet as can be seen in Figure 3.1. If node A broadcasts a packet to its one-hop neighbors and node B has been selected to forward the broadcast, all nodes that are one-hop neighbors of both A and B (those within the shaded area of Figure 3.1) will receive a second copy of the packet. Multiple nodes broadcasting in the same region will cause contention, leading toward the broadcast storm problem discussed in Section 2.1.1.

The LKDT mechanism constructs a limited interior-node tree, which significantly reduces the amount of nodes both forwarding broadcasts, and suffering from redundant coverage.

In describing the operation of LKDT, two important terms are used, defined here. A *forwarder* is a node that is tasked with rebroadcasting any encryption key packet that it receives. A node is *covered*, if it is in the one-hop neighborhood of a forwarder.

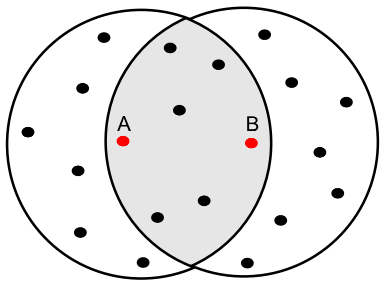


Figure 3.1: Example of overlapping coverage.

3.1 Design Requirements and Assumptions

LKDT is designed for use in a static network. We make no assumptions about the communication range of nodes; uniform transmission ranges are not required. Additionally, LKDT needs to be independent of any specific encryption method or security protocol, so it can be used for distribution of any key or any time delivery of a packet must be guaranteed for all nodes. Measurable power savings over simple broadcast is required, as is the reduction of multiple coverage of nodes for the sake of both power and congestion issues. Finally, high reliability in key delivery is a top priority. All nodes within the WSN which are capable of receiving keys must receive each key in order to maintain network security and key freshness.

3.2 LKDT

There are four stages in the construction and use of an LKDT:

1. Neighborhood Discovery
2. Tree Building
3. Multi-Coverage Reduction
4. Key Distribution and the Key Retry Mechanism

Each of these phases is described below.

3.2.1 Neighborhood Discovery

The neighborhood discovery phase of LKDT is typical of many hello protocols; each node in the network broadcasts a *hello* packet to its one-hop neighbors. In this way, each node learns its full one-hop neighborhood so that building of the broadcast tree may commence. LKDT differs, however, in that a node does not broadcast its *hello* packet until it has received one from a neighbor. The base station sends the first *hello* packet, initiating a wave of *hello*s that ripples out from its location toward the edges of the network. As a result, the base station will be able to continue with the rest of the tree building process while the outer nodes are still learning about their neighbors. Additionally, the wave action reduces the number of broadcasts sent at any one time, thereby limiting the effect of the broadcast storm problem, since only a portion of the nodes are attempting to introduce themselves at any particular moment.

3.2.2 Tree Building

As with neighborhood discovery, the tree building phase occurs as a wave. The base station begins the process by broadcasting a *forwarder decision packet* (FDP) to all nodes within its one-hop neighborhood. The FDP indicates to any receiving node that the sender has elected to be a forwarder and also contains a list of all nodes that the sender has added to the 'covered' portion of the network. (For the base station, this list would contain its entire one-hop neighborhood.) At this point, the base station's work is completed, as it forms the root of the broadcast tree.

We will use the following definitions to describe the building of the rest of the broadcast tree:

- S_i is set of nodes in the one-hop neighborhood of node n_i
- T_i is the set of non-covered nodes in S_i
- C_i is the set of covered nodes in S_i

For all other nodes n_i, tree building (also described in Algorithm 1) begins with the receipt of a FDP from a node $n_j \in C_i$ which has decided to cover it. At this point, the node is part of the covered portion of the network, meaning it must now begin the process of deciding whether or not to become a forwarder. Once n_i has received what it assumes to be its last FDP (nodes may receive more than one FDP), it uses the list of covered nodes from each FDP to determine what nodes in S_i make up its list of non-covered nodes, T_i. Node n_i then broadcasts this list as part of a *neighborhood listing packet* (NLP).

Algorithm 1 Packet Processing for Node n_i

```
  $isCovered$ ← **false**
  **loop**
    **switch** (event):
      **case** *receive*:
        pkt ← packet received
        **switch** (pkt.message):
          **case** *forwarder decision*:
            **if false==** $isCovered$ **then** {$n_i$ is not covered}
              $isCovered$ ← **true**
              Send coverage announcement packet
            **end if**
          **case** *coverage announcement*:
            addToCoveredPacketList(pkt.src, pkt.coveredBy)
          **case** *neighborhood listing*:
            addToTwoHopNeighborhood(pkt.src,pkt.neighbors)
        **endswitch**
      **case** *timer expire*:
        **switch** (timer.event):
          **case** *lastNeighborListRcvd*:
            forwarderStatus = determineForwarderStatus {See Alg. 2}
            **if true==** forwarderStatus
              Send forwarder decision packet
            **end if**
          **case** *lastCoverageAnnouncementRcvd*:
            forwarderStatus = reduceTreeSize() {See Alg. 3}
          **case** *lastForwarderDecisionRcvd*:
            Send neigborhood list packet
        **end switch**
    **end switch**
  **end loop**
```

As nodes receive NLPs, they collect and save packet source and neighborhood lists. Once a node has not received a neighborhood list for some period of time, it uses the source/list pairs to determine whether or not to elect itself as a forwarder. Prior to this point, each node is, by default, a forwarder. The forwarding decision process is described in Algorithm 2. The decision making process is, in part, a greedy one. If there is any node in T_i which cannot be covered by any other node (we call this a 'unique' node), n_i will be a forwarder. If, however, there are no unique nodes in T_i, n_i uses a combination of comparing potential coverage sets and potential coverage counts (how many nodes can be covered if n_i chooses to forward versus $n_j \in C_i$) to determine whether or not it should 'favor' forwarding.

If n_i contains any 'unique' nodes or 'favors' forwarding, it must now send out its own FDP. Unlike the base station which sent out a list of all nodes in its one-hop neighborhood, n_i will only include in its list those nodes which were previously not covered, T_i.

This process continues until all nodes in the network have determined their forwarder status.

The primary idea behind this process is that we wish only those nodes which will add the largest amount of nodes to the tree or will reach nodes that no other node can, to select themselves as forwarders. In this way, our heuristic is decidedly greedy.

3.2.3 Multi-Coverage Reduction

Although the tree building mechanism described in Section 3.2.2 significantly reduces the number of forwarding nodes in the network, it can still leave many nodes receiving duplicate coverage. To reduce the number of forwarding nodes even further, Multi-Coverage Reduction (MCR) is employed following the construction of the tree.

MCR requires that each node, upon receiving its first FDP and determining that it is now covered, send out a *coverage announcement packet* (CAP) which contains the ID of the node providing the coverage, its *parent* node. The CAP informs receiving nodes that the sender has been covered, and by whom. By using this information, once a node is no longer receiving CAPs, it can re-evaluate its forwarder status using Algorithm 3. If the node determines that it no longer

Algorithm 2 Forwarder Status Determination Function for Node n_i

```
Require: $S_i$ is the set of nodes in the 1-hop neighborhood of node $n_i$
Require: $C_i$ is the set of covered nodes in $S_i$
Require: $T_i$ is the set of non-covered nodes in $S_i$
Require: $S_i = C_i \cup T_i$
  $favor \leftarrow 0$
  Construct set $Y_i$ s.t. $\forall$ nodes $n_j \in C_i$, $Y_i = (Y_i \cup T_j) \cap T_i$
  if $Y_i \subset T_i$ then
    {$\exists$ nodes that only $n_i$ can cover}
    return true
  else
    {Determine if $n_i$ covers a sufficient number of nodes}
    for all $n_j \in T$ do
      if $C_j \subseteq C_i$ then
        if $|C_i| = |C_j|$ then
          if node id of $n_i >$ node id of $n_j$ then
            $favor \leftarrow favor + 1$
          else
            $favor \leftarrow favor - 1$
          end if
        else
          $favor \leftarrow favor + 1$
        end if
      else if $|C_i| \geq |C_j|$ then
        $favor \leftarrow favor + 1$
      else
        $favor \leftarrow favor - 1$
      end if
    end for
    if $favor > 0$ and $|C_i| > \frac{1}{|T_i|} \sum_{n_j \in T_i} |C_j|$ then
      return true
    else
      return false
    end if
  end if
```

needs to forward, it can change its status without detriment to the integrity of the broadcast tree.

Algorithm 3 Tree Reduction Function for Node n_i

```
$M \leftarrow$ 1-hop neighbors of $n_i$ that have reported being ‘covered’
for all $m_i \in M$ do
    if $m_i$ reports coverage received from $n_i$ then
        Node $n_i$ is providing coverage and so must remain a forwarder
        return true
    end if
end for
Node $n_i$ is not providing coverage to any nodes
return false
```

As one of the primary goals of this work is to ensure LKDT is lightweight, MCR requires each node to send out only a single packet. We refrain from performing large optimization sweeps that, while providing potential energy savings benefits in the long run, require significant time and extra communication to perform, thereby depleting crucial energy for questionable gains.

3.2.4 Key Distribution and the Key Retry Mechanism

Following the completion of the tree, key distribution occurs as forwarding nodes receive and then rebroadcast encryption key packets. In the event that a forwarder receives the same packet more than one time, only one copy of the packet will be forwarded. Each node that has been ‘covered’ by a forwarder should receive at least one copy of the key packet.

When distributing encryption keys, it is vital that all nodes receive each key, that is, the WSN must have 100% delivery reliability. Packet loss is inevitable in any WSN; collisions, contension, and outside intereference can all cause packet loss. Even the best MAC layer cannot prevent all faults. As such, it becomes necessary to use a Key Retry Mechanism (KRM) to compensate for this loss. Our KRM is a negative acknowledment, or NAK, based mechanism and is detailed in Algorithm 4. Each node keeps track of when the next key broadcast event is expected. Once the time for the expected broadcast arrives, the node enters a *key delivery period.* If a key is successfully received at any point, the node begins waiting for the next key update interval to arrive. If the node does not receive a key before the key delivery period ends, the node sends a Key Retry Request (KRR) to its parent node and a new key delivery period begins.

Algorithm 4 Key Retry Mechanism for Node n_i

```
switch (event):
  case receive:
    pkt ← packet received
    if pkt.type == keyUpdate then
      updateKey
      timer.reset(keyUpdateInterval)
    end if
  case timer expire:
    switch (timer.event):
      case keyUpdateInterval:
        timer.reset(keyDeliveryPeriod)
      case keyDeliveryPeriod:
        Send key retry request packet
        timer.reset(keyDeliveryPeriod)
    end switch
end switch
```

If a forwarder receives a KRR packet from one of its covered nodes, it immediately resends the missing key. All of the parent node's one-hop neighbors will receive the rebroadcast key. This extra transmission, while it sacrifices valuable power in the parent node's neighbors, is necessary to ensure delivery reliability and so is deemed a justifiable expense.

3.3 Summary

LKDT provides a limited interior node broadcast tree for key distribution. The broadcast tree is not optimal due to limited knowledge on the part of the network nodes. This lack of optimization was deemed acceptable and necessary to maintain our goals for a lightweight and low power mechanism. Despite the lack of optimization, LKDT still provides a significantly reduced collection of forwarders while ensuring high levels of coverage and reliable key distribution.

CHAPTER FOUR

SIMULATION AND EXPERIMENTAL RESULTS

In order to evaluate the performance of LKDT, we implemented LKDT, both with and without MCR in the *ns*-2 simulator[17]. We chose to use SBA and simple flooding for comparisons. SBA was selected as it is, like LKDT, a Neighborhood Knowledge based mechanism (see Chapter 2.1.4) and performed well under evaluations by Camp and Williams [10]. The *ns*-2 implementation of SBA used was a port of Camp and Williams' implementation [10, 18]. Parameters common to all simulations can be seen in Table 4.1. In order to evaluate the performance of LKDT under varying network densities, random topologies of 50 to 250 nodes spread over a 1000 x 1000 meter area were used. This yields networks in which nodes have an average one-hop neighborhood size of between approximately 10 to 50 nodes each, given a maximum transmission radius of 250 meters. Details of density and neighborhood size can be seen in Table 4.2.

Our goals are to evaluate LKDT in terms of configuration cost, power usage and network lifetime, network coverage, and delivery reliability. To determine configuration cost, we examine the overhead and power consumption associated with initial setup of the broadcast tree. power usage and network lifetime are examined through analysis of duplicate coverage and average node power level. Network coverage refers the ability of the mechanism to include all nodes in the network as part of the broadcast tree, either as a forwarder or as a leaf. Delivery reliability pertains to the ability of the network to distribute keys to all nodes within communication range.

With the severe energy constraints inherent in WSNs, limiting energy consumption is critical. In our simulations, the base station generates and broadcasts a new key packet every 10 seconds over a total simulation time of 200 seconds. In an actual sensor network, keys would not be updated this frequently, with keys being sent out on the order of minutes, rather than seconds. However, to evaluate LKDT in a reasonable amount of time, we chose to increase the frequency to a level such that broadcasts of different keys would not overlap in time, but a reasonable number of keys would

Table 4.1: Simulation Settings

Simulation Parameter	Value
Simulator	ns-2 v2.29
MAC	802.11
Network Area	1000 x 1000 m
Node Tx Distance	250 m
Node Tx Power	0.6 W
Node Rx Power	0.2 W
Node Battery Power	10.0 J
Node Max. IFQ Length	50
# of Trials	10
Key Update Interval	10 sec

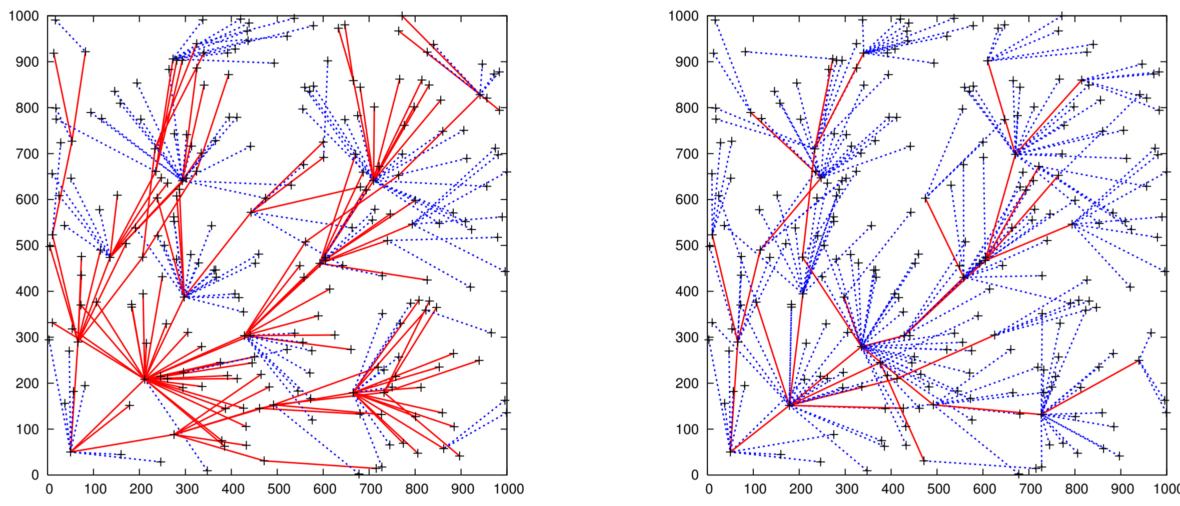
Table 4.2: Network Density

# Nodes	ρ_{avg}	Avg. # Neighbors
50	0.00005	9.817
100	0.00010	19.635
150	0.00015	29.452
200	0.00020	39.270
250	0.00025	49.087

be distributed during the course of the simulation.

4.1 Configuration Cost

In discussing configuration cost for LKDT, we must first examine MCR. Initially, MCR was to be an optional component when deploying networks using LKDT. It was determined however, that due to the significant reduction in forwarders when MCR is employed, it is a critical, not optional component of LKDT. The optimization provided by MCR can be visually observed in Figure 4.1. Solid red lines represent branches of the broadcast tree, connecting each of the forwarding nodes. Connections to leaf nodes are represented by dashed blue lines. In Figure 4.1, the use of MCR reduces the number of forwarders by nearly 75%. For this reason, all analysis of LKDT will use LKDT with MCR. Unless explicityly stated otherwise, all referneces to LKDT will refer to LKDT with MCR.



(a) Network without MCR.

(b) Network with MCR.

Figure 4.1: A 250 node network with base station at (50,50). Solid lines represent branches of the forwarding tree.

Figure 4.2 shows the average remaining battery power for a node in a 250 node WSN during initial configuration. The three versions of LKDT (without MCR and KRM, and with MCR both

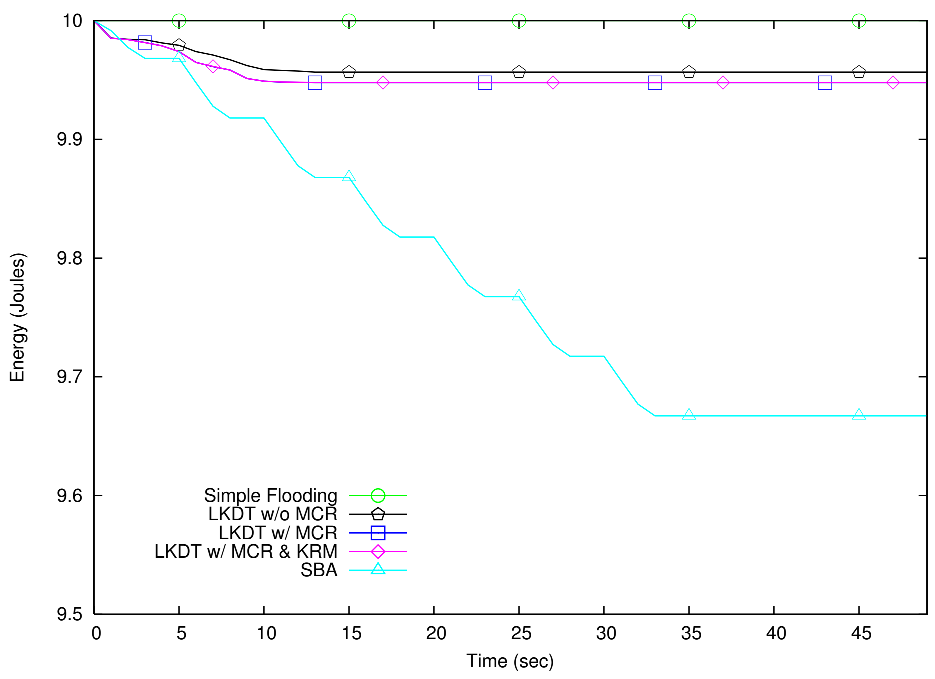


Figure 4.2: Power Usage for Configuration

with and without KRM) have a short setup period, between $t = 0$ and $t = 13$ seconds, during which time the four stages of configuration take place. The added cost of MCR is visible as the difference between energy levels for the LKDT versions. This slight increase in energy use is acceptable given the reduction in duplicate broadcasts that MCR provides.

SBA does not use the same type of configuration mechanism as LKDT, instead using periodic hello messages to keep neighborhood information up to date. Neighborhood data also has an expiration time recorded along with it, to ensure freshness of data. This polling process is highly appropriate for mobile networks, but in static networks it creates excessive overhead. To test SBA in static networks, data expiration was disabled and the protocol was limited to 30 seconds of “hello” time to ensure that nodes had full neighborhood data without overly taxing the nodes through excessive “hellos”. Later simulations reset the battery power on each node at $t = 50$ seconds so that the operation of SBA and LKDT can be compared solely based on key distribution ability from a common starting point.

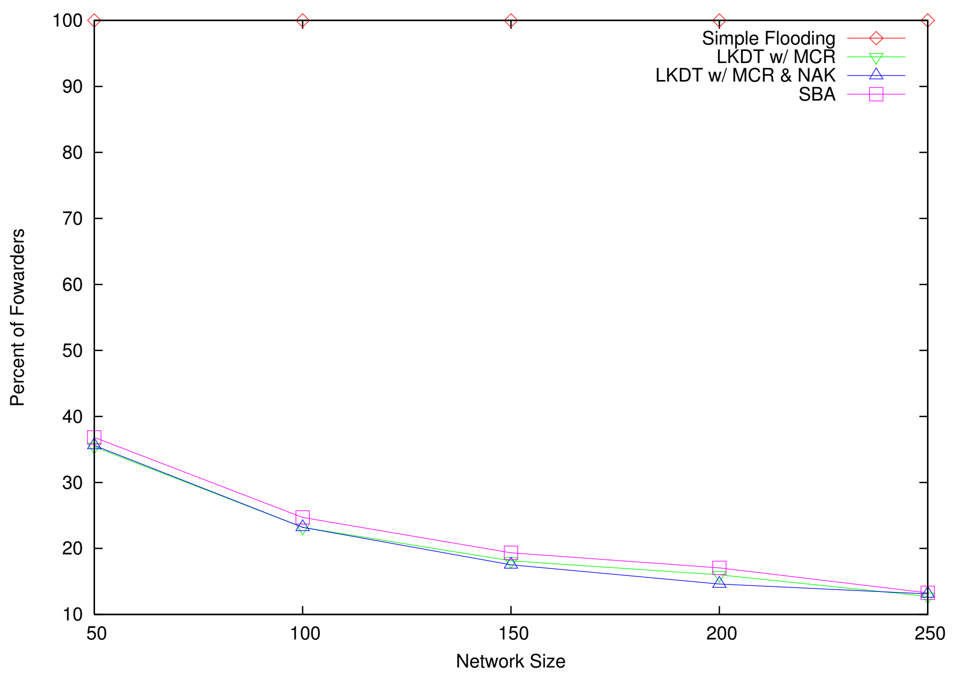


Figure 4.3: Ratio of Forwarders vs. Network Size

4.2 Power, Network Lifetime, and Efficiency

One of the key goals behind LKDT was to decrease the number of forwarding nodes needed to reach the entire WSN. Figure 4.3 shows the average number of forwarders for a given size network. As expected, simple flooding has 100% of the nodes forwarding. The use of LKDT results in fewer forwarders than are used by SBA. This is most likely due to the difference in decision making processes between SBA and LKDT. LKDT factors in relative sizes of neighborhoods when deciding on forwarder status; it is our belief that this extra information is what allows LKDT to select a more optimal set of forwarders.

Figure 4.4 shows the average number of copies of a single key that are received by a node. The ideal number of copies per node is one but, as was discussed in Chapter 3, this is not possible. In looking at the graph, we can see that the use of simple flooding results in substantially more duplicate keys than either version of LKDT or SBA. This is to be expected as all nodes are required to forward each key received. Surprisingly, though SBA typically has more forwarders than LKDT,

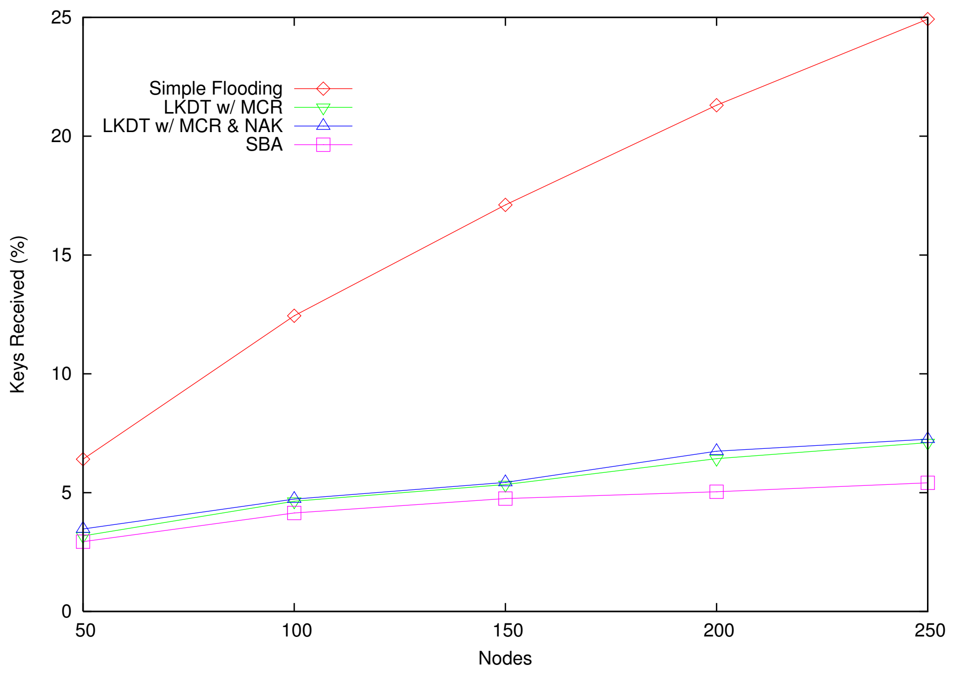


Figure 4.4: Average number of transmissions received by a node per key broadcast vs. network size

its use results in slightly fewer duplicate broadcasts. LKDT still results in significantly fewer duplicates than simple flooding. Comparing the performance of LKDT both with and without KRM, we can see that in the case of the 50 and 200 node networks, KRM was employed, resulting in a slight increase in duplicates. SBA's reduced number of duplicates with respect to LKDT is most likely due to SBA's dynamic decision making process for forwarders; nodes listen to broadcasts occuring around them, then use their neighborhood knowledge to determine if any nodes have been "left out". This would reduce the number of duplicate keys received compared to LKDT but as there is no feedback mechanism, the reliability of key delivery is still questionable.

From Figure 4.5, we can see that both versions of LKDT use a significantly reduced amount of power over the course of the simulation when compared to the simple flooding method. SBA also outperforms simple flooding, but not to the same extent as LKDT. By comparing Figure 4.5(a) and Figure 4.5(b), we can see that as the density of the network increases, so too does the power savings provided by LKDT and SBA. LKDT's power savings over SBA is directly related to the reduced number of forwarders used by LKDT.

4.3 Coverage

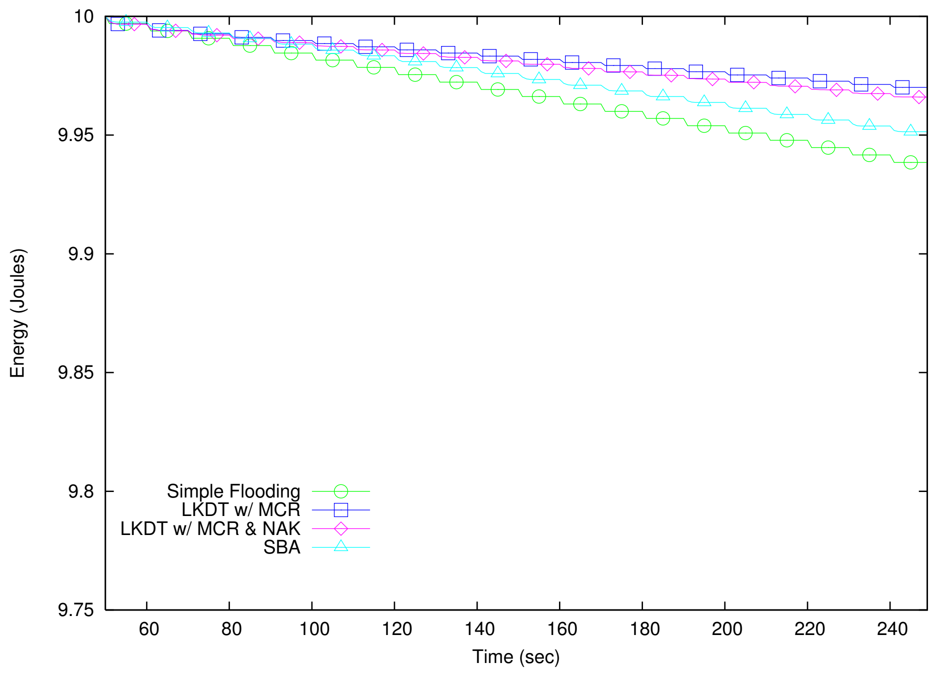
In order to ensure that all nodes within the network are capable of receiving every new network key, we must have full network coverage, meaning that every node is included in the broadcast tree, either as a forwarder or as a leaf node. If a node is excluded from the tree, it cannot receive encryption key broadcasts, and so will be unable to fully participate in the network and will be vulnerable to attacks.

For LKDT to meet our requirements for a good key distribution tree, it needed to successfully cover all reachable nodes in the WSN. We compared LKDT with SBA and simple flooding to determine how our mechanism would perform. Looking at Figure 4.7, we can see that all methods tested perform equally well, providing 100% coverage for networks of 100 or more nodes. Coverage of the 50 node network was only 94% for all methods. After examining the network topologies generated for this test, we found that in two of the ten topologies used there were one or more nodes which were beyond communication range and therefor unreachable and not covered. An example of a network with unreachable nodes can be seen in Figure 4.6. In this figure, neither simple flooding nor LKDT can reach the two excluded nodes. Their location is such that they are outside of communication range for the rest of the network. With this in mind, we can safely say that LKDT provides a high level of coverage when deployed in a network.

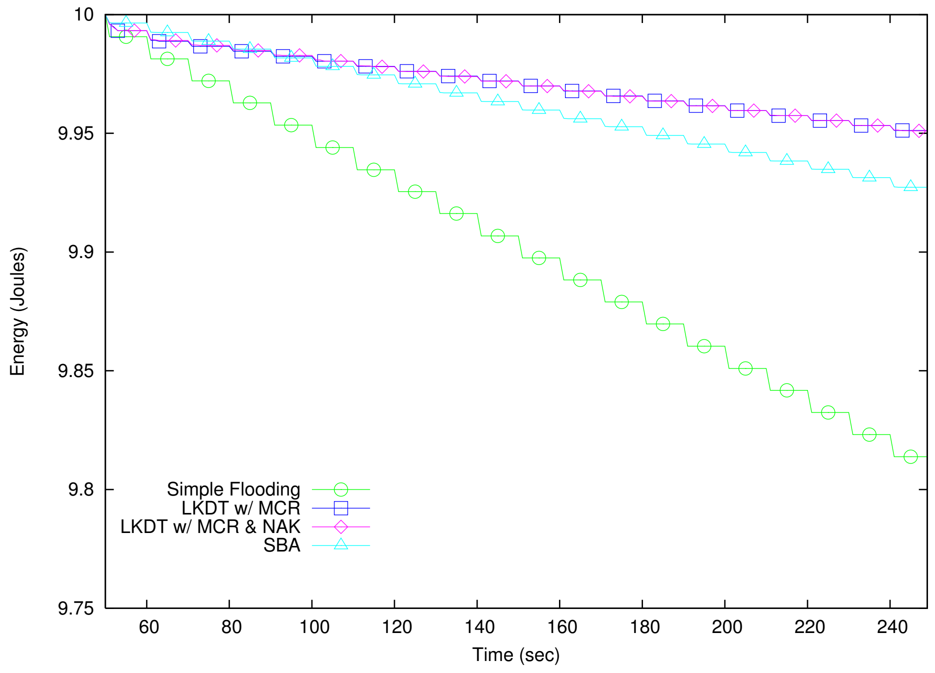
4.4 Reliability

Reliability is possibly the most crucial metric for a key distribution system. Without reliable delivery of keys, network security is threatened and formerly secure nodes can be excluded due to a key that was missed. Figure 4.8 shows the percent of keys received for an average node relative to network size. We can see that LKDT without KRM has very poor reliability. The sparseness of the network's forwarders is to the detriment of delivery reliability. It is important to note, however, that KRM makes a very significant difference in LKDTs reliability, outperforming simple flooding and performing comparably well to SBA for networks of all sizes.

Figure 4.8 shows reliability when keys are the only traffic on the network. When other network traffic (sensor data) is added, it is our belief that the reliability of SBA will decline significantly while LKDT with KRM will continue to perform well, since nodes will have a means to indicate when keys are missing.

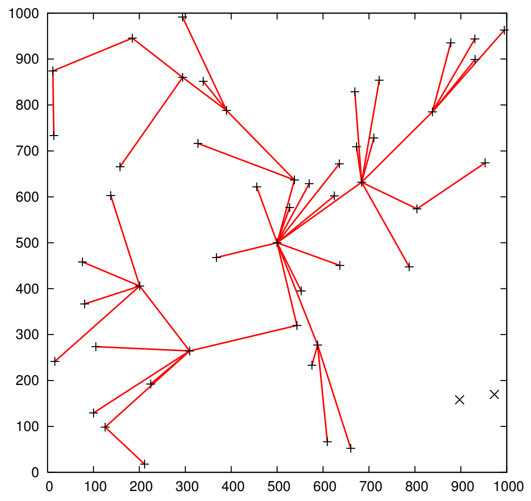


(a) 100 node network.



(b) 250 node network.

Figure 4.5: Average node power level over time for two sizes of WSN.


(a) Simple flooding.

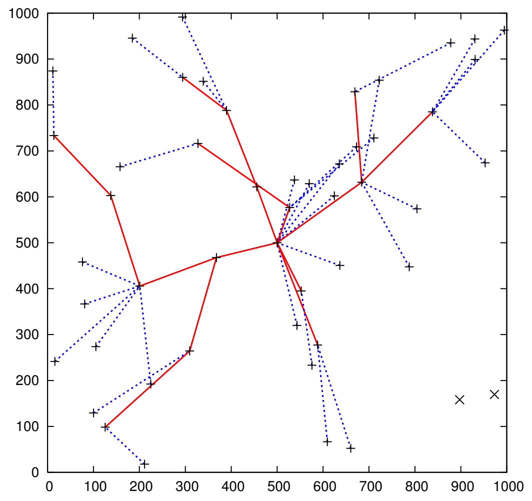

(b) LKDT with MCR.

Figure 4.6: 50 node network with two unreachable nodes.

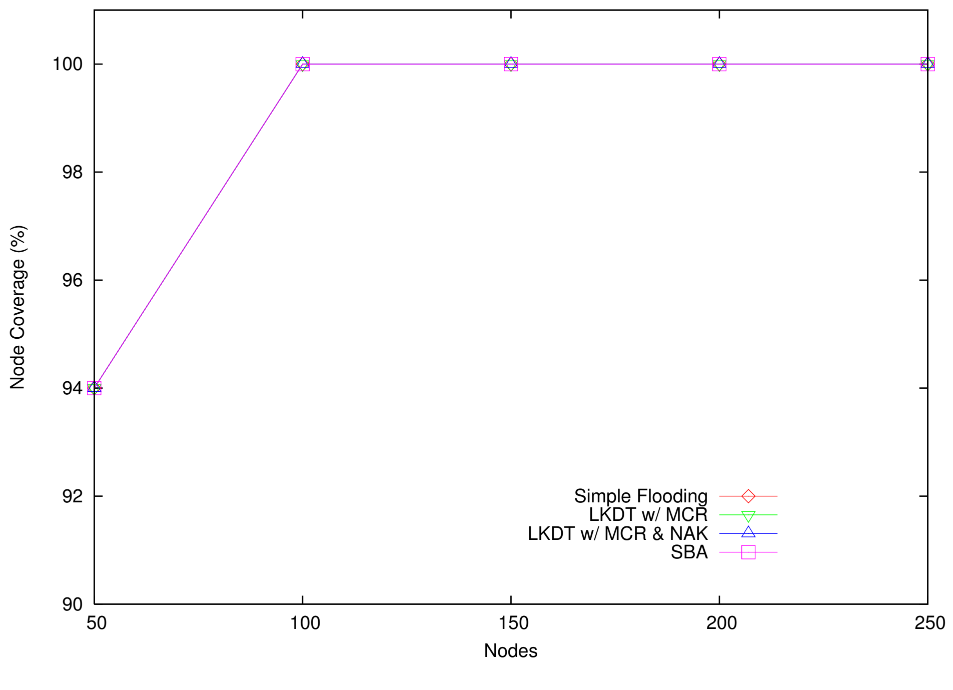


Figure 4.7: Network coverage

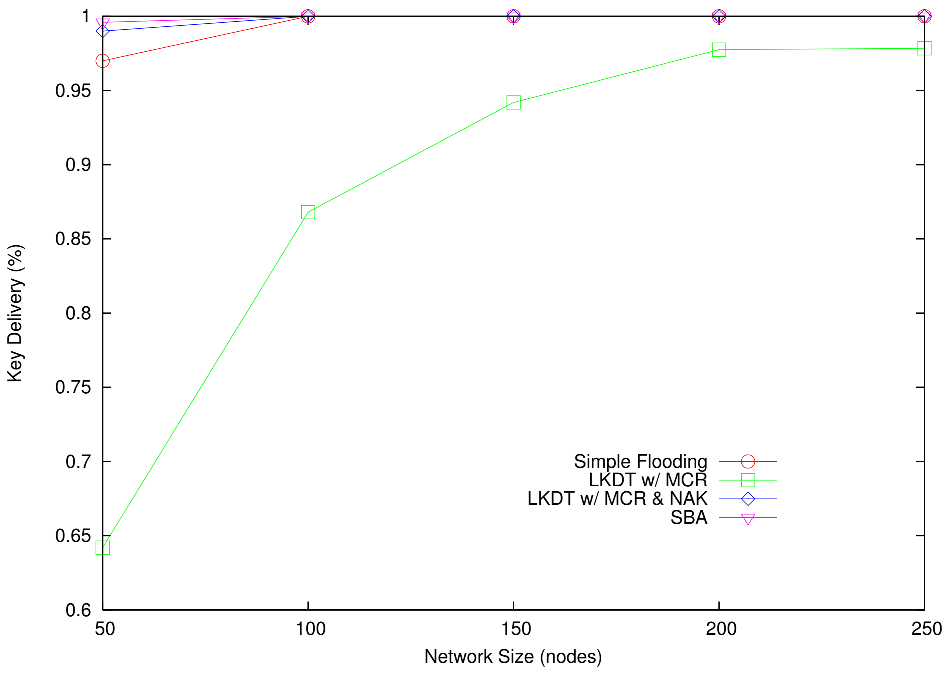


Figure 4.8: Delivery Reliability vs Network Size

CHAPTER FIVE

CONCLUSION

Wireless sensor networks, while versatile in their potential applications, are very constrained by limited battery lifetime. Due to this concern for network longevity, research into sensor networks must emphasize energy savings. Security is no exception to this need for low power usage. The use of encryption to provide packet level security has become the accepted standard in WSN research, but means for efficient physical distribution of these keys is needed.

We developed LKDT, a distributed algorithm to solve the problem of distributing encryption keys used for broadcast (i.e. network-wide) communication in a static wireless sensor network. LKDT constructs a backbone, a key distribution tree, within the WSN, along which encryption keys are forwarded for distribution. Construction of the tree involves each node determining whether to join the distribution tree based upon knowledge of its two-hop neighborhood. We implemented LKDT in the *ns*-2 simulator, and conducted repeatable simulations to test and verify its performance. Our simulations show that use of the backbone results in significant energy savings over standard broadcast flooding.

To help compensate for MAC layer based losses of encryption key packets, we chose to implement a NAK type request for use in the event that a node misses an encryption key. This request asks the forwarder of that node to try resending the key, so as to ensure that all nodes in the network remain up to date. Utilization of the Key Retry Mechanism results in a greater than 99.5% success rate for key delivery to all nodes.

There are several areas which we would like to explore, in hopes of further improving the performance and versatility of LKDT. Future work includes extending LKDT to work in conjunction with MAC protocols that implement sleep/wake periods for nodes in an effort to conserve energy. Examples of these protocols include ECR-MAC[19], S-MAC[20] and LoC-MAC[21]. Other work includes development of a version of LKDT appropriate for mobile WSNs and a tree repair mechanism to be utilized when a forwarding node reaches a critical power level and can no longer perform its duty as a forwarder.

BIBLIOGRAPHY

[1] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, “The tesla broadcast authentication protocol,” *RSA CryptoBytes*, vol. 5, p. 2002, 2002.

[2] S. Zhu, S. Setia, and S. Jajodia, “Leap: efficient security mechanisms for large-scale distributed sensor networks,” in *CCS ’03: Proceedings of the 10th ACM conference on Computer and communications security*. New York, NY, USA: ACM Press, 2003, pp. 62–72.

[3] ——, “Leap+: Efficient security mechanisms for large-scale distributed sensor networks,” *ACM Trans. Sen. Netw.*, vol. 2, no. 4, pp. 500–528, 2006.

[4] C. Ho, K. Obraczka, G. Tsudik, and K. Viswanath, “Flooding for reliable multicast in multi-hop ad hoc networks,” in *DIALM ’99: Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications*. New York, NY, USA: ACM, 1999, pp. 64–71.

[5] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, “The broadcast storm problem in a mobile ad hoc network,” in *MobiCom ’99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*. New York, NY, USA: ACM, 1999, pp. 151–162.

[6] S. Pleisch, M. Balakrishnan, K. Birman, and R. van Renesse, “Mistral: efficient flooding in mobile ad-hoc networks,” in *MobiHoc ’06: Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM, 2006, pp. 1–12.

[7] H. Lim and C. Kim, “Multicast tree construction and flooding in wireless ad hoc networks,” in *MSWIM ’00: Proceedings of the 3rd ACM international workshop on Modeling, analysis*

and simulation of wireless and mobile systems. New York, NY, USA: ACM, 2000, pp. 61–68.

[8] W. Peng and X.-C. Lu, “On the reduction of broadcast redundancy in mobile ad hoc networks,” in *MobiHoc ’00: Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*. Piscataway, NJ, USA: IEEE Press, 2000, pp. 129–130.

[9] A. Qayyum, , L. Viennot, and A. Laouiti, “Distributed policy management and comprehension with classified advertisements,” INRIA: Institut National de Recherche en Informatique et en Automatique, Tech. Rep. 3898, March 2000.

[10] B. Williams and T. Camp, “Comparison of broadcasting techniques for mobile ad hoc networks,” in *MobiHoc ’02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*. New York, NY, USA: ACM, 2002, pp. 194–205.

[11] C. Huitema, “The case for packet level fec,” in *PfHSN ’96: Proceedings of the TC6 WG6.1/6.4 Fifth International Workshop on Protocols for High-Speed Networks V*. London, UK, UK: Chapman & Hall, Ltd., 1997, pp. 109–120.

[12] C. Mallery, S. Medidi, and M. Medidi, “Relative localization with 2-hop neighborhood,” *World of Wireless, Mobile and Multimedia Networks, 2008. WoWMoM 2008. 2008 International Symposium on a*, pp. 1–4, June 2008.

[13] S. Capkun, M. Hamdi, and J.-P. Hubaux, “Gps-free positioning in mobile ad-hoc networks,” *System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on*, pp. 10 pp.–, Jan. 2001.

[14] D. Niculescu and B. Nath, “Ad hoc positioning system (aps) using aoa,” *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, vol. 3, pp. 1734–1743 vol.3, March-3 April 2003.

[15] J. Sucec and I. Marsic, “An efficient distributed network-wide broadcast algorithm for mobile ad hoc networks,” Rutgers University, Tech. Rep. 248, September 2000.

[16] T. Fujie, “An exact algorithm for the maximum leaf spanning tree problem,” *Comput. Oper. Res.*, vol. 30, no. 13, pp. 1931–1944, 2003.

[17] S. McCanne and S. Floyd, “ns Network Simulator,” http://www.isi.edu/nsnam/ns/.

[18] B. Barritt, B. Malakooti, and Z. Guo, “Intelligent multiple-criteria broadcasting in mobile ad-hoc networks,” *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pp. 761–768, Nov. 2006.

[19] Y. Zhou and M. Medidi, “Energy-efficient contention-resilient medium access for wireless sensor networks,” *Communications, 2007. ICC ’07. IEEE International Conference on*, pp. 3178–3183, June 2007.

[20] W. Ye, J. Heidemann, and D. Estrin, “An energy-efficient mac protocol for wireless sensor networks,” *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, pp. 1567–1576 vol.3, 2002.

[21] M. Kohagura, “Local Coordination Medium Access Control for Wireless Sensor Networks,” Master’s thesis, Washington State University, Pullman, WA, August 2008.