Fall 2008

Energy efficient clustering and secure data aggregation in wireless sensor networks

Julia Albath

Department: Computer Science

ENERGY EFFICIENT CLUSTERING AND SECURE DATA AGGREGATION IN WIRELESS SENSOR NETWORKS

by

JULIA GERDA MARIA ALBATH

A DISSERTATION

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

2008

Approved by:

Dr. Sanjay Madria, Advisor
Dr. Jennifer Leopold
Dr. Fikret Ercal
Dr. Sriram Chellappan
Dr. Jaganathan Sarangapani

PUBLICATION THESIS OPTION

This thesis consists of the following four articles that have been published or submitted for publication as follows:

Pages 13-36 have been submitted to the 7^{th} Annual IEEE International conference on Pervasive Computing and Communications.

Pages 37-65 are intended for submission to IEEE Transactions on Mobile Computing.

Pages 66-87 have been published in the 6^{th} International Workshop on Data Engineering for Wireless and Mobile Access.

Pages 88-103 have been submitted for publication to the IEEE Wireless Communications & Networking Conference 2009.

ABSTRACT

Communication consumes the majority of a wireless sensor network's limited energy. There are several ways to reduce the communication cost. Two approaches used in this work are clustering and in-network aggregation. The choice of a cluster head within each cluster is important because cluster heads use additional energy for their responsibilities and that burden needs to be carefully distributed. We introduce the energy constrained minimum dominating set (ECDS) to model the problem of optimally choosing cluster heads in the presence of energy constraints. We show its applicability to sensor networks and give an approximation algorithm of $O(\log n)$ for solving the ECDS problem. We propose a distributed algorithm for the constrained dominating set which runs in $O(\log n \log \Delta)$ rounds with high probability. We show experimentally that the distributed algorithm performs well in terms of energy usage, node lifetime, and clustering time and thus is very suitable for wireless sensor networks. Using aggregation in wireless sensor networks is another way to reduce the overall communication cost. However, changes in security are necessary when in-network aggregation is applied. Traditional end-to-end security is not suitable for use with in-network aggregation. A corrupted sensor has access to the intermediate data and can falsify results. Additively homomorphic encryption allows for aggregation of encrypted values, with the result being the same as the result as if unencrypted data were aggregated. Using public key cryptography, digital signatures can be used to achieve integrity. We propose a new algorithm using homomorphic encryption and additive digital signatures to achieve confidentiality, integrity and availability for in-network aggregation in wireless sensor networks. We prove that our digital signature algorithm which is based on Elliptic Curve Digital Signature Algorithm (ECDSA) is at least as secure as ECDSA. Even without in-network aggregation, security is a challenge in wireless sensor networks. In wireless sensor networks, not all messages need to be secured with the same level of encryption. We propose a new algorithm which provides adequate levels of security while providing much higher availablility than other security protocols. Our approach uses similar amounts of energy as a network without security.

ACKNOWLEDGMENT

I am grateful and thankful to my Ph.D. advisor Dr. Sanjay Madria who has given me the necessary guidance to finish this dissertation. His understanding, support and advice have been crucial factors in my successful completion of this Ph.D. I enjoyed the room to think and grow and felt inspired to be my best. It has been a great experience to work under Dr. Madria and I greatly expanded my knowledge.

I want to thank Dr. Leopold who has mentored me from the beginning. Her guidance has helped me chart a course during my time in Rolla and for my future. I thank Dr. Sarangapani for his assistance, especially during the early years. I thank Dr. Ercal for his invaluable feedback and support as part of my committee. I thank Dr. Chellappan for his feedback and assistance which helped improve my research.

I express my deep appreciation and gratitude to Ryan Underwood. I am grateful for all my friends who made my time in Rolla enjoyable.

TABLE OF CONTENTS

Page

LIST OF ILLUSTRATIONS

1. INTRODUCTION

Wireless sensor motes are small, battery powered computing devices. For example, the Crossbow MICAz motes consist of a CPU, an on-board radio and a sensor-board. The CPU runs from a 128kB flash memory and has 512kB memory for sensor readings. The radio is an IEEE 802.15.4 radio and offers high speed (250 kbps) communication [**?**]. A wide variety of sensor-boards are available for the motes. The sensor-boards include, but are not limited to, seismic activity, temperature, humidity and light. This enables the motes to monitor their environment in a variety of ways.

Wireless sensor networks (WSN) are self-organizing networks of motes. They are used to monitor the environment for events such as forest fires or enemy troop movements. A large number of motes are spread over the area to be monitored. Upon activation, the motes self-organize into a network, which connects to the users via a powerful base station in order to achieve a common goal [**?**]. As each mote surveys the area within its sensing range, the data is sent towards the base station along a multi-hop path. A WSN is able to remotely cover a large sensing area since these low-cost motes organize into a multi-hop network without human assistance.

The lifetime of a WSN is limited. Each MICAz mote is powered by two AA batteries. The motes can operate under various settings, including active listening mode, sleep mode and idle mode. The average lifespan of a mote which has a random activity pattern is nine months [1]. Under normal circumstances, it will not be feasible to replace used batteries in motes. In order to extend the lifetime of the WSN, energy saving techniques must be applied.

Communication via the on-board radio is the most expensive operation for the sensor nodes [2]. Protocols such as LEACH [3], TOPDISC [4] and HEED [**?**] reduce energy consumption and increase the lifetime of the network. The basic idea in these protocols is to cluster sensors into groups, so that sensors communicate only to their cluster head. The cluster heads then communicate the aggregated information to the processing center. Clustering has been shown to greatly reduce power consumption, is easily scalable, and is robust in case of node failures [3]. A good clustering scheme is one that takes into account one or more of the following: communication range,

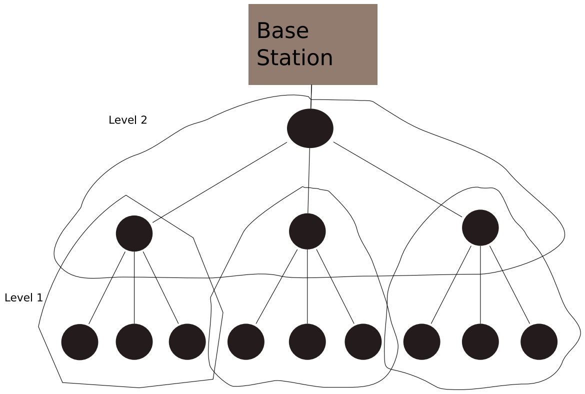


Figure 1.1. Motivating Example

number and type of sensors, geographical location and remaining energy. Clustering and proper cluster head selection in order to maximize the lifetime of the network is an important consideration when designing protocols and algorithms for sensor networks.

In-network aggregation in WSN is another approach that allows for a large savings of energy. In in-network aggregation, intermediate results are calculated along the multi-hop path whenever two or more messages are routed along the same path. Depending on the routing structure, energy savings may be by as much as eight times [3]. Consider the network depicted in Figure 1.1. Without aggregation, a total of 9+12+13 = 34 messages are sent; with aggregation, only 9+3+1 = 13 messages are sent. Through the sample network in Figure 1.1 it is clear that in-network aggregation greatly reduces the number of messages sent in a WSN, which leads to large energy savings. In-network aggregation and clustering algorithms can be used in conjunction; many aggregation algorithms naturally build upon the clustered network. Clustering and aggregation algorithms combine the necessary energy savings that will ensure a long network lifetime.

Most wireless sensor network applications are deployed in environments which can affect the motes. The effects of wind, rain, heat or sand on the motes is unpredictable. Sensor networks may be used in hostile environments and be subject to attacks by enemies. An attacker may be able to gain physical access to a sensor node and introduce nodes into the network or simply inject messages into the communication channel. Due to such manipulation by an attacker, a sensor network may generate incorrect readings or aggregates. Corrupted or spurious sensors may fail to participate in the common tasks of a sensor network. In secure situations sensors may be corrupted due to the physical effects of wind, rain, or heat. Water could penetrate the physical housing and cause the CPU to generate incorrect readings or calculate an aggregate incorrectly.

Security in WSNs includes confidentiality, integrity and availability. Confidentiality in sensor networks is accomplished by preventing outsiders from eavesdropping on transmissions. Integrity implies that any aggregate result is made up of only legitimate data without faulty inclusions or additions, and also that internal, corrupted sensors cannot interfere with operations. Availability in sensor networks is of great concern to the user of the network. Any useful sensor protocol needs to ensure that the generated data reaches the user in a timely manner. Unfortunately, many existing security primitives cannot be used in sensor networks, either because the computing power of the sensors is too limited or the additional work created by the protocols causes excessive network traffic [**?**].

Security is even more of a challenge in WSNs when in-network aggregation is utilized. A corrupted sensor may appear to participate in the mission of the network but falsify sensor readings, improperly apply an aggregation function, exclude legitimate messages from the aggregate result or create a fictitious result. A sensor corrupted by an attacker may behave in this way in order to get the base station to accept an incorrect result that is favorable to the attacker. Hence in order to secure data aggregation in a sensor network, we must not only provide protection against eavesdroppers, but we should also prevent intermediate sensors from having access to the data.

This dissertation is organized as follows. First, a thorough review of the current literature on clustering and secure data aggregation. Next we introduce the “Energy Constrained Clustering” in Paper 1. We expand on clustering with Paper 2, which gives several extensions to the original ECDS algorithm. In Paper 3 we present a practical algorithm for data security (PADS) in wireless sensor networks. The last paper, Paper 4 introduces a novel way of proving aggregate digital signatures in wireless sensor networks.

2. REVIEW OF LITERATURE

2.1. CLUSTERING

Existing work on clustering algorithms can be classified into several categories: k-hop-cluster formation, where each node is at most k hops from its cluster head, connected dominating set (CDS) and weakly connected dominating set (WCDS) based algorithms, and general clustering algorithms.

2.1.1. DOMINATING SET

A connected dominating set based clustering algorithm enables the cluster heads to communicate directly with each other. This allows for a virtual back-bone formation in the network. In applications in which the cluster heads need to be able to work together to complete a task, connected dominating sets offer an advantage. The weakly connected dominating set slightly relaxes the requirement of connected cluster heads and allows them to be, at most, k-hops away from the next cluster head. The work presented in [5, 6, 7, **?**, 8] are examples of connected and weakly connected dominating set algorithms.

The algorithm HEED presented in [**?**] selects cluster heads according to residual energy and node proximity to neighbors or node degree. HEED uses several fixed level of transmission power levels. One level is used to communicate with intra-cluster nodes and another is used by cluster heads to communicate with other cluster heads. A cluster head aggregates the data and sends it via other cluster heads to the base station. Messages are send toward the base station and can be forwarded by any node, not just the cluster heads.

In [5], a series of approximation algorithms for finding a small, weakly-connected dominating set (WCDS) in a given graph is presented for use in clustering mobile ad hoc networks. The main contribution of the work is a completely distributed algorithm for finding small WCDS. The performance of the algorithm is shown to be very close to that of the centralized approach.

In [6], the authors provide three approximation algorithms for the minimum connected dominating set (MCDS) in mobile ad hoc networks. The algorithms provide approximation guarantees of $2H(\Delta) + 1$ and $2H(\Delta)$, where $H(\Delta) = \sum_{i=1}^{\Delta} 1/i \leq \ln \Delta + 1$. The guarantee of $c + 1$, applies to graphs where the maximum degree is Δ and c is some constants such that $\Delta \leq c$.

The connected minimum dominating set is considered in [7]. The authors provide two approximation algorithms which achieve approximation factors of $2H(\Delta) + 2$ and $H(\Delta) + 2$, where Δ is the maximum degree in the graph and H is the harmonic function. The authors also consider the traveling tourist problem and the Steiner CDS and provide the corresponding approximation algorithms.

Several distributed poly-logarithmic time algorithms are presented in [8]. The algorithms compute connected and weakly connected dominating sets with an approximation factor of $O(\log \Delta)$, where Δ is the maximum degree of the graph. The authors also show distributed algorithms to construct low-stretch dominating sets. A low-stretch dominating set is one in which every pair of nodes has dominators of no more then $O(\log n)$.

2.1.2. GENERAL CLUSTERING

In [9], cluster heads are chosen so that the energy consumption over the entire network is even, ensuring that the network lives as long as possible. A fixed number of cluster head candidates are selected and the cluster heads with the most residual energy are chosen from that set. A node will chose a cluster head to ensure the overall energy consumption in the entire network is even.

In [10], a new, fully distributed approximation algorithm based on LP relaxation techniques is presented. For an arbitrary parameter k and maximum degree Δ, the algorithm computes a dominating set of expected size $O(k\Delta^{2/k} \log \Delta \, |DS_{OPT}|)$ in $O(k^2)$ rounds where each node has to send $O(k^2\Delta)$ messages of size $O(\log \Delta)$. This is the first algorithm that achieved a non-trivial approximation ratio in a constant number of rounds.

The work described in [11] is a primal-dual based distributed algorithm for the weighted, capacitated vertex cover problem. In [11] the each vertex is assigned a weight, as well as a capacity, and the goal is to minimize the sum of the weights without exceeding the capacity of any vertex. The authors provide a $(2+\epsilon)\,OPT$ approximation algorithm. Additionally the running time of the algorithm is shown to be $O\left(\log\left(nW\right)/\epsilon\right)$, where n is the number of nodes and $W = wt_{max}/wt_{min}$ is the ratio of the largest weight to the smallest weight.

A randomized distributed algorithm is presented in [12]. The algorithm runs in $O(\log n \log \Delta + 1)$ rounds and the size of the dominating set obtained has a high probability of being within $O(\log n)$ of the optimal. Each round has a constant number of messages that are exchanged among neighbors. The authors also cover a generalization to the weighted dominating set and as well as the case in which each node is required to be covered by multiple nodes.

2.1.3. K-HOP CLUSTERING

The work in [13, 14, 15] features k-cluster algorithms. After the algorithm is executed, each node is either a cluster head or at most k-hops from its cluster head.

In [13], an identity-based heuristic to form k-clusters in wireless ad-hoc networks is presented in which d is a parameter. When the heuristic terminates, a node is either a cluster head or, at most, k hops away from its cluster head. This heuristic extends its former ones which restricted themselves to 1-hop clusters, thus helping to reduce the number of cluster heads.

A fast, distributed algorithm is presented in [15]. It is used to compute a small k-dominating set D (for any fixed k) and its induced graph partition (which breaks the graph into radius k clusters centered around the vertices of D). The time complexity of the algorithm is $O(k\ \log^* n)$ where $\log^*$ is the inverse Ackermann function.

For the special family of graphs that represent ad hoc wireless networks modeled as unit disk graphs, [14] introduces a two phase distributed polynomial time and message complexity k-clustering approximation solution with $O(k)$ worst case ratio over the optimal solution.

2.2. IN-NETWORK AGGREGATION

Once the sensor network is properly clustered, the sensor nodes can work on achieving their goal of securely transmitting data to the base station. Existing work on data security in wireless sensor networks can be classified based on the number of aggregators that are supported. Some algorithms support only one aggregator, the base station. In this case data are securely transmitted to the base station, which applies the aggregation function. By using end-to-end security, in which each packet is secured by the sender and can only be deciphered by the base station, total security is provided. However, this security is provided at a high communication cost of $O(N \log N)$ at best and $O(N^2)$ at worst depending on the topology of the network.

Other algorithms provide for multiple aggregators, a subset of the sensor nodes in the network. Additionally, it is possible to distinguish between multiple aggregators on the same level of the hierarchy and multiple aggregators on multiple levels of the hierarchy. In order to accomplish data aggregation, the data are no longer encrypted end-to-end, but rather hop-by-hop. Data confidentiality now needs to include assurance to the base station and the sending nodes that the data were included in the aggregate result. Data integrity must now mean that the data were included as-is, an assurance that the aggregate was calculated from exactly the inputs, nothing more, and nothing less, without modification to the inputs.

Clustering algorithms, where the cluster head aggregates the information in its cluster prior to sending the aggregate information to the base station, are an example of a multiple aggregator, single level algorithm. Depending on the number of hops the cluster head is away from the base station, a clustered aggregation algorithm can provide for large savings in the communication costs over base station aggregation.

A tree hierarchy in which each node combines its own reading with those of its descendants is an example of a multiple aggregator, multiple level algorithm. Multiple aggregators at multiple levels of the structure lead to the largest energy savings. Each node may receive multiple transmissions but will only send one message to its parents. However, multiple aggregators at multiple levels also have the highest security risk because each aggregator may malfunction and corrupt the aggregate result.

2.2.1. SINGLE AGGREGATOR

The development of TinySec was an important first step toward providing security in wireless sensor networks [16]. TinySec showed that while sensors have limited processing abilities, they are capable of applying adequate levels of security. TinySec implemented public key cryptography using SkipJack or RC5 and link-layer security. TinySec uses an 8-byte key and can be used either as hop-by-hop or end-to-end security. If hop-by-hop security is used, then no guarantees can be made regarding compromised aggregators. If end-to-end security is used, then aggregation is only possible at the receiving point. However, TinySec is important because many of the approaches that have followed it rely on encryption/decryption within their algorithms. The simulations performed with networks of 36 nodes during the development of TinySec showed that TinySec adds a 10% overhead in energy consumption, latency and bandwidth utilization.

Single aggregator models provide total security for confidentiality, integrity, and availability by employing end-to-end security. This gives the base station the knowledge that every sensed value received has been included in the calculation because intermediate nodes cannot modify the values in transit. However, availability may suffer because each node contributes its own reading and forwards any messages received, depending on the layout of the network, nodes at higher levels of the hierarchy will expend large amounts of energy to forward messages. In the worst case, the sensors are in a straight line, each with one parent and at most one child. Each node will transmit one more message than it received, and the last node will transmit N messages, where N is the number of nodes in the network. The total number of messages transmitted will be bound by $O(N^2)$. At best, each sensor will be able to transmit directly to the base station, incurring a communication cost of $O(N)$ at most. The edge complexity can be as high as $O(N)$ or as low as $O(1)$ for either of the extremes, with a typical sensor network somewhere in between. Edge complexity refers to the maximum messages sent along a single edge.

2.2.2. MULTIPLE AGGREGATORS

A single-level multi-aggregator algorithm allows for multiple aggregators. Each aggregator sends its aggregate result to the base station. A clustered topology is an example of such a setup. The cluster head calculates the aggregate for its cluster and then communicates that result to the base station. The base station takes that information from all the cluster heads and calculates the total aggregate for the entire network. Such a setup is also useful in sensor networks with reactive sensing capabilities. Instead of generating a steady stream of data, reactive sensor networks send data only in response to an event; often only a few regions of the network are active at any given time. Clustered topologies allow for aggregates to be calculated in the affected regions and the result communicated to the base station.

The authors in [17] proposed the use of interactive proofs to force the aggregator to show that the calculated aggregate is a good approximation of the true value. Three phases are involved in the aggregate-commit-prove (ACP) protocol. During the first phase, the aggregator collects the data and computes the aggregate result. The aggregator is able to verify the authenticity of each message because a shared key exists between the aggregator and each other node. During the second phase, the aggregator calculates a commitment, which is used during the third phase to prove that the aggregate is valid. The base station checks to see if the aggregator is cheating by observing whether the result received from the aggregator is close to the correct result aggregated from the committed data. This approach has an overall message complexity of $O(3N)$ and an edge complexity of $O(1/2N)$, depending on how many nodes the base station queries to verify the commitment.

In SecureDAV, the authors proposed a secure data Aggregation and Verification Protocol using elliptic curve cryptography (ECC) and a threshold digital signature scheme (EC-DSA) [18, **?**]. SecureDAV is a two-part protocol. During the initial phase, each cluster generates a secret cluster key using verifiable secret sharing (VSS). At the end of the protocol, each sensor in a cluster will have a share of the secret cluster key. Because each node has only a share of the cluster key, the cluster key cannot be determined by an attacker. The secret cluster key is used to digitally sign messages to the base station, which can verify the signature using the corresponding public key. In a threshold signature scheme, the signatures generated by each holder of a share

are combined into a full signature, which is equivalent to a signature with the full key by a single signer. In the second part of the protocol, each sensor sends its reading to the cluster head. The cluster head calculates the average and then broadcasts the average to all nodes in the cluster. Each node checks to see if the average is within a small threshold of its own reading. If the test passes, then the sensor signs the average with its share of the signature and sends the partial signature to the cluster head. The cluster head combines the signatures into a full signature and sends the average along with the signature to the base station. The base station can verify the signature using the public key, and thus it is assured that the average is acceptable to the sensor nodes in the cluster. To provide data integrity, a Merkle Hash Tree is generated by the cluster head from the encrypted readings of the sensors, the base station uses the Merkle Hash Tree and repeated queries to the cluster head to verify integrity. The major limitation of SecureDAV is that it is designed to work only with the calculation of the average. Calculating any other function, such as the SUM is impossible using this protocol, because a node will be unable to tell if its reading has been added to the sum. This algorithm has an overall message complexity of $O(3N)$ and an edge complexity of $O(1/2N)$, again, depending on how many messages are necessary for the base station to verify the integrity of the result.

2.2.3. HIERARCHICAL AGGREGATION

Multiple aggregators at multiple levels of the hierarchy will lead to the biggest savings over aggregation at the base station. Consider Figure 2.2, in which no aggregation will result in a total of 42 messages sent in the entire network, 14 of which will be the edge complexity cost on the link to the base station. Opportunistic aggregation will result in 14 messages sent, one from each node. The edge complexity cost in the entire network is 1. Clearly, hierarchical aggregation results in overwhelming communication costs savings and also has very good edge complexity costs. However, the security in hierarchical aggregation is very much a challenge. The base station now needs to be assured that each and every aggregator correctly computed the aggregate. Each sensor needs to know that its reading was properly carried through all levels of aggregation. Additionally, in corruption due to attacks,

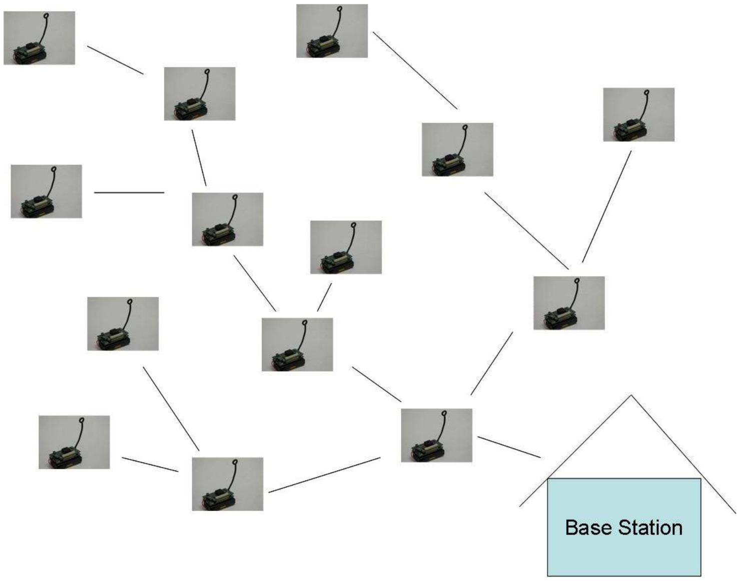


Figure 2.2. A sample hierarchical sensor network

several aggregators may decide to collude together to overcome the restrictions of the algorithms.

In [19], the authors presented a secure aggregation algorithm, which is secure against injected nodes and compromised nodes. The algorithm delays aggregation by one level in the hierarchy. A node sends its data secured with a message authentication code (MAC) to its parent at level i-1. The parent computes the aggregate result and sends it, plus the data received from the children, to its parent at level i-2. It retains a copy of the original data along with the MAC. The nodes at level i-2 can now also calculate the same aggregate as the nodes at level i-1, thus ensuring that the nodes at level i-1 correctly computed the result. This process of sending the calculated result along with the data that went into the calculation continues until the data reach the base station. The algorithm also employs delayed authentication using μTESLA [20], which means that the base station reveals the appropriate keys after a delay. The nodes can then use these keys to verify the authenticity of the MACs received from their descendants. However this algorithm does not provide any security against colluding nodes at different levels of the hierarchy. If a node at level i-1 and a node at level i-2 collude to falsify an aggregate result, then this algorithm does not provide protection. However, it does provide for multiple levels of aggregation, and no limitations exist on the types of aggregates that can be computed. The overall message complexity for this approach is $O(N)$ with an edge complexity of $O(1)$; however, colluding nodes can defeat this algorithm.

Chan et al. first defined a direct data injection attack, in which an attacker controls any number of sensors and causes them to submit a value that is different from the sensed value to the in-network aggregation process [21]. They introduced an algorithm that is secure against aggregator misbehavior in the sense that an attacker cannot deviate the resulting aggregate any more than would be possible with a direct data injection attack. This means that an adversary cannot gain an advantage by controlling an aggregating node over controlling nodes that only input their data. This algorithm will work with any arbitrary tree topology and is able to cope with any number of malicious nodes. The algorithm is explained in detail for a SUM aggregation, but it is also shown how a similar approach would work with MEDIAN, COUNT, and AVERAGE aggregation. The message complexity for this approach is $O(\log^2 N)$, while the edge complexity is bounded by $O(\log^2 N)$. The algorithm works by building a commitment structure on the aggregate and by decoupling the physical communication network from a logical aggregation network. Instead of relying solely on the base station for result checking as in [17], the result checking is fully distributed and becomes part of the responsibilities of each sensor node. The base station distributes the final commitment to the network. After receiving the partial commitments from off-path nodes, a node is able to verify that its inputs were correctly applied. The node then informs the base station whether it agrees with the aggregate. The major disadvantage of this approach is that while the edge complexity is only $O(\log^2 N)$, the overall communication complexity of this approach is $O(N \log^2 N)$. Additionally, this approach will only work on certain aggregate functions.

I. ENERGY CONSTRAINED CLUSTERING FOR WIRELESS SENSOR NETWORKS

Julia Albath and Sanjay Madria
Missouri University of Science and Technolgy
Department of Computer Science
julia.albath@acm.org, madrias@mst.edu

Mayur Thakur
Google Inc.
Mountain View, CA
maythakur@google.com

Abstract— **Using partitioning in wireless sensor networks to create clusters for routing, data management, and other protocols has been proven as a way to ensure scalability and to deal with sensor network shortcomings such as limited communication ranges and energy. Choosing a cluster head within each cluster is important because cluster heads use additional energy for their responsibilities and that burden needs to be carefully passed around. Many existing protocols either choose cluster heads randomly or use nodes with the highest remaining energy. We introduce the energy constrained minimum dominating set (ECDS) to model the problem of optimally choosing cluster heads with energy constraints. We show its applicability to sensor networks and give an approximation algorithm of $O(\log n)$ for solving the ECDS problem. We propose a distributed algorithm for the constrained dominating set which runs in $O(\log n \log \Delta)$ rounds with high probability. We experimentally show that the distributed algorithm performs well in terms of energy usage, node lifetime, and clustering time and, thus, is very suitable for wireless sensor networks.**

I. INTRODUCTION

A wireless network consists of a large number of small sensors with low-power transceivers. These sensors are an effective tool for gathering data for a variety of purposes, such as border protection, surveillance of forests for fire, and tracking of animal movements. The data collected by each sensor is communicated via a multi-hop path in the network to a single processing center, the base station. The base station uses all reported data to determine the characteristics of the environment or detect an event.

Communication via the on-board radio is the most expensive operation of the sensor nodes [1]. In radio communications, the signal strength decreases proportional to the square of the propagation distance [2]. In other words, to have the same signal strength reach twice the distance, four times the amount of energy is required. Protocols such as LEACH [3], and those described in [4] and [5] reduce energy consumption and increase the lifetime of the network. The basic idea in these protocols is to cluster sensors into groups and to choose a cluster head such that sensors communicate only to their cluster head. The cluster heads then communicate the aggregated information to the processing center. Clustering has been shown to greatly reduce power consumption, is easily scalable, and is robust in the face of node failures [3]. A good clustering scheme takes into account one or more of the following: communication range, number and type of sensors, geographical location, and remaining energy [6]. Clustering and proper cluster head selection in order to maximize the lifetime of the network are important considerations when designing protocols and algorithms for sensor networks [7].

A sensor network can be expressed as a graph $G = (V, E)$, where each of the vertices represents a sensor node and there is an edge between two vertices if their corresponding sensor nodes are within each other's communication range. A dominating set of a graph $G = (V, E)$ is a subset $V' \subseteq V$ such that each $x \in V - V'$ has a neighbor in V'. The assignment of nodes to cluster heads is often modeled as a dominating set (DS) problem [8]. The minimum dominating set problem is NP-complete for general graphs [9] and remains NP-complete for planar graphs, unit disk graphs, bi-partite graphs, and chordal graphs, but it does admit a Polynomial Time Approximation Scheme (PTAS) for planar graphs and unit disk graphs [10]. The dominating set problem models the optimization problem of finding a small number of cluster heads.

Clustering in sensor networks and in Mobile Ad-Hoc Networks (MANET) benefits from using a dominating set approach. The dominating set approach leads to better clustering because dominating set based clustering can be executed in a constant number of rounds [11]. A DS based approach works because every node in the network is either a dominating node or is only one hop from a dominating node [12]. Single-hop

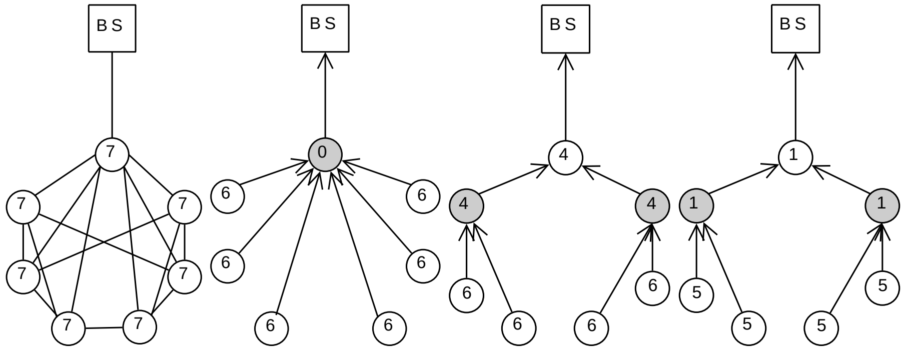


Fig. 1. Energy Constrained Clustering Example: (a) Original Network (b) Without Constraints After One Round (c) With Constraints After One Round (d) With Constraints After Two Rounds

communications within clusters is appropriate because most nodes will be close to their cluster head and their links are of good quality [13].

Cluster heads spend additional energy on message transmission, so a small set of cluster heads might not be optimal from a network survivability standpoint. For instance, using a dominating set as the set q of cluster heads comes with the disadvantage that the network might lose a few cluster heads and become fragmented fairly soon. Consider the graph shown in Figure 1. Let's assume that one unit is used for each receive or send and the nodes in the dominating set combine all received data into one outgoing message. Each node has the same amount of energy (7 units) (Figure 1(a)) at the beginning. The optimal dominating set is one node (Figure 1(b)), however the network will become disconnected after only one time step. On the other hand a slightly non-optimal dominating set using the heuristic "Don't give a cluster head more than three nodes" results in a network that survives two time steps as shown in Figures 1(c) and 1(d) (the shaded nodes represent the cluster heads).

A wireless sensor has many constraints, energy being one of the important ones. Other constraints include bandwidth, storage and computational abilities. A wireless sensor networks needs to consider these and other constraints when choosing cluster heads and assigning nodes to clusters. For example, each node sends one packet per round to its cluster head. The length of a round is limited; this limits the number of nodes a cluster head can support. Additionally, a cluster head has to store received messages until they are combined at the end of each round; this also limits how many nodes a cluster head can support. For this work, we have chosen to concentrate on the limited energy available to each sensor and the natural limitations on the size of each cluster that follow.

Motivated by the above examples, we introduce the energy constrained minimum dominating set of a graph in order to achieve these objectives. The contributions of this paper are summarized as follows:

- We introduce and define the Energy Constrained Dominating Set (ECDS) Problem in Section II.
- We describe the problem of developing clusters so that the energy consumption during each round of processing is minimized without exhausting the available resources of any given node. In the energy constrained DS problem, we are given integer constraints on each node that denote the maximum number of relay links a node can handle if it is chosen as a cluster head. The objective is to minimize the size of the cluster head set subject to the constraint that no node has more work then it can handle and that every node is either in the constrained DS or one hop from such a node. We give a centralized algorithm in Section IV.
- Since the clustering algorithm typically runs repeatedly (for example, when nodes move or a cluster heads dies, a distributed algorithm is much more practical than a centralized algorithm. We give a practical distributed algorithm in Section V.
- We prove that the distributed algorithm runs in $O(\log n \log \Delta)$ rounds with high probability, where Δ is the maximum degree of a node in the graph. We provide the proof in Section VI.
- We support our theoretical analysis with extensive simulations using TOSSIM. We compare the performance of the distributed ECDS algorithm to the HEED algorithm [14]. HEED selects cluster heads according to residual energy and with node proximity to neighbors or node degree. Our protocol uses local information about the connectivity of each node and the connectivity of its neighbors in addition to the residual energy to decide which node should become a cluster head. ECDS takes less time and fewer rounds to cluster the network, allowing more messages to

reach the base station. For the scenarios in our study, ECDS clustering takes 3.5 rounds or less, compared to 4.5 rounds or less for HEED. The number of cluster heads is as expected and the number of nodes in each cluster remains steady. Our algorithm results in very few single node clusters. The number of cluster heads, the size of the clusters and the number of clusters which contain only the cluster head are much better in ECDS then in HEED. The lifetime of the sensor network, measured in terms of time of first node death and time of last node death, is better in ECDS then in HEED. While the overall energy consumption is slightly higher for ECDS, when considering that ECDS produces more useful data, the energy consumption per message is much lower. The results of said simulation are available in Section VII.

II. DEFINITIONS AND NOTATIONS

This section describes the notations used in the rest of the paper and defines the dominating set, network clustering, and energy constrained connected dominating set.

Definition 2.1: For a graph G and a subset S of the vertex set $V(G)$, denote by $N_G[S]$ the set of vertices in G which are in S or adjacent to a vertex in S. If $N_G[S] = V(G)$, then S is said to be a *dominating set of* G.

Definition 2.2: Given an undirected graph $G = (V, E)$, and, for each $v_i \in V(G)$, a constraint $r(v_i) \in \mathbb{N}$, the energy-constrained dominating set (ECDS) of G is a pair (S, C), where C is an assignment from $x \in S$ to $V_x \subseteq V$ such that (a) $\{V_x | x \in S\}$ is a partition of V, (b) for each $x \in S$, $x \in V_x \subseteq N_G[\{x\}]$, and (c) for each $x \in S$, $\|V_x\| \leq r(x) + 1$.

In the definition of ECDS, we assume that when a node is selected as a cluster head, it includes itself in the cluster. (See part (b) of the definition.) Also note from the "+1" in condition (c) that we allow a node to cover itself for free. That is, the constraint $r(x)$ for x denotes the maximum number of nodes that x can cover in addition to itself.

ECDS is also related to, but different from, the Network Clustering problem [15]. ECDS has a constraint parameter that is not present in Network Clustering. Also, the clusters must form a partition in ECDS, whereas they may overlap in the Network Clustering problem. The general dominating set can be described as a constrained dominating set where each constraint is equal to n, the number of nodes in the graph. It trivially follows that the constrained minimum dominating set is NP-complete. The minimum dominating set in the general form has approximation algorithms of within $1 + \log \|V\|$. Since the constrained dominating set problem is a special case of the general dominating set problem, no improvement on these bounds will be possible.

In Wireless Sensor Networks (WSN) coverage generally means ensuring that the entire area has proper sensor distribution to ensure even sensing. In this work, we define coverage as one node's ability to handle the relaying of messages to and from other nodes in its cluster.

III. RELATED WORK

In [16], cluster heads are chosen so that the energy consumption over the entire network is even, ensuring that the network lives as long as possible. A node will chose a cluster head to ensure the overall energy consumption in the entire network is even. Our algorithm, on the other hand, requires only local information about the topology and residual energy.

In [17] each vertex is assigned a weight, as well as a capacity, and the goal is to minimize the sum of the weights without exceeding the capacity of any vertex. The authors provide a $(2 + \epsilon) \, OPT$ approximation algorithm.

A randomized distributed algorithm that runs in $O(\log n \log \Delta + 1)$ rounds and where the size of the dominating set obtained is, with high probability, within $O(\log n)$ of the optimal, is presented in [18]. Our distributed algorithm is based on this algorithm and extends its ideas to vertices with constraints and applies it to wireless sensor networks.

A fast, distributed algorithm is presented in [19]. It is used to compute a small k-dominating set D (for any fixed k) and its induced graph partition. The time complexity of the algorithm is $O(k \ \log^* n)$, where $\log^*$ is the inverse Ackermann function.

In [20], a series of approximation algorithms for finding a small, weakly-connected dominating set (WCDS) in a given graph is presented for use in clustering mobile ad hoc networks. The main contribution of the work is a completely distributed algorithm for finding small WCDS. Our work focuses on wireless sensor networks and creates connected dominating sets.

The connected minimum dominating set is considered in [21]. The authors provide two approximation algorithms which achieve approximation factors of $2H(\Delta) + 2$ and $H(\Delta) + 2$ where Δ is the maximum degree in the graph and H is the harmonic function.

IV. CENTRALIZED ALGORITHM

We now describe a centralized algorithm for the constraint dominating set problem. A general overview of Algorithm 1 follows. The input is an undirected graph $G = (V, E)$ and constraints $r(v)$ for each $v \in V$. $V' \subseteq V$, the set of vertices dominating the graph, is initially empty.

Many applications of sensor networks occur in heterogeneous networks, where the sensor have the support of several powerful computers in addition to the base station. The centralized version of the algorithm corresponds to a sensor network in those settings.

Each vertex x has a capacity $r(x_i)$. Since sensor networks are resource constrained, we chose to use the energy as a representative for the constraint. A function that maps the available resource to the constraint is required.

In each iteration, the greedy algorithm makes two choices. First, it chooses a node v from the set $V - V'$ and adds it to V'. Second, given the updated set V' it chooses which vertices this set will cover. In fact, v is chosen as the node $x \in V - V'$ such that the maximum number of nodes covered (subject to constraints and over all possible assignments of nodes to dominators) by $V' \cup \{x\}$ is maximized. Thus, in each iteration, we first compute the maximum coverage of $V' \cup \{x\}$ (denoted N_x), for each $x \in V - V'$, and then add to V' the node v such that N_v is maximum.

The maximum coverage N_x for each vertex $x \in V - V'$ is calculated using the max-flow algorithm. Let $S = V' \cup \{x\}$ and let $T = V - S$. Define a directed graph G' as follows: An artificial source s is connected in turn to each vertex $y \in S$ with a capacity equal to $r(y)$, the constraint of y. Next, each vertex $y \in S$ is connected to its neighbors in T with a capacity of one and each node in T is connected to an artificial sink t, also with capacity one. We compute the max-flow from s to t in G'. Since the incoming capacity to each node $y \in S$ is equal to the constraint, no node $y \in S$ will be able to provide flow to more nodes than its constraint. Since all the capacities are integral, we can efficiently find a max-flow that has integral flow on all edges [22]. Because the capacity of each edge coming into T is one and the outgoing capacity of the edge from any node in T to t is one, it follows that each node $z \in T$ will satisfy exactly one of the following: (A) z has exactly one incoming edge with flow $= 1$ (and the rest with 0 flows) or (B) all edges incoming to z have 0 flow. We will say that a node $z \in T$ is covered if it is of type (A) and we will say that y is covered by the (unique) node w such that the flow on (w, z) is 1. Let C be this partial assignment (partial clustering) of nodes in T to nodes in S.

Claim 4.1: Assignment C obeys all constraints $r()$. Further, the number of nodes in T that are covered in assignment C is is the maximum number of nodes that can be covered (subject to constraints $r()$) by nodes in S.

Proof: The capacity from s to a node $w \in S$ is equal to the constraint $r(w)$, and the flows are integral, w dominates at most $r(w)$ nodes in T. Thus, the constraints are obeyed.

Let k be the number of T nodes that are covered in the assignment C. By our construction, k is the max flow of G'. To prove that C is an optimal assignment, consider an assignment C' of T nodes to S nodes such that the number of nodes covered (subject to constraints $r()$) is greater than k. Since this assignment can be converted into a valid flow by giving a flow of 1 to each edge (w, v) such that $w \in S$ and $v \in T$ and w dominated v (equivalently, v is assigned to w) in C'. The flow achieved is greater than k, which contradicts the assumption that the max flow in G' is k. □

In each iteration, the algorithm tests each $x \in V - V'$, and includes the node that has the largest max-flow. Note that the partial clustering is recomputed at every stage. Thus, a node a that is assigned to clusterhead c at stage i might not be assigned to c at step $i + 1$. In fact, a might not even be covered at step $i + 1$.

Theorem 4.2: There is a $O(\log n)$ approximation algorithm for ECDS.

Proof: The proof is a simple modification of the standard set cover proof. Let G be the input graph and $r(v)$ be the constraint on node v. Let X be an optimal energy-constrained dominating set of size OPT. Note that there is an assignment (clustering) C of nodes in $V - X$ to nodes in X such that all constraints are obeyed. Let Y be the output of our algorithm.

Order the nodes in V by the step in which they are covered. (For nodes that are covered in the same step, order them arbitrarily.) Consider the jth node. At the beginning of the step k in which j was covered, there were at least $n - j$ uncovered nodes. Let

Algorithm 1 Constraint Minimum Dominating Set Algorithm

```
Require: Graph $G = (V, E)$, $V = \{v_1, v_2, \ldots, v_n\}$, constraint t($v_i$) on vertices
Ensure: $V' \subseteq V$, set of currently chosen vertices
Connect $V$ to sink with capacity $= 1$
$V' = \emptyset$
$Covered = \emptyset$
while $Covered \neq V - V'$ do
    for each vertex $x \in V - V'$ do
        Connect $x_i$ to source with capacity = $t(x_i)$
        Connect $x_i$ to each $y \in N(x_i)$ with capacity $1$
        Connect $y \in N(x_i)$ to the terminal with capacity $1$
        Calculate $N_{x_i}$ = maximum coverage of $V' \cup \{x_i\}$
    end for
    Pick $x_i$ such that $N_{x_i}$ is maximum
    $V' = V' \cup x_i$
    Update $Covered$ with $y \in V$ such that there was flow from $x$ to sink
end while
Output $V'$
```

the partial dominating set at the beginning of step k be S. Then $S \cup X$ will cover all the nodes since X by itself covers all the nodes. Let x be the node selected at step k. Then, node x covers at least $c_x = \frac{n-j}{OPT}$ nodes. Associate a weight of $1/c_x$ with each node that x covers. Then, the total weight associated with all nodes that x covers is at least 1. Also, the jth node gets a weight of at most $\frac{OPT}{n-j}$. There is an upper bound of the size of Y in our algorithm, achieved by adding the weights of all nodes. This upper bound is $\sum_{j=0}^{n-1} \frac{OPT}{n-j} = OPT[\frac{1}{n} + \frac{1}{n-1} + \ldots + \frac{1}{1}]$. Thus, the size of Y is $O(\log n) * OPT$. □

V. DISTRIBUTED ALGORITHM

This algorithm is a modification of the local randomized greedy (LRG) algorithm from [18]. The LRG algorithm is a modification of the distributed version of the greedy algorithm of [23]. To enable comparison, we first informally describe the LRG algorithm and then describe our modification to the algorithm.

The LRG algorithm proceeds in rounds. At the start of a round, each node that is not already in the dominating set decides whether it wants to be a candidate dominator in that round. Candidacy is determined by letting only nodes that can cover a large number of nodes be candidates. These nodes must have a large number of neighbors remaining. Note that a node can be covered by multiple candidates. A node defines its *support* as the number of candidates that cover it. Each node is selected for the dominating set with a probability that is the inverse of the median of supports of all nodes that it covers. Once a node is selected, its neighbors are considered "covered". The round has ended; if uncovered nodes remain, another round starts.

The reason for using the median of supports follows: If we pick all the candidates, then we might pick too many nodes for the dominating set. If we pick only one candidate, then we may require too many rounds.

We will now describe our modification of the LRG algorithm and show an effective $(O(\log n \log \Delta))$ randomized distributed algorithm for ECDS. In this version, the algorithm is required to obey the constraints *in expectation*. In particular, for each node u, $E[\#$ nodes covered by $u \mid u$ is selected as a cluster head$] \leq r(u)$. In fact, our algorithm obeys the constraint in expectation in an even stricter sense. Note that in the above formulation, it is possible for certain sets of cluster heads to grossly violate the constraints. For example, the above constraint allows an algorithm to have the following behavior: Whenever the algorithm outputs the set $\{v_1\}$ as the cluster head, all the constraints are violated grossly. Our algorithm does not have this undesirable behavior. In fact, our algorithm obeys the following: Let u be an arbitrary node and let $U \subseteq V - \{u\}$. Then, $E[\#$ nodes covered by $u | U \bigcup \{u\}$ is selected as a cluster head$] \leq r(u)$. This basically says that the nodes obey the constraints in expectation independent of one another.

The following issues must be handled:

1. What is the support of a node?

2. Given that we select a node x to be a dominator, how do we select which nodes to cover/dominate from among the neighbors of x?

To address issue 1, we say that the *constrained span* $c(x)$ of a node x at a given step in our algorithm is the smaller of the following two quantities: the number of uncovered neighbors of x and the constraint of x. (The set of neighbors of a node x includes x and all nodes with which x shares a high quality communication link/edge). Let x be a candidate and let y be a node that is adjacent to x. The *out-support* $s_{out}(x)$ of x is the ratio of the constrained span $c(x)$ to the number of uncovered neighbors of x. For example, if a candidate x has 5 uncovered neighbors and the constraint of x is 3, then $c(x) = 3$, $s_{out}(x) = 3/5$. The out-support of x is the fractional support a node gives to each of its neighbors. The *in-support* $s_{in}(y)$ of y is the sum of the out-supports of each neighbor of y. Thus, the in-support of a node is the total support a node will get if all its neighbors are dominators and each gives fractional support to all the neighbors. Roughly speaking, the larger a node's in-support, the larger the probability that it will be covered in a randomly chosen dominating set.

How should we decide which nodes to select as dominators? Certainly, selecting all nodes would be overkill. Consider a node x whose neighbors y_1, y_2, $\ldots$, y_k have in-supports (in increasing order) $s_{in}(1) \leq s_{in}(2) \leq \ldots \leq s_{in}(k)$. Clearly, y_k needs node x as a dominator at most as much as y_{k-1} needs x because $s_{in}(k-1) \leq s_{in}(k)$. Similarly, y_{k-1} needs x at most as much as y_{k-2} needs x, and so on. Thus, to decide whether we want to select x as a dominator (as in issu 2 above), we use the inverse of the median of $s_{in}(i)$'s. More specifically, we select a candidate x with probability equal to the inverse of the median of the in-supports of the neighbors of x.

The complete algorithm—the weighted local randomized greedy (WLRG) algorithm—is described in Algorithm 2.

Explanatory notes on the algorithm: Let $D = C = \emptyset$. D will denote the set of nodes selected to be in the dominating set. C will denote the set of nodes already covered by the dominators. Also, the set of neighbors of x with which x shares a good communication link are determined using the received signal strength indicator (RSSI). RSSI is inversely proportional to the signal strength. This allows nodes to communicate only with other nodes to which there is a strong connection. Fewer retransmissions will be required to achieve a successful transmission over such links. While there is only a weak correlation between RSSI and node distance, the link quality does impact the amount of energy required for communications.

- An intuitive way to think about $s_{in}(y)$ is the following: Suppose all candidate nodes were made dominators. Suppose also that each dominator x selected $c(x)$ neighbors—the maximum number of nodes that x can dominate—at random from its neighbors. Then, $s_{in}(x)$ is the expected number of dominators that cover the uncovered node x.

Algorithm 2 Weighted Local Randomized Greedy Algorithm

```
Require: Graph $G = (V, E)$, constraint $r(v_i)$ on vertices
Ensure: Subset $D \subseteq V$, set of currently chosen vertices
  Span calculation: Compute the constrained span $c(x)$ by computing the minimum of
    the constraint and the number of uncovered neighbors of $x$.
  Candidate selection: Compute whether $\hat{c}(x)$ is at least as much as the constrained
    span of each node within a distance of 2 from $x$. If so, $x$ is a candidate.
  Constrained out-support calculation: If $x$ is a candidate, compute the constrained
    out-support of $x$ as follows: If $c(x) = 0$, let $s_{out}(x) = 0$. Else,
        $s_{out}(x) = \frac{c(x)}{||N(x) - C||}.$
    Note that $||N(x) - C||$ is the number of uncovered neighbors of $x$.
  Constrained in-support calculation: If $x$ is an uncovered node, let $A(x)$ be the set of
    neighbors of $x$ that are candidates. Compute the constrained in-support $s_{in}(x)$ of $x$ as
        $s_{in}(x) = \sum_{y \in A(x)} s_{out}(y).$
  Dominator selection: If $x$ is a candidate, find the median $m$ of $\{s_{in}(y) | y \in N(x) - C\}$.
    Let $p = 1/m$. With probability $p$, add $x$ to $D$.
  Neighbor selection: If $x$ is selected, add $x$ to $D$, and for each neighbor $y \in N(x) - C$,
    select $y$ with probability $s_{out}(x)$ and add it to $V_x$. Set $C = \bigcup_{x \in D} V_x$.
  Go to the next round.
```

- A candidate whose uncovered neighbors all have large s_{in}'s intuitively need not be selected as a dominator, because its neighbors will likely get covered by *other* nodes. On the other hand, if we only select very few dominators, then the algorithm will run for many rounds. This is the principle for selecting a dominator with probability equal to the median of the inverse of s_{in}'s.

We can show the algorithm described above (with a slight modification) returns a dominating set that obeys the constraints with high probability (whp). The number of rounds is $O(\log n \log \Delta)$ (Δ is the maximum constrained degree) whp.

VI. ANALYSIS OF THE DISTRIBUTED ALGORITHM WLRG

WLRG (Weighted Local Randomized Greedy) is described in Section V. We now show that WLRG terminates in $O(\log n \log \Delta)$ rounds with high probability.

Theorem 6.1: WLRG on a graph $G = (V, E)$ terminates in $O(\log n \log \Delta)$ where n is the number of nodes and Δ is $\max\{\min(t(v), d(v)) | v \in V\}$, where $t(v)$ is the constraint on v and $d(v)$ is the degree of v.

We will now give the proof of this result. The structure of this proof closely follows the analysis of LRG [18]. In fact, since ECDS is a generalization of the dominating set problem, WLRG is a generalization of LRG. The key difference between our analysis and the analysis of LRG is that (a) we need a notion of *partial coverage* and (b) we need to incorporate in our analysis the neighbor selection step, a step that is not present in the LRG algorithm.

Let $G = (V, E)$ be the sensor node graph. In the proof, we will focus on a round (say the ith round) of WLRG. Let C be the set of nodes covered in an earlier round. Let $H = (V', E')$ be the subgraph of G such that V' is the union of all candidate nodes X (as defined by the candidate selection step) and all uncovered nodes Y adjacent to some $x \in X$, and E' consists of edges $(u, v) \in E$ where u is a candidate and v is an uncovered node.

Lemma 6.2: (Equivalent to Lemma 3.1 of [18].) All candidates in a connected component of H must have the same constrained span.

Proof: Let v_1 and v_2 be two candidates in a connected component of H. Consider a path p from v_1 to v_2 in H. Then there cannot be two consecutive nodes in p such that both are non-candidates. (This is because at least one end-point of each edge in H is a candidate node.) Since any two candidates within a distance of 2 must have the same constrained span, we have that all candidates that lie on p must have the same constrained span. And it follows that all candidates in a connected component of H have the same constrained span.

□

We will now show using a potential function argument that WLRG terminates in $O(\log n \log \Delta)$ rounds with high probability. We define the potential at the start of a round as follows: Let m be the maximum constrained span of any node at the start of a round.

Define Φ as

$$\Phi = \sum_{v:\hat{c}(v)=m} c(v).$$

Lemma 6.3: (Equivalent to Lemma 3.2 of [18].) Let Φ_i and Φ'_i be the potentials at the beginning and end of round i. There is a $d > 0$ such that $E[\Phi'_i] \leq d\Phi_i$.

Note that the potential at the start of round $i+1$ might not be the same as the potential at the end of round i because the underlying graph changes due to some nodes being covered in round i.

Proof: Recall that X is the set of candidates. For each candidate v, let $U(v)$ denote the set of uncovered neighbors of v. Sort the elements of $U(v)$ in nonincreasing order of their in-supports $s_{in}()$'s. Let $T(v)$ (respectively, $B(v)$) denote the set of the first $\lceil ||U(v)||/2 \rceil$ (last $\lceil ||U(v)||/2 \rceil$) elements of $U(v)$. For a candidate v and a node $u \in U(v)$, we say that v is a *top dominator* for u if $u \in T(v)$. The probability that a top dominator v of u is selected is $1/m$, where m is the median of $\{s_{in}(y) | y \in U(v)\}$. Since $u \in T(v)$, $1/m \geq 1/s_{in}(u)$.

For an uncovered node u in H, we say that u is a *top heavy* node if at least $s_{in}(u)/4$ of its in-support comes from candidates that are top dominators for u. An uncovered node is *bottom heavy* if it is not top heavy. □

Lemma 6.4: If u is top heavy, then the probability that u is covered in this round by a top dominator of u is at least $1 - e^{-1/4}$.

Proof: Let $P_c(u)$ be the probability that u is covered in this round by a top dominator. Then, the probability that u is not covered in this round by a top dominator is $1 - P_c(u)$. Since u is not covered if none of the top dominators adjacent to u cover u, we can write this probability as:

$$\Pi_{v \in X : u \in T(v)} P[u \text{ is not covered by } v].$$

We will determine an upper bound this term.

Let $P_d(v)$ be the probability that v is picked to be a dominator in this round. If u is not covered by v, then exactly one of the following events happen:

- v is not picked to be a dominator (with probability $1 - P_d(v)$); or

- v is picked to be a dominator (with probability $P_d(v)$) and yet v does not cover u (with probability $1 - s_{out}(v)$).

Thus,

$$P[u \text{ is not covered by } v] = (1 - P_d(v)) + P_d(v)(1 - s_{out}(v)),$$

which simplifies to $1 - P_d(v)s_{out}(v)$. As shown above, if $u \in T(v)$, then $P_d(v) \geq 1/s_{in}(u)$. Thus,

$$\Pi_{v \in X : u \in T(v)}(1 - P_d(v)s_{out}(v)) \leq \Pi_{v \in X : u \in T(v)}(1 - \frac{s_{out}(v)}{s_{in}(u)}).$$

Define $x_v = \frac{s_{out}(v)}{s_{in}(u)}$.

Note that since u is top heavy, it follows from definition, that

$$\Sigma_{v \in X : u \in T(v)} s_{out}(v) \geq \frac{s_{in}(u)}{4}.$$

Thus,

$$\Sigma_{v \in X : u \in T(v)} x_v \geq \frac{1}{4}.$$

Let there be n elements in the set $\{v \in X | u \in T(v)\}$.

$$\Pi_{v \in X : u \in T(v)}(1 - x_v) \leq (1 - \frac{1}{4n})^n \leq e^{1/4}.$$

Since $1 - P_c(u) \leq e^{1/4}$, it follows that $P_c(u) \geq 1 - e^{1/4}$. □

Consider an arbitrary edge $(v, u) \in E'$. (Recall that E' is the set of edges (v, u) in H such that v is a candidate and u is an uncovered node.) This edge can be one of four types:

1. v is a top dominator for u and u is top heavy (call this set of edges E_{tt}),

2. v is a top dominator for u and u is bottom heavy (call this set of edges E_{tb}),

3. v is a bottom dominator for u and u is top heavy (call this set of edges E_{bt}), or

4. v is a bottom dominator for u and u is bottom heavy (call this set of edges E_{bb}).

Let $S_{tt} = \sum_{(v,u)\in E_{tt}} s_{out}(v)$. Similarly, define S_{tb}, S_{bt}, and S_{bb}. Let S be the sum over all edges (v,u) such that v is a candidate and u is an uncovered node in H, of $s_{out}(v)$.

Note that $E_{tt} \cap E_{bt}$ or $E_{bt} \cap E_{bb}$ might not be empty because a node v can be both a top and a bottom dominator for a node u. Certainly, though, $E_{tt} \cap E_{tb} = E_{bt} \cap E_{bb} = \emptyset$.

Lemma 6.5: (equivalent to Lemma 3.4 of [18].) Let S_{tt} and S be as defined above. Then,

$$S_{tt} \geq (1/3)S.$$

Proof: Consider a bottom heavy node u.

$$\sum_{v\in X:u\in B(v)} s_{out}(v) < \frac{s_{in}(u)}{4}.$$

Thus,

$$\sum_{v\in X:u\in B(v)} s_{out}(v) \geq \frac{3s_{in}(u)}{4}.$$

Thus,

$$\sum_{v\in X:u\in B(v)} s_{out}(v) > 3\sum_{v\in X:u\in T(v)} s_{out}(v).$$

If we sum both sides of the above inequality over all bottom heavy nodes, we have that $S_{bb} \geq 3S_{tb}$. We also know that $S_{bb} \leq (1/2)S$. Thus, $S_{tb} \leq (1/6)S$. Now,

$$S_{tt} + S_{tb} \geq (1/2)S.$$

Thus, $S_{tt} \geq (1/2 - 1/6)S = (1/3)S$. □

We can now use these results to prove Lemma 6.3 and also Theorem 6.1 in exactly the same manner as [18]. The only difference between the two proofs is that in our proof Δ is $\max\{\min(t(v), d(v)) | v \in V\}$, a global upper bound on the constrained span for any node in any round, while in [18] Δ is the maximum degree of the graph, also a global upper bound on the span of a node in the graph.

VII. EXPERIMENTS

In order to test the distributed clustering algorithm, we implemented the algorithm in TinyOS and ran simulations in TOSSIM [24]. We compared the ECDS algorithm against the HEED algorithm [7] and used a random topology for each simulation. We ran a simulated 15 minutes for network sizes of 30, 45, 60, and 75 nodes. Our algorithm is independent of the routing protocol used, but for our experiments we use the Surge multi-hop application that is part of TinyOS. HEED also uses the Surge multi-hop routing protocol. Each node generates a reading every 20 seconds. The cluster heads aggregate the readings. Surge uses a link estimation and parent selection (LEPS) mechanism to determine multi-hop routes. All traffic received at each node is monitored and used to update the internal neighbor table. The neighbor table tracks all neighbors and selects the next hop based on shortest path semantics. The default destination is the base station. We use a credit-point system for updating the mote energy budget as used with iHEED [25]. ECDS and HEED use energy for tasks such as sending and receiving and points are deducted propotional to the actual amount of energy used. Each node starts with the same amount of points and for each send/receive an amount proportional to the size of the message is deducted. In our implementation, a cluster head receives many more messages from nodes in its cluster than it sends. All messages are sent with the same power level, therefore we do not consider the distance when determining the cost of each send/receive [26]. For ECDS, the initial energy allows a constraint of 20. Whenever the network re-clusters, the constraint is updated and is based on the energy available at each node. For each network size, the experiments were repeated 30 times. We measured the size of the dominating set and compared it to the expected size of the dominating set for each round, which allowed us to show that the algorithm performs as expected. We measured the number of rounds the algorithm executed until the entire network was clustered. We compared the time of the first node's death to the last node's death. Having all nodes die at approximately the same time provides the most useful WSN. Additionally, we measured the time it took for the entire network to cluster. A fast clustering algorithm ensures a useful WSN.

A. Cluster Generation

In a distributed environment it is important to evaluate how long it takes for a clustering protocol to finish. There are two measurements for WSN: time and the number of rounds of execution. Figure 2(a) shows the average number of rounds to cluster the network for various sizes. An ideal distributed clustering algorithm will cluster in a constant number of rounds. Both the ECDS and the HEED algorithm execute in a constant number of rounds, but the ECDS algorithm finished in fewer rounds. The algorithm depends on the routing information obtained from the (independent) routing protocol. This routing information may not be complete, especially in the earlier rounds. Incomplete routing information will exclude some nodes from joining a cluster at each round. Similar behavior can be seen in Figure 2(b), which shows the average time it took for the networks to cluster. Clearly, the number of rounds and the time are related and both are important measurements. An algorithm that runs over several short rounds may still outperform an algorithm that runs in a constant number of long rounds. Again, it is important that an algorithm takes a constant amount of time, no matter the size of the network. Both the ECDS and the HEED algorithm take a constant amount of time, but the ECDS algorithm is faster.

B. Cluster Goodness

Our algorithm uses a randomized, probabilistic approach. At each round, the sum of the probabilities is equal to the number of expected cluster heads. Figure 2(c) shows the average expected number of cluster heads versus the average actual number of cluster heads for each network size. For all networks, the average number of expected cluster heads is close to the average actual number of cluster heads, indicating that our algorithm performs as expected. Figure 3(a) shows the average size of the dominating set. The dominating set is the number of cluster heads selected for each simulation run. Each node starts with the same amount of energy, an amount that can support up to 20 nodes in a cluster. Another important consideration is the number of nodes assigned to each cluster. Scalability is improved when clusters are of similar size regardless of network size. Figure 3(b) shows the average number of nodes in each cluster. In ECDS the number of nodes assigned to a cluster remains relatively constant, while the size decreases asymptotically for HEED. Not only the number of nodes in each cluster and the number of clusters matter, but also

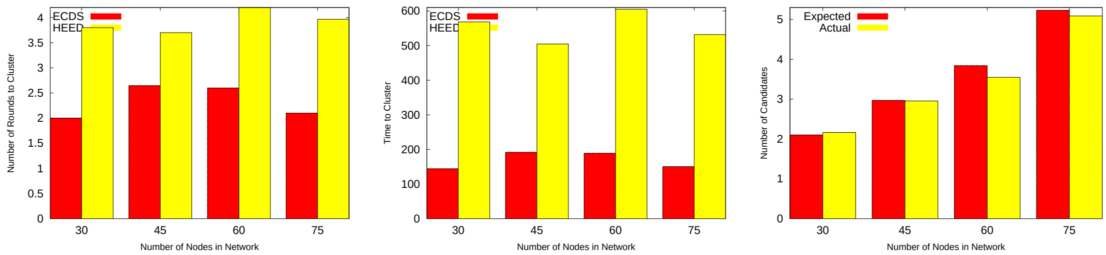


Fig. 2. (a) Rounds to Cluster (b) Time to Cluster (c) Expected vs. Actual Number of Cluster Heads

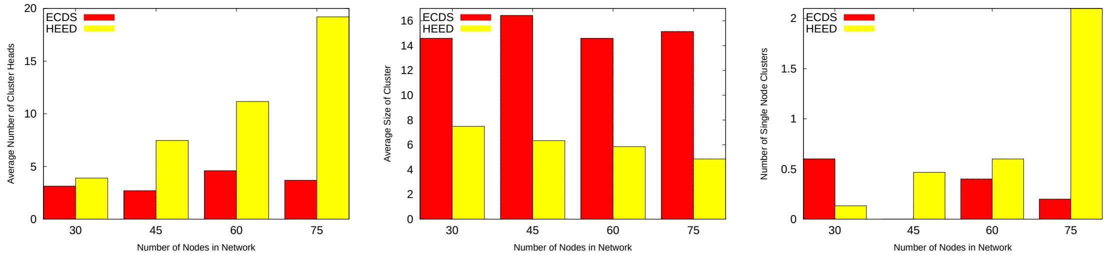


Fig. 3. (a) Dominating Set Size (b) Cluster Size (c) Average Number of Single Node Clusters

how many of those clusters are single-node clusters (clusters in which the cluster head is the only node). A single node cluster does not improve performance, but it is generally unavoidable. A good algorithm will minimize the number of such clusters. Figure 3(c) shows the average number of single node clusters for ECDS and HEED. For ECDS the number of single node clusters decreases as the size of the network grows. ECDS choses only neighbors which are “near” as cluster heads, some nodes will not be near a cluster head and thus create single node clusters. As the network grows, each node has more opportunities to find a near cluster head, hence the decrease. On the other hand, HEED’s single node clusters increase in number as the network grows.

C. Lifetime of Sensor Nodes

In a wireless sensor network, the early death of some nodes can disconnect other nodes from the base station. This situation can lead to a reduced usefulness of the network because some data cannot reach the base station. We measure lifetime in two ways: (1)

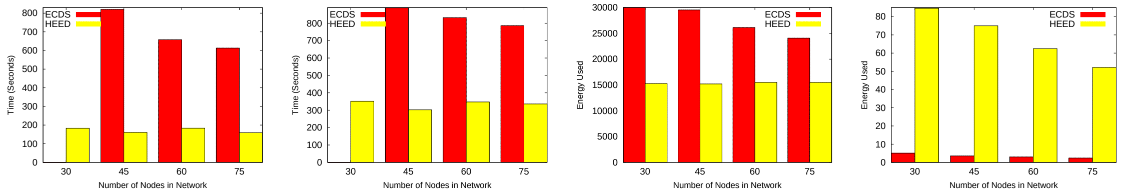


Fig. 4. (a) First Death (b) Last Death (c) Average Energy (d) Average Energy Per Message

the time at which the first node dies and (2) the time at which the last node dies. The time at which the first node dies is important because it can lead to a disconnection of part of the network. The time at which the last node dies shows how long nodes are able to run the protocol. Figure 4(a) shows the time at which the first node died for the ECDS and the HEED algorithm. The time of the first death asymptotically decreases in ECDS and is constant for HEED. Figure 4(b) shows the time of death for the last node in the network. It is equally important that all nodes die around the same time. A single node that outlives others by a large margin is of little use. It can be estimated that the lifetimes will be similar for ECDS and HEED in large networks. For both ECDS and HEED the first and last deaths are within 200 seconds of each other, indicating an even energy consumption across the network.

D. Energy Consumption

The amount of energy used during the execution of a protocol is very important in sensor networks. Figure 4(c) shows the average energy consumption for the two protocols. The energy consumption of the HEED algorithm is linear, while the energy consumption of the ECDS algorithm is asymptotically decreasing. As the networks grow larger, the energy consumption for ECDS and HEED will be similar. In sensor networks, the energy consumption for each message sent should be considered in addition to the overall energy consumption. A sensor network that uses very little energy is not useful if it does not produce an adequate amount of data. Figure 4(d) shows the average energy consumption for each message sent. Since ECDS clusters faster, it generates more messages.

VIII. CONCLUSION AND OPEN PROBLEMS

In this paper, two different algorithms are presented to address the problem of energy constrained clustering for wireless sensor networks. For the greedy algorithm we provide an $O(\log n)$ approximation guarantee. The second algorithm presented is a distributed algorithm for the energy constrained dominating set. We proved that this algorithm runs in $O(\log n \log \Delta)$ rounds whp. This algorithm performs well on the random graphs in our simulations. Our simulations showed that our algorithm performs very well in terms of time to cluster, cluster size, and energy consumption. We compared our algorithm with the HEED algorithm. It outperformed HEED in terms of cluster size, time to cluster, and energy consumption per message sent. Future work will include extending the algorithm to consider node proximity when selecting cluster heads and deciding which nodes to add to the cluster. Considering node proximity will produce tighter clusters and minimize the overall energy consumption within each cluster. Secondly, we plan on extending the algorithm to allow for multi-hop clusters. Currently every node is one hop from its cluster head. We will extend the algorithm to allow nodes to be k-hops from their cluster heads. Additionally, we plan on extending the algorithm to allow each node to have multiple cluster heads which will ensure that each node has access to at least one cluster head at all times. Ensuring multiple coverings for each node will allow for the use of multi-path routing in clustered networks.

REFERENCES

[1] W. R. Heinzelman, J. Kulik, and H. Balakrishnan, “Adaptive protocols for information dissemination in wireless sensor networks,” in *MobiCom ’99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*. ACM Press, 1999, pp. 174–185.

[2] S. Lee, C. Kim, and S. Kim, “Constructing energy efficient wireless sensor networks by variable transmission energy level control,” in *Computer and Information Technology, 2006. CIT ’06. The Sixth IEEE International Conference on*, 2006, pp. 225–225.

[3] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-efficient communication protocol for wireless microsensor networks,” in *HICSS ’00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8*. Washington, DC, USA: IEEE Computer Society, 2000, p. 8020.

[4] D. Wei and A. H. Chan, “Clustering algorithm to balance and to reduce power consumptions for homogeneous sensor networks,” in *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*, 2007, pp. 2723–2726.

[5] R. Krishnan and D. Starobinski, “Efficient clustering algorithms for self-organizing wireless sensor networks,” *Ad Hoc Networks*, vol. 4, no. 1, pp. 36–59, January 2006.

[6] C. Duan and H. Fan, “A distributed energy balance clustering protocol for heterogeneous wireless sensor networks,” in *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*, 2007, pp. 2469–2473.

[7] O. Younis and S. Fahmy, “HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks,” *Mobile Computing, IEEE Transactions on*, vol. 3, no. 4, pp. 366–379, 2004.

[8] F. Kuhn and R. Wattenhofer, “Constant-time distributed dominating set approximation,” in *In Proc. of the 22nd ACM Symposium on the Principles of Distributed Computing*, 2003.

[9] R. Karp, “Reducibility among combinatorial problems,” in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds. Plenum Press, 1972, pp. 85–103.

[10] F. Kuhn, T. Moscibroda, and R. Wattenhofer, “Unit disk graph approximation,” in *DIALM-POMC ’04: Proceedings of the 2004 joint workshop on Foundations of mobile computing*. ACM Press, 2004, pp. 17–23.

[11] J. Wu and H. Li, “Domination and its applications in ad hoc wireless networks with unidirectional links,” in *ICPP ’00: Proceedings of the Proceedings of the 2000 International Conference on Parallel Processing*. Washington, DC, USA: IEEE Computer Society, 2000, p. 189.

[12] D. Dubhashi, A. Mei, A. Panconesi, J. Radhakrishnan, and A. Srinivasan, “Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons,” in *SODA ’03: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2003, pp. 717–724.

[13] V. Mhatre and C. Rosenberg, “Design guidelines for wireless sensor networks: communication, clustering and aggregation,” *Ad Hoc Networks*, vol. 2, no. 1, pp. 45–63, 2004.

[14] M. Younis, M. Youssef, and K. Arisha, “Energy-aware routing in cluster-based sensor networks,” in *MASCOTS ’02: Proceedings of the 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS’02)*. Washington, DC, USA: IEEE Computer Society, 2002, p. 129.

[15] S. Banerjee and S. Khuller, “A clustering scheme for hierarchical control in multi-hop wireless networks,” in *IEEE INFOCOM*, April 2001, pp. 1028–1037.

[16] M. Chen and J. Wu, “EECS: an energy efficient clustering scheme in wireless sensor networks,” *Performance, Computing, and Communications Conference, 2005. IPCCC 2005. 24th IEEE International*, pp. 535–540, 2005.

[17] F. Grandoni, J. Knemann, A. Panconesi, and M. Sozio, “Primal-dual based distributed algorithms for vertex cover with semi-hard capacities,” in *PODC ’05: Proceedings of the twenty-fourth annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*. New York, NY, USA: ACM Press, 2005, pp. 118–125.

[18] L. Jia, R. Rajaraman, and T. Suel, “An efficient distributed algorithm for constructing small dominating sets,” *Distrib. Comput.*, vol. 15, no. 4, pp. 193–205, 2002.

[19] S. Kutten and D. Peleg, “Fast distributed construction of k-dominating sets and applications,” in *PODC ’95: Proceedings of the fourteenth annual ACM symposium on Principles of distributed computing*. ACM Press, 1995, pp. 238–251.

[20] Y. Chen and A. Liestman, “Approximating minimum size weakly-connected dominating sets for clustering 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 Press, 2002, pp. 165–172.

[21] S. Guha and S. Khuller, “Approximation algorithms for connected dominating sets,” in *ESA ’96: Proceedings of the Fourth Annual European Symposium on Algorithms*. London, UK: Springer-Verlag, 1996, pp. 179–193.

[22] T. T. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to algorithms*. Cambridge, MA, USA: MIT Press, 1990.

[23] B. Liang and Z. Haas, “Virtual backbone generation and maintenance in ad hoc network mobility management,” in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, 2000, pp. 1293–1302 vol.3.

[24] P. Levis, N. Lee, M. Welsh, and D. Culler, “Tossim: accurate and scalable simulation of entire tinyos applications,” in *SenSys ’03: Proceedings of the 1st international conference on Embedded networked sensor systems*. New York, NY, USA: ACM Press, 2003, pp. 126–137.

[25] O. Younis and S. Fahmy, “An experimental study of routing and data aggregation in sensor networks,” *Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on*, pp. 8 pp.–, Nov. 2005.

[26] A. Boukerche, X. Fei, and R. B. Araujo, “An optimal coverage-preserving scheme for wireless sensor networks based on local information exchange,” *Computer Communications*, vol. 30, no. 14-15, pp. 2708–2720, October 2007.

II. ENERGY CONSTRAINED CLUSTERING ALGORITHMS FOR WIRELESS SENSOR NETWORKS

Julia Albath, *Student Member, IEEE*, Mayur Thakur, *Member, IEEE*, and Sanjay Madria, *Senior Member, IEEE*

Abstract— Using partitioning in sensor networks to create clusters for routing, data management, and other protocols has been proven as a way to ensure scalability and to deal with sensor network shortcomings such as limited communication ranges and energy. Choosing a cluster head within each cluster is important because cluster heads use additional energy for their responsibilities and that burden needs to be carefully passed around. Many existing protocols either choose cluster heads randomly or use nodes with the highest remaining energy. We introduce the energy constrained minimum dominating set (ECDS) to model the problem of optimally choosing cluster heads with energy constraints. We show its applicability to sensor networks and give an approximation algorithm for solving the ECDS problem. We propose a distributed algorithm for the constrained dominating set. We experimentally show that the distributed algorithm performs well in terms of energy usage, node lifetime, and clustering time and, thus, is very suitable for wireless sensor networks.

1 INTRODUCTION

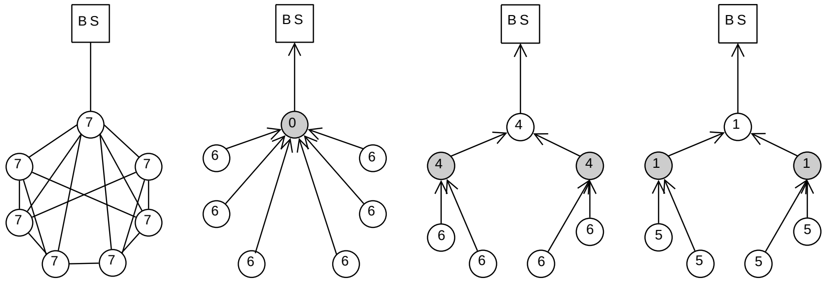
As sensor networks mature and are used in many situations, the necessity arises for new and improved protocol and algorithms. Each sensor has a limited amount of energy available. Combined with other limited resources such as processing power, radio bandwidth and memory, new protocols and algorithms need to be developed. Routing protocols need to work within this limited environment while achieving their goals.

The limited resources are important and need to be considered because they affect the lifetime of the network. In most applications of wireless sensor networks, it is impossible to replace batteries in order to extend the lifetime of the network. Adding additional sensors may be a possibility in some situations, but not in cases such as battlefield deployment. This illustrates the need for protocols which extend the lifetime of the network and hence extend the usefulness of the network.

Routing protocols and algorithms which organize the network into clusters have been shown to greatly improve network lifetime [1]. Clustering has been shown to greatly reduce power consumption, is easily scalable, and is robust in face of node failures [2]. A good clustering scheme takes into account one or more of the following: communication range, number and type of sensors, geographical location, and remaining energy [3]. Clustering protocols need to make two important decisions, one is cluster head selection and the other is which nodes to assign to which cluster head.

Cluster head selection is important because cluster heads spend more energy aggregating, forwarding messages, doing general routing maintenance and similar actions. A small set of cluster heads may not be optimal in terms of network lifetime. A cluster head uses additional energy and could be depleted much sooner then other nodes. More cluster heads may mean that each cluster head has less work to do and each cluster head may survive a longer amount of time. Consider the graph shown in Figure 1. Each node starts with the same amount of energy (7 units) (Figure 1(a)) and one unit is used for each receive or send. The cluster head aggregates the received messages into one outgoing message. The optimal dominating set is one node (Figure 1(b)), but the network becomes disconnected after only one time step. On the other hand a slightly non-optimal dominating set using the heuristic “Don’t give a cluster head more than three nodes” results in a network that survives two time steps as shown in Figures 1(c) and 1(d) (the shaded nodes represent the cluster heads).

Cluster head selection in wireless sensor networks and in Mobile Ad-Hoc Networks (MANET) benefits from using a dominating set approach. Dominating set clustering can be executed in a constant number of rounds which leads to better clustering [4]. In a dominating set approach each node is either a cluster head or one hop from a cluster head [5]. This can be extended to allow each node to be at most k-hops from its cluster head. Allowing for k-hops within a cluster improves scalability in very large networks. Very large networks, even when clustered will exhibit problems similar to unclustered, smaller networks [6]. The Energy Constrained Dominating Set (ECDS) was introduced. Each node in a wireless sensor network is constrained by its limited resources, especially energy.



1(a) Original Network

1(b) Without Constraints After One Round

1(c) With Constraints After One Round

1(d) With Constraints After Two Rounds

Fig. 1. Energy Constrained Clustering Example

Other constraints include, but are not limited to, bandwidth, storage and computational abilities. ECDS models the problem of optimally choosing cluster heads with energy constraints. The distributed ECDS algorithm runs in $O(\log n \log \Delta)$ rounds whp (with high probability). Experimental evaluations showed the distributed algorithm to be very well suited for wireless sensor networks. Motivated by the above examples, we improve the Weighted Local Randomized Greedy Algorithm. The contributions of this paper are summarized as follows:

- We introduce an improved ECDS (iECDS) algorithm. iECDS improves the performance of ECDS by considering the connection quality to neighbors when making decisions. This reduces the power required to successfully send messages and it extends the lifetime of the network. We improved ECDS by modifying the candidate selection. In iECDS candidate selection is based on the rounded constrained span unlike ECDS which uses the constrained span. The rounded constrained span is the smallest power of 2 that is at least as much as the constrained span. This increases the number of candidates elected at each round and leads to faster network clustering. The iECDS algorithm is described in Section 4.2.

- We introduce a k-hop cluster ECDS (kECDS) algorithm. In kECDS a node can be up to k hops from its cluster head. kECDS, like iECDS and ECDS, uses multi-hop from the cluster heads to the base station. Allowing for k-hops within a cluster leads to larger clusters and fewer small clusters. Section 4.3 provides the details of the kECDS algorithm.

- We provide extensive simulations using TOSSIM in Section 6. We compared the performance of the iECDS and kECDS algorithms to ECDS and HEED[7]. HEED selects cluster heads according to residual energy and node proximity to neighbors or node degree. ECDS uses local information about the connectivity of each node and the connectivity of its neighbors in addition to the residual energy to decide which node should become a cluster head.

2 SYSTEM MODEL

In this paper we make assumptions regarding the environment in which the network will operate. The network consists of a number of nodes randomly placed in the area to be monitored. Each sensor has the same abilities and the network includes one base station, such as a laptop, which has additional processing powers. Other assumptions are detailed here.

- The sensor nodes are stationary, that is, nodes cannot move from one location to another on their own accord. Nodes can be moved by external forces, such as animals, attackers or strong winds. Any node movement will be taken into account whenever the network reclusters. After moving, a node may no longer be in communication range of its cluster head. From the time when a node moves until the network is reclustered, the node will not be able to contribute data. When the network is reclustered, the node choses a new cluster head and contributes data.

- Communication links are symmetric. If node u can hear node v, then node v can hear node u. The algorithms use the local neighborhood connectivity and layout in order to elect cluster heads, so it is important that all nodes in the neighborhood are aware of the connections. A node needs to know that its neighbors can hear it. If node u is a candidate cluster head and knows node v is in its neighborhood, it is important that v can hear u.

- Nodes are not location aware. The nodes are not equipped with GPS receivers and do not know their location within the covered area. Nodes do not know in which direction other nodes are located, nor do they know what other nodes can communicate with each other. Nodes do not know the physical distance to the base station. Nodes do know how many hops they are from the base station.

- Each sensor knows its local neighborhood. Nodes communicate with each other using wireless communication to construct a connected network. Each node knows which nodes it is able to communicate with and it knows the quality of this connection. Messages are exchanged using some underlying multi-hop routing protocol. The algorithms are independent of the routing protocol used.

- The nodes and the network will be left unattended after deployment. Some messages may be sent from the base station to the network. These messages will not contain global network information. Algorithms need to be able to take the local information and make global decisions.

- Some nodes are able to communicate with the base station, but the majority are not in direct communication range of the base station. Those nodes will use the underlying multi-hop routing protocol to send their messages to the base station. The ECDS algorithms are independent of the routing protocol. Any multi-hop routing protocol can be used with the ECDS algorithms.

- The nodes are loosely time-synchronized. Time-synchronization is necessary in order to make sure that all nodes participate in the ECDS algorithms within the same interval. This time-synchronization can be handled by existing hardware on most nodes. The ECDS algorithms do not require special time-synchronization algorithms.

3 PROBLEM STATEMENT

In a typical wireless sensor network, n nodes are distributed randomly over the coverage area. Our goal is to ensure that the area is properly covered to ensure even sensing. In this work, coverage also refers to one node's ability to handle the relaying of messages to and from other nodes in its cluster. We want to organize the network into clusters such that each cluster has one cluster head. All other nodes send their messages to the cluster head. The cluster head will aggregate the data received from the cluster and send the aggregate to the base station.

A sensor network can be expressed as a graph $G = (V, E)$, where each of the vertices represents a sensor node and there is an edge between two vertices if their corresponding sensor nodes are within each other's communication range. A dominating set of a graph $G = (V, E)$ is a subset $V' \subseteq V$ such that each $x \in V - V'$ has a neighbor in V'. The assignment of nodes to cluster heads is often modeled as a dominating set (DS) problem [8]. The minimum dominating set problem is NP-complete for general graphs[9] and remains NP-complete for planar graphs, unit disk graphs, bi-partite graphs, and chordal graphs, but it does admit a Polynomial Time Approximation Scheme (PTAS) for planar graphs and unit disk graphs [10], [11]. The dominating set problem models the optimization problem of finding a small number of cluster heads.

We will now describe the notations using in the rest of the paper and define dominating set and energy constrained dominating set.

Definition 3.1: The dominating set for a graph G and a subset S of the vertex set $V(G)$, denote by $N_G[S]$ the set of vertices in G which are in S or adjacent to a vertex in S. If $N_G[S] = V(G)$, then S is said to be a *dominating set of* G.

Definition 3.2: Given an undirected graph $G = (V, E)$, and, for each $v_i \in V(G)$, a constraint $r(v_i) \in N$, the energy-constrained dominating set (ECDS) of G is a pair (S, C), where C is an assignment from $x \in S$ to $V_x \subseteq V$ such that (a) $V_x | x \in S$ is a partition of V, (b) for each $x \in S, x \in V_x \subseteq N_G[x]$, G so that the following condition is met, and (c) for each $x \in S, ||V_x|| \leq r(x) + 1$.

In the definition of ECDS we assume that when a node is selected as cluster head, it includes itself in the cluster. (See part (b) of the definition). Also note from the "+1" in condition (c) that we allow a node to cover itself for free. That is, the constraint $r(x)$ for x denotes the maximum number of nodes that x can cover in addition to itself.

ECDS is also related to, but different from, the Network Clustering problem [12]. ECDS has a constraint parameter that is not present in Network Clustering. Also, the clusters must form a partition in ECDS, whereas they may overlap in the Network Clustering problem. The general dominating set can be described as a constrained dominating set where each constraint is equal to n, the number of nodes in the graph. It trivially follows, that the constrained minimum dominating set is NP-complete. The minimum dominating set in the general form has approximation algorithms of $1 + \log ||V||$. Since the constrained dominating set problem is a special case of the general dominating set problem, no improvement on these bounds will be possible.

4 ENERGY CONSTRAINED CLUSTERING PROTOCOLS

First we present the original ECDS protocol followed by the improved ECDS (iECDS) and kECDS, which is a version of ECDS that allows for k-hop communication within the cluster.

4.1. Energy Constrained Dominating Set

In this section we describe the ECDS protocol. First, we give the algorithm and discuss the parameters used in the clustering process. Second, we will show the theoretical performance bounds of the ECDS algorithms. This enables us to compare the iECDS and kECDS algorithms to the original ECDS algorithm.

4.1.1. ECDS Algorithm

We will explain and define several terms used in order to facilitate the understanding of the algorithm. The *constrained span* $c(x)$ of a node x in the ECDS algorithm is defined as the smaller of: the number of uncovered neighbors of x and the constraint of x. The neighbors of x are defined as x and all nodes with which x shares a communication link of high quality. We use the received signal strength indicator (RSSI) to determine link quality. RSSI is inversely proportional to the signal strength. This allows us to communicate only with nodes to which we have a strong connection and therefore require fewer retransmissions to achieve our goal. This improves the cluster's ability to achieve its goal. While there is a weak correlation between RSSI and node distance, the link quality does impact the amount of power required for communications. Adjusting a nodes' neighborhood based on link quality allows us to use the lowest possible power setting for transmissions in order to communicate with the nodes in the cluster. This leads to additional energy savings. Let x be a candidate cluster head and y be a neighbor of x. Then the *out-*

support $s_{out}(x)$ of x is the ratio of the constrained span $c(x)$ to the number of uncovered neighbors of x. Assume that node x is a candidate and has 6 neighbors therefore $c(x)$ = 4 and $s_{out}(x) = 4/6$. In other words, the out-support of a candidate is the fractional support a node can offer to each neighbor. The *in-support* $s_{in}(y)$ of y is the sum of the out-support of each neighbor of y. Thus, the in-support of a node is the total support a node will get if all its neighboring dominators gives fractional support to each of its neighbors. The larger a node's in-support, the larger the probability that it will be covered by at least one dominator. We can say that $s_{in}(x)$ is the expected number of dominators that can cover x. Intuitively, we do not need to make dominators who have uncovered neighbors that all have large s_{in}'s. Those neighbors have a high probability of being covered by *other* nodes. This is the intuition for selecting a dominator with probability equal to the median of the inverse of s_{in}'s.

This raises the question of how should we decide which nodes to select as dominators? On one hand, selecting all nodes would be overkill. On the other hand, if we only select very few dominators, then the algorithm will run for many rounds. Let node x whose neighbors y_1, y_2, $\ldots$, y_k have in-supports (in increasing order) $s_{in}(1) \leq s_{in}(2) \leq \ldots \leq s_{in}(k)$. Clearly, y_k needs node x as a dominator at most as much as y_{k-1} needs x because $s_{in}(k-1) \leq s_{in}(k)$. Similarly, y_{k-1} needs x at most as much as y_{k-2} needs x, and so on. Thus, to decide whether we want to select x as a dominator, we use the inverse of the median of $s_{in}(i)$'s.

The completed weighted local randomized greedy (WLRG) algorithm is described in Algorithm 1. Each node x executes the algorithm in each round until the node and its neighbors are covered. Initially let $D = C = \emptyset$. D will denote the set of nodes selected to be in the dominating set. C will denote the set of nodes already covered by the dominators. Additionally, $N(x)$ denotes the set of neighbors of x, this includes x. Also note that by definition of $c(y)$, $c(y) \leq ||N(y) - C||$. Thus, if $||N(y) - C|| = 0$, then $c(y) = 0$, and so $s_{out}(y) = 0$.

4.1.2. Complexity Analysis

We now show that WLRG terminates in $O(\log n \log \Delta)$ rounds with high probability.

Algorithm 1 Weighted Local Randomized Greedy Algorithm

```
Require: Graph $G = (V, E)$, constraint $r(v_i)$ on vertices
Ensure: Subset $D \subseteq V$, set of currently chosen vertices
    Span calculation: Compute the constrained span $c(x)$ by computing the minimum of
        the constraint and the number of uncovered neighbors of $x$.
    Candidate selection: Compute whether $c(x)$ is at least as much as the constrained
        span of each node within a distance of 2 from $x$. If so, $x$ is a candidate.
    Constrained out-support calculation: If $x$ is a candidate, compute the constrained
        out-support of $x$ as follows: If $c(x) = 0$, let $s_{out}(x) = 0$. Else,
            $s_{out}(x) = \frac{c(x)}{||N(x) - C||}.$
        Note that $||N(x) - C||$ is the number of uncovered neighbors of $x$.
    Constrained in-support calculation: If $x$ is an uncovered node, let $A(x)$ be the set of
        neighbors of $x$ that are candidates. Compute the constrained in-support $s_{in}(x)$ of $x$ as
            $s_{in}(x) = \sum_{y \in A(x)} s_{out}(y).$
    Dominator selection: If $x$ is a candidate, find the median $m$ of $\{s_{in}(y) \mid y \in N(x) - C\}$. Let $p = 1/m$. With probability $p$, add $x$ to $D$.
    Neighbor selection: If $x$ is selected, add $x$ to $D$, and for each neighbor $y \in N(x) - C$,
        select $y$ with probability $s_{out}(x)$ and add it to $V_x$. Set $C = \bigcup_{x \in D} V_x$.
    Go to the next round.
```

Theorem 4.1: WLRG on a graph $G = (V, E)$ terminates in $O(\log n \log \Delta)$ where n is the number of nodes and Δ is $\max\{\min(t(v), d(v)) \mid v \in V\}$, where $t(v)$ is the constraint on v and $d(v)$ is the degree of v.

We will now give the proof of this result. The structure of this proof closely follows the analysis of LRG [13]. Let $G = (V, E)$ be the sensor node graph. In the proof, we will focus on a round (say the ith round) of WLRG. Let C be the set of nodes covered in an earlier round. Let $H = (V', E')$ be the subgraph of G such that V' is the union of all candidate nodes X (as defined by the candidate selection step) and all uncovered nodes Y adjacent to some $x \in X$, and E' consists of edges $(u, v) \in E$ where u is a candidate and v is an uncovered node.

Lemma 4.2: (Equivalent to Lemma 3.1 of [13].) All candidates in a connected component of H have the same span.

Proof: Let v_1 and v_2 be two candidates in a connected component of H. Consider a path p from v_1 to v_2 in H. Then there cannot be two consecutive nodes in p such that both are non-candidates. (This is because at least one end-point of each edge in H is a candidate node.) Since any two candidates within a distance of 2 must have the same span, we have that all candidates that lie on p must have the same span. And it follows that all candidates in a connected component of H must have the same span. □

We will now show using a potential function argument that WLRG terminates in $O(\log n \log \Delta)$ rounds with high probability. We define the potential at the start of a round as follows. Let o be the maximum span of any node at the start of a round. Define Φ as

$$\Phi = \sum_{v:c(v)=o} c(v).$$

Lemma 4.3: (Equivalent to Lemma 3.2 of [13].) Let Φ_i and Φ'_i be the potentials at the beginning and end of round i. There is a $d > 0$ such that $E[\Phi'_i] \leq d\Phi_i$.

Note that the potential at the start of round $i+1$ might not be the same as the potential at the end of round i because the underlying graph changes due to some nodes being covered in round i.

Proof: Recall that X is the set of candidates. For each candidate v, let $U(v)$ denote the set of uncovered neighbors of v. Sort the elements of $U(v)$ in non-increasing order of their in-supports $s_{in}()$'s. Let $T(v)$ (respectively, $B(v)$) denote the set of the first $\lceil ||U(v)||/2 \rceil$ (last $\lceil ||U(v)||/2 \rceil$) elements of $U(v)$. For a candidate v and a node $u \in U(v)$, we say that v is a *top dominator* for u if $u \in T(v)$. The probability that a top dominator v of u is selected is $1/m$, where m is the median of $\{s_{in}(y) \mid y \in U(v)\}$. Since $u \in T(v)$, $1/m \geq 1/s_{in}(u)$.

For an uncovered node u in H, we say that u is a *top heavy* node if at least $s_{in}(u)/4$ of its in-support comes from candidates that are top dominators for u. An uncovered node is *bottom heavy* if it is not top heavy. □

Lemma 4.4: If u is top heavy, then the probability that u is covered in this round by a top dominator of u is at least $1 - e^{-1/4}$.

Proof: Let $P_c(u)$ be the probability that u is covered in this round by a top dominator. Then, the probability that u is not covered in this round by a top dominator is $1 - P_c(u)$.

Since u is not covered if none of the top dominators adjacent to u cover u, we can write this probability as:

$$\Pi_{v \in X : u \in T(v)} P[u \text{ is not covered by } v].$$

We will determine an upper bound on this item.

Let $P_d(v)$ be the probability that v is picked to be a dominator in this round. If u is not covered by v, then exactly one of the following events happen:

- v is not picked to be a dominator (with probability $1 - P_d(v)$ or
- v is picked to be a dominator (with probability $P_d(v)$) and yet v does not cover u (with probability $1 - s_{out}(v)$)

Thus,

$$P[u \text{ is not covered by } v] = (1 - P_d(v)) + P_d(v)(1 - s_{out}(v)),$$

which simplifies to $1 - P_d(v)s_{out}(v)$. As shown above, if $u \in T(v)$, then $P_d(v) \geq 1/s_{in}(u)$. Thus,

$$\Pi_{v \in X : u \in T(v)}(1 - P_d(v)s_{out}(v)) \leq \Pi_{v \in X : u \in T(v)}(1 - \frac{s_{out}(v)}{s_{in}(u)}).$$

Define $x_v = \frac{s_{out}(v)}{s_{in}(u)}$.

Note that since u is top heavy, it follows from definition, that

$$\Sigma_{v \in X : u \in T(v)} s_{out}(v) \geq \frac{s_{in}(u)}{4}.$$

Thus,

$$\Sigma_{v \in X : u \in T(v)} x_v \geq \frac{1}{4}.$$

Let there be n elements in the set $\{v \in X \mid u \in T(v)\}$.

$$\Pi_{v \in X : u \in T(v)}(1 - x_v) \leq (1 - \frac{1}{4n})^n \leq e^{1/4}$$

Since $1 - P_c(u) \leq e^{1/4}$, it follows that $P_c(u) \geq 1 - e^{1/4}$. □

Consider an arbitrary edge $(v, u) \in E'$. (Recall that E' is the set of edges (v, u) in H such that v is a candidate and u is an uncovered node.) This edge can be one of four types:

1. v is a top dominator for u and u is top heavy (call this set of edges E_{tt}),
2. v is a top dominator for u and u is bottom heavy (call this set of edges E_{tb}),
3. v is a bottom dominator for u and u is top heavy (call this set of edges E_{bt}), or
4. v is a bottom dominator for u and u is bottom heavy (call this set of edges E_{bb}).

Let $S_{tt} = \sum_{(v,u) \in E_{tt}} s_{out}(v)$. Similarly, define S_{tb}, S_{bt}, and S_{bb}. Let S be the sum, over all edges (v, u) such that v is a candidate and u is an uncovered node in H, of $s_{out}(v)$.

Note that $E_{tt} \cap E_{bt}$ or $E_{bt} \cap E_{bb}$ might not be empty because a node v can be both a top and a bottom dominator for a node u. Certainly, though, $E_{tt} \cap E_{tb} = E_{bt} \cap E_{bb} = \emptyset$.

Lemma 4.5: (equivalent to Lemma 3.4 of [13].) Let S_{tt} and S be as defined above. Then,

$$S_{tt} \geq (1/3)S.$$

Proof: Consider a bottom heavy node u.

$$\sum_{v \in X : u \in B(v)} s_{out}(v) < \frac{s_{in}(u)}{4}.$$

Thus,

$$\sum_{v \in X : u \in B(v)} s_{out}(v) \geq \frac{3 s_{in}(u)}{4}.$$

Thus,

$$\sum_{v \in X : u \in B(v)} s_{out}(v) > 3 \sum_{v \in X : u \in T(v)} s_{out}(v).$$

If we sum both sides of the above inequality over all bottom heavy nodes, we have that $S_{bb} \geq 3S_{tb}$. We also know that $S_{bb} \leq (1/2)S$. Thus, $S_{tb} \leq (1/6)S$. Now,

$$S_{tt} + S_{tb} \geq (1/2)S.$$

Thus, $S_{tt} \geq (1/2 - 1/6)S = (1/3)S$. □

We can now use these results to prove Lemma 4.3 and also Theorem 4.1 in exactly the same manner as [13]. The only difference between the two proofs is that in our proof Δ is $\max\{\min(t(v), d(v)) \mid v \in V\}$, a global upper bound on the constrained span for any node in any round, while in [13] Δ is the maximum degree of the graph, also a global upper bound on the span of a node in the graph.

4.2. Improved ECDS (iECDS)

We will now describe the iECDS protocol. iECDS improves upon ECDS by using a *rounded constrained span* $\hat{c}(x)$. The rounded constrained span $\hat{c}(x)$ is the smallest power of 2 that is at least as much as $c(x)$. Using the rounded span generates larger connected components and thus will lead to a clustered network in fewer rounds. By using the rounded constrained span more nodes will become candidates in the early rounds and therefore more cluster heads will be elected. This means that the entire network is clustered in fewer rounds.

4.2.1. iECDS Algorithm

We provide the entire iECDS algorithm in Algorithm 2.

4.2.2. Complexity Analysis

The proof for iECDS follows the proof for ECDS almost exactly. The only change that it requires is for Lemma 4.2, substituting ”rounded span” for ”span”.

Lemma 4.6: All candidates in a connected component of H must have the same rounded span.

Proof: Let v_1 and v_2 be two candidates in a connected component of H. Consider a path p from v_1 to v_2 in H. Then there cannot be two consecutive nodes in p such that both are non-candidates. (This is because at least one end-point of each edge in H is a candidate

Algorithm 2 Improved ECDS Algorithm

```
Require: Graph $G = (V, E)$, constraint $r(v_i)$ on vertices
Ensure: Subset $D \subseteq V$, set of currently chosen vertices
Span calculation: Compute the constrained span $c(x)$ by computing the minimum of the constraint and the number of uncovered neighbors of $x$. Also, compute $\hat{c}(x)$, the rounded constrained span as the smallest power of 2 that is at least as much as $c(x)$.
Candidate selection: Compute whether $\hat{c}(x)$ is at least as much as the constrained span of each node within a distance of 2 from $x$. If so, $x$ is a candidate.
Constrained out-support calculation: If $x$ is a candidate, compute the constrained out-support of $x$ as follows: If $c(x) = 0$, let $s_{out}(x) = 0$. Else,
    $s_{out}(x) = \frac{c(x)}{||N(x) - C||}.$
    Note that $||N(x) - C||$ is the number of uncovered neighbors of $x$.
Constrained in-support calculation: If $x$ is an uncovered node, let $A(x)$ be the set of neighbors of $x$ that are candidates. Compute the constrained in-support $s_{in}(x)$ of $x$ as
    $s_{in}(x) = \sum_{y \in A(x)} s_{out}(y).$
Dominator selection: If $x$ is a candidate, find the median $m$ of $\{s_{in}(y) \mid y \in N(x) - C\}$. Let $p = 1/m$. With probability $p$, add $x$ to $D$.
Neighbor selection: If $x$ is selected, add $x$ to $D$, and for each neighbor $y \in N(x) - C$, select $y$ with probability $s_{out}(x)$ and add it to $V_x$. Set $C = \bigcup_{x \in D} V_x$.
Go to the next round.
```

node.) Since any two candidates within a distance of 2 must have the same rounded span, we have that all candidates that lie on p must have the same rounded span. And it follows that all candidates in a connected component of H must have the same rounded span. □

This change does not improve the theoretical bound, but our simulations show an improvement in the clustering algorithm, as can be seen in Section 6.

4.3. ECDS for k-hop Clusters (kECDS)

Our next improvement to ECDS allows the creation of clusters where a node can be up to k-hops from the cluster head. ECDS and iECDS required that each node can reach its cluster head within one hop. k-hop clusters will improve scalability, especially in very large networks. In very large networks, single hop clustering requires a large number of cluster heads and could lead to the same kind of problems as can be found in unclustered networks [6]. Cluster heads use additional energy, thus it is desirable to minimize the number of cluster heads. Allowing k-hops within a cluster will reduce the number of cluster heads. From the definition of dominating set, each node is either a cluster head or a neighbor of a cluster head. In k-hop clustering that definition is relaxed. A node is either a cluster head or at most k-hops from a cluster head. This allows clusters to be more spread out and requires fewer cluster heads to cover the same number of nodes. It also reduces the frequency of single node clusters, because a node has more clusters to join.

4.3.1. kECDS Algorithm

For kECDS we change the definition of neighborhood of x. $N(x)$ denotes the set of neighbors of x within k hops of x and includes x itself. The details for the kECDS algorithm are given in 3.

4.3.2. Complexity Analysis

The proof for kECDS follows the proof for iECDS almost exactly. The only change that it requires is for Lemma 4.6, substituting $k + 1$ for 2.

Lemma 4.7: All candidates in a connected component of H have the same rounded span.

Proof: Let v_1 and v_2 be two candidates in a connected component of H. Consider a path p from v_1 to v_2 in H. Then there cannot be two consecutive nodes in p such that both are non-candidates. (This is because at least one end-point of each edge in H is a candidate node.) Since any two candidates within a distance of $k + 1$ must have the same rounded span, we have that all candidates that lie on p must have the same rounded span. And it follows that all candidates in a connected component of H must have the same rounded span. □

This change does not improve the theoretical bound, but our simulations show an improvement in the clustering algorithm, as can be seen in Section 6.

4.4. Multi-Path ECDS (mECDS)

Our next improvement to the ECDS algorithm allows the use of multipath routing intra and inter cluster. Multipath routing is an energy efficient, load-balanced, fault-tolerant and

Algorithm 3 k-Hop ECDS Algorithm

```
Require: Graph $G = (V, E)$, constraint $r(v_i)$ on vertices
Ensure: Subset $D \subseteq V$, set of currently chosen vertices
Span calculation: Compute the constrained span $c(x)$ by computing the minimum of the constraint and the number of uncovered neighbors of $x$. Also, compute $\hat{c}(x)$, the rounded constrained span as the smallest power of 2 that is at least as much as $c(x)$.
Candidate selection: Compute whether $\hat{c}(x)$ is at least as much as the constrained span of each node within a distance of $k + 1$ from $x$. If so, $x$ is a candidate.
Constrained out-support calculation: If $x$ is a candidate, compute the constrained out-support of $x$ as follows: If $c(x) = 0$, let $s_{out}(x) = 0$. Else,
    $$s_{out}(x) = \frac{c(x)}{||N(x) - C||}.$$
    Note that $||N(x) - C||$ is the number of uncovered neighbors of $x$.
Constrained in-support calculation: If $x$ is an uncovered node, let $A(x)$ be the set of neighbors of $x$ that are candidates. Compute the constrained in-support $s_{in}(x)$ of $x$ as
    $$s_{in}(x) = \sum_{y \in A(x)} s_{out}(y).$$
Dominator selection: If $x$ is a candidate, find the median $m$ of $\{s_{in}(y) \mid y \in N(x) - C\}$. Let $p = 1/m$. With probability $p$, add $x$ to $D$.
Neighbor selection: If $x$ is selected, add $x$ to $D$, and for each neighbor $y \in N(x) - C$, select $y$ with probability $s_{out}(x)$ and add it to $V_x$. Set $C = \bigcup_{x \in D} V_x$.
Go to the next round.
```

reliable routing approach [14]. Multipath routing can be applied to routing from cluster heads to the base station to improve reliability and provide energy efficiency and fault-tolerance. Additionally, a multipath approach can also be used from each node to its cluster head. Using multiple cluster heads ensures that a node has another cluster head available when one fails even when single path routing is used. However, we have to make the assumption that the link between a node and its cluster head is no more reliable then any other link in the network. By using multipath routing, these type of problems can be mitigated. Most clustering protocols could be extended to work with multipath routing. Our extension of ECDS will lead to better performance and more energy savings.

4.4.1. mECDS Algorithm

In mECDS, each node has a coverage requirement. The coverage requirement is an indicator of the number of cluster heads a node desires. This coverage requirement may be the same throughout the network or it is decided by each node depending on the link quality to its neighbors. A cluster head will chose to cover a node with larger coverage requirements first. This will result in energy efficient clustering, while ensuring that all nodes have have the maximum coverage possible which will allow for largest energy savings while using multipath routing. The details can be found in Algorithm 4. Each round has several steps. During the first step, each node calculates its constrained span. A node is considered uncovered until it has the number of cluster heads it requires or all its neighbors are cluster heads. With every additional cluster head a node acquires, it becomes more “covered”. A node without a cluster head is completely uncovered. A node with one cluster head is $1/k$ covered, with two cluster head $2/k$ covered and so on, where k is the coverage requirement. A node that has a constrained span at least as large as any node in its 2 hop neighborhood elects itself as a candidate. Each candidate calculates its out-support, a measure of its ability to cover other nodes. Each node calculates its in-support, an indicator of the neighborhood’s ability to cover the node. A candidate becomes a cluster head with probability $1/m$, where m is the median in-support of all nodes in its neighborhood. A node joins a cluster with probability $1/in - support$.

5 RELATED WORK

In [15], cluster heads are chosen so that the energy consumption over the entire network is even, ensuring that the network lives as long as possible. A fixed number of cluster head candidates are selected and the cluster heads with the most residual energy are chosen from that set. A node will choose a cluster head to ensure the overall energy consumption in the entire network is even. Our algorithm, on the other hand, requires only local information about the topology and residual energy.

In [8], a new, fully distributed approximation algorithm based on LP relaxation techniques is presented. For an arbitrary parameter k and maximum degree Δ, the algorithm computes a dominating set of expected size $O(k\Delta^{2/k})$(log $\Delta\,|DS_{OPT}|$)in $O(k^2)$

Algorithm 4 Distributed Multi-Path Randomized Greedy Algorithm

```
Require: Graph $G = (V, E)$, constraint $t(v_i)$ on vertices, the coverage requirement $k$ for a node, the number of current cluster heads $k_i$ of a node
Ensure: Subset $D \subseteq V$, set of currently chosen vertices
1: Span calculation: Compute the constrained span $c(x)$ by computing the minimum of the constraint and the sum of "coverdness" ($k_i/k$) of the neighbors.
2: Candidate selection: Compute whether $c(x)$ is at least as much as the constrained span of each node within a distance of 2 from $x$ . If so, $x$ is a candidate.
3: Constrained support calculation: Let $A(x)$ be the set of neighbors of $x$ that are candidates. Each $y \in A(x)$ computes its constrained out-support:
       $$s_{out}(x) = \frac{c(y)}{\sum (k - k_i/k)}.$$
   Every node $x$ computes its constrained in-support $s(x)$ as
       $$s_{in}(x) = \sum_{y \in A(x)} s_{out}(y).$$
   Note that $\sum (k - k_i/k)$ is the sum of coverage needs that lie in the neighborhood of $y$.
4: Dominator selection: If $x$ is a candidate, find the median $m$ of $\{s_{in}(y) \mid y \in \sum (k - k_i/k)\}$. Let $p = 1/m$. With probability $p$, add $x$ to $D$.
5: Neighbor selection: If $x$ is selected as cluster head, for each neighbor $y \in \sum (k - k_i/k)$, select $y$ with probability $1/s_{in}(x)$ such that $y$ isn't already covered by $x$.
6: Go to the next round.
```

rounds where each node has to send $O(k^2\Delta)$ messages of size $O(log\Delta)$. This is the first algorithm that achieved a non-trivial approximation ratio in a constant number of rounds.

The work described in [16] is a primal-dual based distributed algorithm for the weighted, capacitated vertex cover problem. In [16] each vertex is assigned a weight, as well as a capacity, and the goal is to minimize the sum of the weights without exceeding the capacity of any vertex. The authors provide a $(2 + \epsilon) OPT$ approximation algorithm. Additionally the running time of the algorithm is shown to be $O(\log(nW)/\epsilon)$ where n is the number of nodes and $W = wt_{max}/wt_{min}$ is the ratio of the largest weight to the smallest weight.

A randomized distributed algorithm that runs in $O(\log n \log \Delta + 1)$ rounds and where the size of the dominating set obtained is, with high probability, within $O(\log n)$ of the optimal, is presented in [13]. Our distributed algorithm is based on this algorithm and extends its ideas to constrained vertices and applies it to wireless sensor networks.

In [17], an identity-based heuristic to form d-clusters in wireless ad-hoc networks is presented in which d is a parameter. When the heuristic terminates, a node is either a cluster head or at most d hops away from its cluster head.

A fast, distributed algorithm is presented in [18]. It is used to compute a small k-dominating set D (for any fixed k) and its induced graph partition (which breaks the graph into radius k clusters centered around the vertices of D). The time complexity of the algorithm is $O(k \ \log^* n)$, where $\log^*$ is the inverse Ackermann function.

For the special family of graphs that represent ad hoc wireless networks modeled as unit disk graphs, [19] introduces a two phase distributed polynomial time and message complexity k-clustering approximation solution with $O(k)$ worst case ratio over the optimal solution.

In [20], a series of approximation algorithms for finding a small, weakly-connected dominating set (WCDS) in a given graph is presented for use in clustering mobile ad hoc networks. The main contribution of the work is a completely distributed algorithm for finding small WCDS. Our work focuses on wireless sensor networks, whose challenges differ from those of ad-hoc networks.

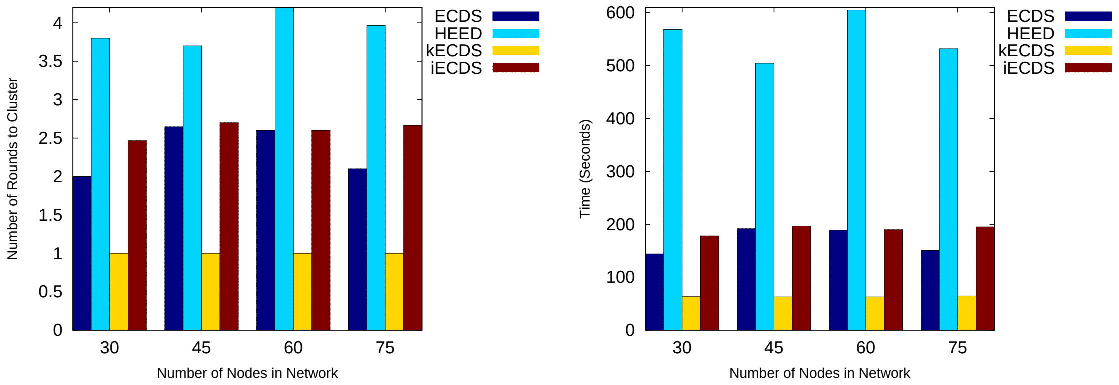
In [21], the authors provide three approximation algorithms for the minimum connected dominating set (MCDS) in mobile ad hoc networks. The algorithms provide approximation guarantees of $2H(\Delta)+1$ and $2H(\Delta)$, where $H(\Delta) = \sum_{i=1}^{\Delta} 1/i \leq \ln \Delta + 1$. The guarantee of $c + 1$, applies to graphs where the maximum degree is Δ and c is some constant such that $\Delta \leq c$. Our algorithm also considers dominating sets and provides algorithms suited for wireless sensor networks.

The connected minimum dominating set is considered in [22]. The authors provide two approximation algorithms which achieve approximation factors of $2H(\Delta) + 2$ and $H(\Delta) + 2$, where Δ is the maximum degree of the graph and H is the harmonic function.

Several distributed poly-logarithmic time algorithms are presented in [5]. The algorithms compute connected and weakly connected dominating sets with an approximation factor of $O(\log \Delta)$, where Δ is the maximum degree of the graph.

6 EXPERIMENTAL EVALUATION

In order to test the clustering algorithms, we implemented them in TinyOS and ran simulations in TOSSIM [23]. We compared the iECDS and kECDS algorithms against the ECDS and HEED algorithms [24]. A random topology was used for each simulation. We ran a simulated 15 minutes for network sizes of 30, 45, 60, and 75 nodes. Our algorithms are independent of the routing protocol used, but for these experiments we use the Surge multi-hop application that is part of TinyOS. HEED and the ECDS algorithms also use the Surge multi-hop routing protocol. Each node generates a reading every 20 seconds. The cluster heads aggregate the readings and forward a single message. Surge uses a link estimation and parent selection (LEPS) mechanism to determine multi-hop routes. All traffic received at each node is monitored and used to update the internal neighbor table. The neighbor table tracks all neighbors and selects the next hop based on shortest path semantics. The default destination is the base station. We use a credit-point system for updating the node energy budget as used with iHEED [7]. Energy is used for tasks such as sending and receiving and points are deducted proportional to the actual amount of energy used. Each nodes starts with the same amount of points and for each send/receive an amount proportional to the size of the message is deducted. In our implementation, cluster heads receive many more messages from nodes in their cluster than it sends messages to nodes in its cluster. All messages are sent with the same power level, therefore we do not consider the transmission distance when determining the cost of each send/receive [25]. For ECDS type algorithms, the initial energy allows for a constraint of about 20. Whenever the network re-clusters, the constraint is updated and is based on the energy available at each node. All kECDS algorithms run with $k = 2$, that is, a node can be up to two hops from its cluster head. For each network size, the experiments were repeated 30 times. We measured the number of rounds the algorithm executed until the entire network was clustered. We compared the time of the first node's deaths to the last node's death. Having all nodes die at approximately the same time provides the most useful WSN. Additionally, we measured the time it took for the entire network to cluster. A fast clustering algorithm ensures a useful WSN. We consider the energy used during the simulation and we look at the energy usage per useful message received at the base station.



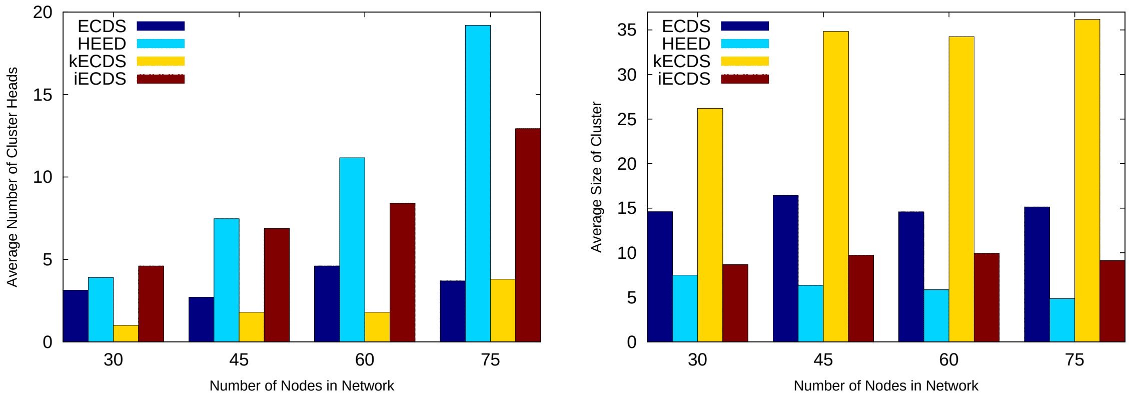
2(a) Rounds to Cluster

2(b) Time to Cluster

Fig. 2. Rounds and Time to Cluster

6.1. Cluster Generation

In a distributed environment it is important to evaluate how long it takes for a clustering protocol to finish. There are two measurements for WSN: time and the number of rounds of execution. Figure 2(a) shows the average number of rounds to cluster the network for various sizes. An ideal distributed clustering algorithm will cluster in a constant number of rounds. Both the ECDS and the HEED algorithms execute in a constant number of rounds, but the ECDS algorithms finished in fewer rounds. The algorithm depends on the routing information obtained from the (independent) routing protocol. This routing information may not be complete, especially in the earlier rounds. Incomplete routing information will exclude some nodes from joining a cluster. Hence, additional rounds may be required. kECDS finishes in the fewest number of rounds as nodes can join cluster heads up to k-hops away. Similar behavior can be seen in Figure 2(b), which shows the average time it took for the networks to cluster. Clearly, the number of rounds and the time are related and both are important measurements. An algorithm that runs over several short rounds may still outperform an algorithm that runs in a constant number of long rounds. Again, it is important that an algorithm takes a constant amount of time, independent of the size of the network. Both the ECDS and the HEED algorithm take a constant amount of time, but the ECDS algorithms are faster.



3(a) Dominating Set Size

3(b) Cluster Size

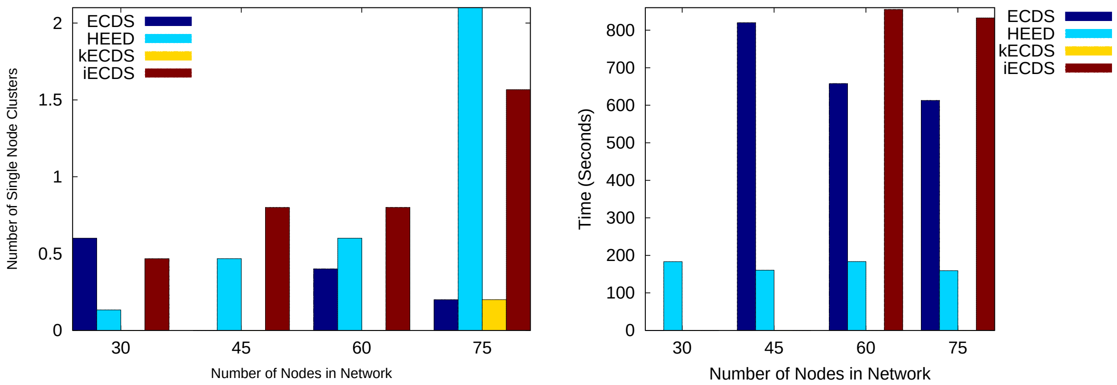
Fig. 3. Dominating Set and Cluster Size

6.2. Cluster Goodness

Figure 3(a) shows the average size of the dominating set. The dominating set is the number of cluster heads selected for each simulation run. Each node starts with the same amount of energy, an amount that can support about 20 nodes in a cluster. The number of cluster heads for kECDS grows slowly which means that the size of each cluster grows slowly as the network gets larger. Knowing that a (near) constant number of cluster heads are required can be used to support a few nodes with additional power, allowing them to act as cluster heads for longer periods of time. iECDS and HEED increase the number of cluster heads as the network grows; this indicates that the cluster size remains relatively constant independent of the size of the network. Scalability is improved when clusters are of similar size regardless of network size. iECDS uses the rounded constraint span, which allows more candidates at each round, thus more cluster heads are selected. Figure 3(b) shows the average number of nodes in each cluster. In kECDS the number of nodes assigned to a cluster grows slowly, as predicted. The size decreases asymptotically for HEED and seems to lead to very small clusters in large networks. As discussed, the cluster size remains constant for iECDS. Not only the number of nodes in each cluster and the number of cluster heads matter, but also how many of those clusters are single-node clusters (clusters in which the cluster head is the only node). Single node clusters do not improve performance, but they are generally unavoidable. A good algorithm will minimize the number of such clusters. Figure 4(a) shows the average number of single node clusters. For ECDS the number of single node clusters decreases as the size of the network grows. ECDS choses only neighbors which are “near” as cluster heads; some nodes will not be near a cluster head and thus create single node clusters. As the network grows, each node has more opportunities to find a near cluster head, hence the decrease. On the other hand, HEED’s single node clusters increase in number as the network grows. kECDS has very few or no single node clusters. In kECDS a node can join a cluster even if it is not “near” the cluster head, as long as it within k-hops from the cluster head. This allows most nodes the option of joining some cluster, reducing the number of single node clusters. iECDS, on the other hand, has an increase in single node clusters. In iECDS more candidates and thus (more) cluster heads are elected at each round. This can lead to a situation where a node finds itself surrounded by cluster heads, without the ability to join any one of them. Such a node will elect itself as cluster head as well.

6.3. Lifetime of Sensor Nodes

In a wireless sensor network, the early death of some nodes can disconnect other nodes from the base station. This situation can lead to a reduced usefulness of the network because some data cannot reach the base station. We measure lifetime in two ways: (1) the time at which the first node dies and (2) the time at which the last node dies. The time at which the first node dies is important because it can lead to a disconnection of part of the network. The time at which the last node dies shows how long nodes are able to run the protocol. Figure 4(b) shows the time at which the first node died. The time of the first death asymptotically decreases in ECDS and is constant for HEED. kECDS and iECDS have no death during the simulation. iECDS has more cluster heads with smaller cluster sizes, leading to improved work distribution and energy usage. kECDS clusters faster, which reduces the need for, and the number of, clustering and other “routing” messages. This reduces the overall energy consumption giving the network a longer lifetime. Figure 5(a) shows the time of death for the last node in the network. It is equally important that all nodes die around the same time. A single node that outlives others by a large margin is of little use. It can be estimated that the lifetimes will be similar for the ECDS and HEED algorithms in large networks. For both ECDS and HEED the first and last deaths are within 200 seconds of each other, indicating an even energy consumption across the network.



4(a) Average Number of Single Node Clusters

4(b) First Death

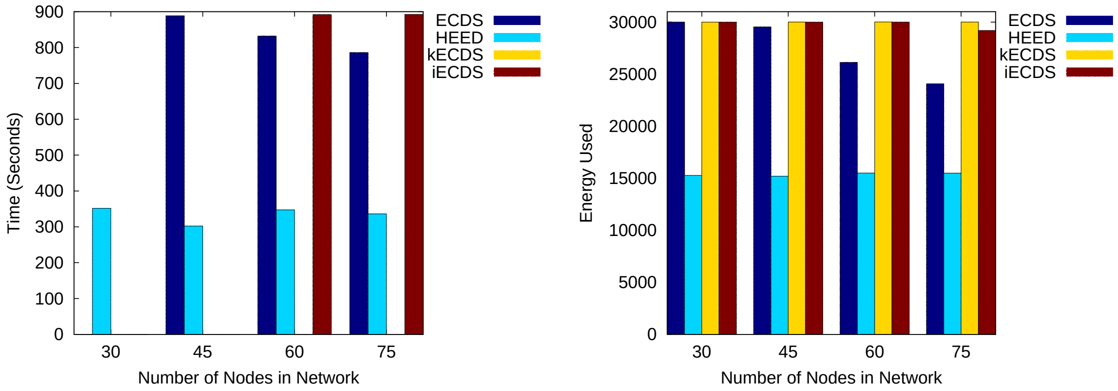
Fig. 4. Cluster Size, Average Number of Single Node Clusters, and First Death

6.4. Energy Consumption

The amount of energy used during the execution of a protocol is very important in sensor networks. Figure 5(b) shows the average energy consumption for the four protocols. The energy consumption of the HEED algorithm is linear, while the energy consumption of the ECDS algorithm is asymptotically decreasing. Both the kECDS and the iECDS algorithm have linear consumption. As the networks grow larger, the energy consumption for ECDS and HEED will be similar while kECDS and iECDS will remain about the same.

In sensor networks, the energy consumption for each message received should be considered in addition to the overall energy consumption. A sensor network that uses very little energy is not useful if it does not produce an adequate amount of data. Figure 6(a) shows the average energy consumption for each message sent. Since the ECDS algorithms cluster faster, they generate more usable data. All ECDS type algorithms outperform HEED when the energy consumption is viewed in respect to the amount of data received.

Figure 6(b) we show the average energy used for communication per message sent. Because communication is the most expensive operation in sensor networks, it is important to measure its cost. This measurement must be considered in terms of the amount of data generated in the network. A network with high communication energy consumption that generates little data indicates that the protocols used requires too much communication energy. Figure 7(a) and Figure 7(b) show the respective maximum and minimum energy used by any node for each network size.



5(a) Last Death

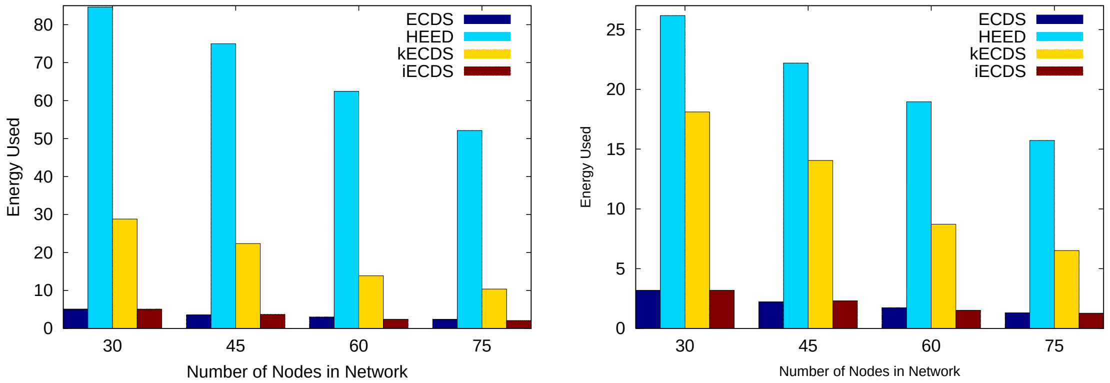
5(b) Average Energy Consumption

Fig. 5. Last Death and Average Energy Consumption

It is important to have even energy consumption across the network. For all the ECDS type algorithms, the maximum and minimum amount of energy consumed is close, while HEED consumes at twice the rate of other algorithms. This is a strong indicator that the energy consumption in the ECDS type algorithms is very even across the network, while HEED has "hotspots" which use energy at twice the speed then other nodes. In HEED this is a contributing factor to the early death of some nodes and the overall reduced lifetime of the network.

7 CONCLUSION AND OPEN PROBLEMS

In this paper, three improvements to the ECDS algorithm are presented to address the problem of energy constrained clustering for wireless sensor networks. The first improvement extends the ECDS algorithm to clusters where a node can be up to k hops from its cluster head. This version is named kECDS. The second algorithm improves ECDS by introducing the rounded span which allows a larger cluster head selection during each round of the algorithm. This algorithm is called iECDS. The last algorithm extends ECDS to be used with a multi-path routing protocol. This version of the algorithm allows a node to choose up to m cluster heads, which provides robustness by enabling multi-path routing. With mECDS, multi-path routing can be used not only from the cluster head to the base station, but also from the node to one or more cluster heads. We proved that the



6(a) Average Energy Consumption per Message

6(b) Average Communication Energy Consumption per Message

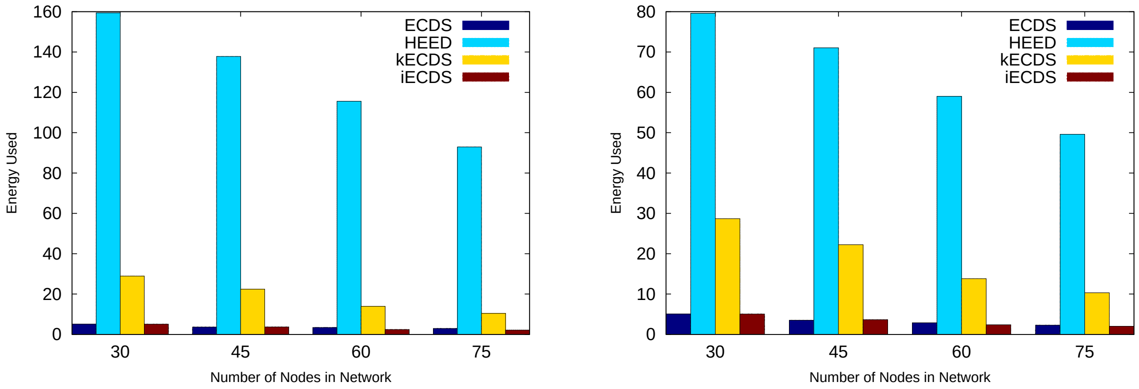
Fig. 6. Average Energy and Average Communication Energy per Message

kECDS and iECDS algorithms run in $O(\log n \log \Delta)$ rounds whp. The kECDS and iECDS algorithms perform well on the random graphs in our simulations. Our simulations showed that our algorithms perform very well in terms of time to cluster, cluster size, and energy consumption. We compared our algorithms with the HEED algorithm. They outperformed HEED in terms of cluster size, time to cluster, and energy consumption per message sent. Future work will include extending the ECDS algorithm to work in a secure environment. Currently, each node decides whether or not to become a cluster head. Each node has to trust the information received from other nodes, which it uses to determine is cluster head status. A node could avoid becoming a cluster head by misrepresenting its information, or it may choose to be cluster head and use its status as message router to drop messages and disable the network. We plan on extending the ECDS algorithms so that the cluster can be formed securely, with a guarantee that all messages are received by the base station.

REFERENCES

[1] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, “An application-specific protocol architecture for wireless microsensor networks,” *Wireless Communications, IEEE Transactions on*, vol. 1, no. 4, pp. 660–670, 2002.

[2] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-efficient communication protocol for wireless microsensor networks,” 2000, pp. 10 pp. vol.2+.



7(a) Maximum Energy Consumption per Message

7(b) Minimum Energy Consumption per Message

Fig. 7. Average Communication Energy, Maximum and Minimum Energy

[3] C. Duan and H. Fan, "A distributed energy balance clustering protocol for heterogeneous wireless sensor networks," in *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*, 2007, pp. 2469–2473.

[4] J. Wu and H. Li, "Domination and its applications in ad hoc wireless networks with unidirectional links," in *Parallel Processing, 2000. Proceedings. 2000 International Conference on*, 2000, pp. 189–197.

[5] D. Dubhashi, A. Mei, A. Panconesi, J. Radhakrishnan, and A. Srinivasan, "Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons," *J. Comput. Syst. Sci.*, vol. 71, no. 4, pp. 467–479, November 2005.

[6] A. Youssef, M. Younis, M. Youssef, and A. Agrawala, "Wsn16-5: Distributed formation of overlapping multi-hop clusters in wireless sensor networks," in *Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE*, 2006, pp. 1–6.

[7] O. Younis and S. Fahmy, "An experimental study of routing and data aggregation in sensor networks," 2005, pp. 8 pp.+.

[8] F. Kuhn and R. Wattenhofer, "Constant-time distributed dominating set approximation," *Distrib. Comput.*, vol. 17, no. 4, pp. 303–310, May 2005.

[9] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds. Plenum Press, 1972, pp. 85–103.

[10] F. Kuhn, T. Moscibroda, and R. Wattenhofer, "Unit disk graph approximation," in *DIALM-POMC '04: Proceedings of the 2004 joint workshop on Foundations of mobile computing*. New York, NY, USA: ACM, 2004, pp. 17–23.

[11] Harry, M. V. Marathe, V. Radhakrishnan, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns, "Nc-approximation schemes for np- and pspace-hard problems for geometric graphs," *J. Algorithms*, vol. 26, no. 2, pp. 238–274, February 1998.

[12] S. Banerjee and S. Khuller, "A clustering scheme for hierarchical control in multi-hop wireless networks," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, 2001, pp. 1028–1037 vol.2.

[13] L. Jia, R. Rajaraman, and T. Suel, "An efficient distributed algorithm for constructing small dominating sets," *Distrib. Comput.*, vol. 15, no. 4, pp. 193–205, December 2002.

[14] W. Cheng, K. Xing, X. Cheng, X. Lu, Z. Lu, J. Su, B. Wang, and Y. Liu, "Route recovery in vertex-disjoint multipath routing for many-to-one sensor networks," in *MobiHoc '08: Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM, 2008, pp. 209–220.

[15] M. Ye, C. Li, G. Chen, and J. Wu, "Eecs: an energy efficient clustering scheme in wireless sensor networks," in *Performance, Computing, and Communications Conference, 2005. IPCCC 2005. 24th IEEE International*, 2005, pp. 535–540.

[16] F. Grandoni, J. Könemann, A. Panconesi, and M. Sozio, "Primal-dual based distributed algorithms for vertex cover with semi-hard capacities," in *PODC '05: Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*. New York, NY, USA: ACM, 2005, pp. 118–125.

[17] A. D. Amis, R. Prakash, T. H. P. Vuong, and D. T. Huynh, "Max-min d-cluster formation in wireless ad hoc networks," vol. 1, 2000, pp. 32–41 vol.1.

[18] S. Kutten and D. Peleg, "Fast distributed construction of ¡italic¿k¡/italic¿-dominating sets and applications," in *PODC '95: Proceedings of the fourteenth annual ACM symposium on Principles of distributed computing*. New York, NY, USA: ACM, 1995, pp. 238–251.

[19] Y. Fernandess and D. Malkhi, "K-clustering in wireless ad hoc networks," in *POMC '02: Proceedings of the second ACM international workshop on Principles of mobile computing*. New York, NY, USA: ACM, 2002, pp. 31–37.

[20] Y. P. Chen and A. L. Liestman, "Approximating minimum size weakly-connected dominating sets for clustering 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. 165–172.

[21] B. Das and V. Bharghavan, "Routing in ad-hoc networks using minimum connected dominating sets," in *Communications, 1997. ICC 97 Montreal, 'Towards the Knowledge Millennium'. 1997 IEEE International Conference on*, vol. 1, 1997, pp. 376–380 vol.1.

[22] S. Guha and S. Khuller, "Approximation algorithms for connected dominating sets," *Algorithmica*, vol. 20, no. 4, pp. 374–387, April 1998.

[23] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: accurate and scalable simulation of entire tinyos applications," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*. New York, NY, USA: ACM Press, 2003, pp. 126–137.

[24] O. Younis and S. Fahmy, "Heed: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *Mobile Computing, IEEE Transactions on*, vol. 3, no. 4, pp. 366–379, 2004.

[25] A. Boukerche, X. Fei, and R. B. Araujo, "An optimal coverage-preserving scheme for wireless sensor networks based on local information exchange," *Computer Communications*, vol. 30, no. 14-15, pp. 2708–2720, October 2007.

III. PRACTICAL ALGORITHM FOR DATA SECURITY (PADS) IN WIRELESS SENSOR NETWORKS

Julia Albath
University of Missouri-Rolla
Department of Computer Science
jgadkc@umr.edu

Sanjay Madria
University of Missouri-Rolla
Department of Computer Science
madrias@umr.edu

ABSTRACT

When data are generated in sensor networks, highspeed data streams travel through the network. Traditional security approaches are often unable to keep up with the rates of the streams or they introduce overhead, which shortens the life of the network. The approach proposed in this paper is one that solves the problems posed above. By embedding a one-time pad, the actual value is distorted enough to make any information gleaned from eavesdropping useless to an attacker. The use of the one-time pad ensures that the data were indeed received from a particular sensor, and it gives adequate protection against injected messages. The simulation shows this approach provides security with negligible overhead while the throughput is similar to the same network without security.

1. INTRODUCTION

There are many possible applications of sensor networks from environmental monitoring to applications in military and homeland security [2]. Because radio communication is the most expensive operation in terms of energy usage, research has focused on finding ways to conserve energy [5]. Until recently, data security has received little consideration. In order for ubiquitous computing to become a reality, system security and privacy protection need to be assured. Securing data streams in sensor networks is important because traditional encryption and authentication protocols such as TinySec are often unable to keep up with high stream rates, and they deplete the network of energy too quickly [8].

The challenge in applying security to wireless sensor networks lies in the need to balance data integrity, confidentiality, and availability as well as preserving constrained energy resources. In sensor networks where radios are used for communication, an attacker can easily acquire data by eavesdropping anywhere the signal is transmitted. New security devices, methods, and approaches that safeguard sensitive data are needed. Among the current research directions are efficient routing protocols [1], security schemes [8], and data management tools such as aggregation and data fusion.

Previous works in data security include implementing link layer security for sensor networks and public key cryptography using elliptic curves [8, 13]. These existing security protocols must be adapted to work in sensor networks because sensor networks have constraints on energy, communication and computation. Many other security protocols cannot be adapted for sensor networks and thus we require new ways of providing security. In [17], the authors consider the case of resilient rights protection of data streams through watermarks. The protocol provides a way to identify the original owner of a data stream. Given that the stream is available, this technique can be used to determine whether the integrity of the data has been compromised; however no protection against eavesdropping is provided. TinySec provides security through traditional link layer encryption and authentication schemes using the RC5 or SkipJack ciphers [8]. As with any cryptographic security scheme, TinySec increases the payload size of the packet, which results in increased energy consumption. In [14], the authors introduce SPINS, a three-part approach that provides authenticated streaming broadcast, data confidentiality, two-party data authentication, and data freshness as well as an authenticated routing protocol. The SNEP protocol of the SPINS suite of protocols provides data confidentiality. SNEP has low communication overhead by only adding 8 bytes to the packet. The rates at which data are produced may be much faster than the rates at which a security scheme can process and secure the data. While there is a need to tightly encrypt some type of messages, in which case a reduction in availability is often acceptable, other types of data may only require a light security scheme but demand high availability. Many security schemes have considered different approaches for messages in different levels in the network, but have not considered different levels of security depending on the message type. This paper presents a method to provide protection against passive eavesdropping by employing confidential transmissions of data messages. For each transmission, a one-time pad (OTP) is created.

A secret key and the MAC calculated over the data are used to create the OTP. When the OTP is combined with the data, data encryption and data integrity are achieved at the same time. The complete algorithmic details are given in Section 4. The PADS protocol only adds 4 bytes, making it even more efficient than SNEP. Just as with SNEP, PADS also provides data confidentiality and semantic security. Semantic security implies that eavesdroppers cannot infer the data even if they see the same data encrypted several times. The PADS protocol has dynamic key sizes and the ability to change the number of bytes to be encrypted. By using one-to-one routing and pair-wise shared keys, data integrity is guaranteed.

At the worst, the algorithms run in linear time on the size of the reading, regardless of the size of the network. The correctness of the algorithms was proven and the theoretical analysis of the energy usage shows that the application is appropriate for sensor networks. Simulations have shown this protocol to be suitable for sensor networks. Refer to Section 6 where the simulation results are provided. The simulation shows that the overhead due to the algorithm is negligible, and the performance of a network with the protocol is similar to a network without additional security. Compared with a network using encryption and authentication using TinySec, the performance of a network using this study's protocol is much better in terms of throughput, delay and energy consumption per message received at the base station.

2. RELATED WORK

Researchers have investigated securing data in wireless sensor networks in order to ensure that the base station can trust the answers it receives. The authors in [15] propose the use of interactive proofs to force the aggregator to show that the answer previously provided is a good approximation of the true value. An aggregator is a node that applies an aggregate function such as a sum or average to all the data received from its child nodes. All types of stealthy attacks by the aggregator are considered. A stealthy attack is an attack where the aggregator wants the base station to accept results that are different from the true value, while at the same time evading detection. The aggregate-commit-prove (ACP) protocol works as follows. The aggregator collects the data and locally computes the aggregation results. Next, the aggregator calculates a commitment, which during the proving phase will show that the aggregator used the values provided by the sensor to calculate the aggregation value. Whenever the home server requests, the aggregator proves to the home server that the result is valid. The home server checks to see if the aggregator is cheating, in the sense that the aggregation result is or is not close to the correct result aggregated from the committed data. In contrast, our work concentrates on securing the data on the routing path. PADS provides security against an eavesdropper and would work well in conjunction with ACP to provide confidentiality and integrity in wireless sensor networks.

The work in [17] is a novel idea to provide copyright protection to data stream owners and authorized users. Consider the case where a stream is generated and safely transmitted from the sensors to the base station. A watermark is applied to the stream at the base station. The data are then transmitted to an authorized user. The owner and authorized users need a way to show that the data were generated by them and they want to prove that the stream was illegally obtained by the attacker. One commonly accepted way to prove ownership is the use of embedded watermarks. This technique works by embedding a watermark bit into major extremes, which are extremes that will survive any uniform sampling. Because the watermark bits are embedded in the major extremes, they can then be extracted and used to show copyright and ownership can be established. Extremes are chosen because even after alteration and aggregation, most extremes will be recoverable and able to ensure an overlap when rebuilding the watermark. During detection, all extremes, not just the major ones, are identified, and the same selection criteria as during embedding are used to identify the potential watermark recipients. For each selected extreme, its corresponding 1-bit watermark is extracted and the global watermark is gradually reconstructed. Similarly, PADS could be considered a watermark, which is embedded in the data, the difference being that a new watermark is generated for each data item.

An example of stream security is [19]. The author explores message authentication in sensor networks. He compares several authentication possibilities: end-to-end, hop-by-hop, and physical and virtual multipath authentication. While end-to-end authentication provides the greatest level of security, hop-by-hop authentication can be implemented with relatively little overhead, but it provides security only against a very restricted attacker. The author shows that physical multicast authentication provides a good intermediate level of security, and that virtual multicast authentication retains similar properties as physical multicast authentication but also has reduced energy demands.

In contrast to our work on securing messages, Zhu et al. introduce a protocol that provides protection against messages injected by an attacker with the goal to deceive the base station or to shorten the lifetime of the network by depleting the resources of the sensors in the network [21]. The authors present an interleaved hop-by-hop authentication scheme, which guarantees that the base station will detect injected messages as long as fewer than t nodes are compromised. When nodes are compromised, false messages are created and injected in the network.

3. SYSTEM MODEL AND ARCHITECTURE

Sensor networks can be modeled as graphs. Let $G = (V, E)$ be an undirected graph with a set of n nodes $V \subset \mathbb{R}^2$ in the Euclidean plane and a set of m edges $E \subset V^2$ [16]. The vertices in this model represent the nodes in the sensor network and the edges represent communication links between the nodes. While the network could be modeled as a directed graph, only an undirected graph is considered and so only bi-directional links are modeled. This study defines a round as a time interval in which each node is to send its gathered data to the base station. In some cases, a sensor will continuously generate, process, and send data as it monitors its environment.

A data stream is defined as an infinite set of values generated by a sensor. Furthermore, each sensor node generates one data packet each round. While in many cases the timestamp will be preserved, the notion of time is only introduced to show the sequential nature of such data streams. The timestamp information may or may not be routed with the data. In many cases the timestamps lose their meaning and as such, should not be considered as important.

A sensor network most often generates numeric data, but at times may produce data from other domains. However, this study consider all data only at the level of a string. All data, whether numeric or not, are just a sequence of ones and zeros, and thus this research technique will work equivalently on numeric or other data.

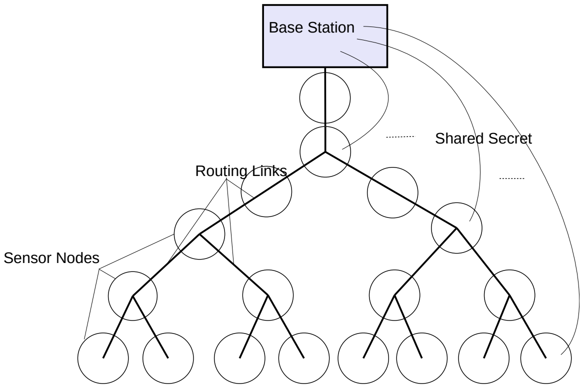


Fig. 3.1. Network Architecture

The following assumptions are made. Each node shares a secret key with the base station. This key is used to generate a new key for every transmission using a key derivation function. This approach provides protection against an attacker who can overhear transmissions and who can also inject false messages and nodes. The sensor nodes can communicate with a powerful base station via relay nodes. A simple multi-hop routing protocol is used [12] as shown in Figure 3.1. This study assumes that the base station and the sensors are time synchronized, which can be handled as described in [3]. Here sensor nodes synchronize with their parent nodes in a hierarchical structure until a synchronization chain has been built up to the root or base station. In cases where the sensors send data at periodic intervals, an explicit synchronization exists given that each packet has a sequential packet number. In aperiodic sensing of events the protocol in [3] can be used to keep the network synchronized.

Definition 3.1: A message authentication code (MAC) is the result of applying a public function to the input using a secret key. The MAC is of fixed length, is attached to the input and serves to prove integrity and authenticity of the input. A MAC is also known as a cryptographic checksum [18].

Definition 3.2: A MAC function is a function used to generate a MAC. A MAC function has three properties. It is a one-way function, that is, given a MAC it is computationally infeasible to find the original input. It offers weak collision restistance, that is, it is computationally difficult to find a second input ($\neq$ the first input) which produces the same MAC. It offers strong collision resistance, that is, it is computationally difficult to find two inputs which produce the same MAC[18].

Definition 3.3: A CBCMAC function is a function which uses a block cipher in cipher block chaining (CBC) mode to generate MAC codes. A CBCMAC is created by encrypting the input into a chain of blocks, such that each block is dependent on the previous block [18].

Definition 3.4: A one-time pad is a sequence of bits that is XOR to the reading in order to provide protection against eavesdropping.

Definition 3.5: k_i is the secret master key shared between node i and the base station. The value of k_i can be updated similar to updating keys in [22] and [14].

Definition 3.6: k_{ij} is the j^{th}secret key shared between node i and the base station. The key is periodically updated to guarantee freshness.

Definition 3.7: x_{ij} is the data value x generated at time j by node i. x_i is the current value of x generated by node i and $p(x_{ij})$ is the packet generated by node i at time j, which contains the sensor value x.

Definition 3.8: $b(x)$ is the bit size of the value x. $b(MAC)$ is the bit size of the Message Authentication Code (MAC).

Definition 3.9: α determines the length of the subsequence of the MAC. β determines the starting location of the aforementioned subsequence of the MAC. $\alpha, \beta \in \mathbb{Z}$.

4. ALGORITHMS

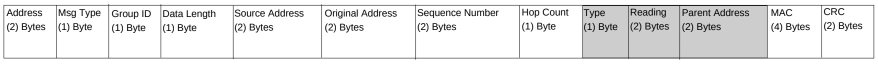
The one-time pad is constructed by the nodes using only information contained within the data packet and the secret key shared between the sender and the base station. First, the MAC (message authentication code) is calculated based on the sensor reading. Next, a one-time pad is constructed from the MAC and the key shared with the receiver. The MAC is attached to the sensor reading. The variables α and β are calculated based on the MAC, and a subsequence of the MAC is extracted and used to secure the sensor reading. The receiving node uses the attached MAC, removes the encryption, and calculates its own MAC. To show that the message has not been tampered with, the two $MACs$ are compared for equality.

4.1. Embedding Of A One-Time Pad

Algorithm 1 Basic Embedding Algorithm: **embed**

```
Require: $k_{ij}$, packet $p(x_i)$ of sensor value $x_i$
Ensure: $p(x_i)$ XOR with the one-time pad
calculate $MAC$ over $p(x_i)$ (excluding routing information, see Figure 4.1) and save in $p(x_i)$
$k_{time} = k_{ij}$ modified and time synced
$\alpha = k_{time} \mod b(MAC) + 1$
$\beta = k_{time} \mod (b(MAC) - \alpha + 1)$
$temp\_pad =$ substring of $MAC$ starting at $\alpha$ for $\beta$ bits
$\widetilde{pad} =$ Append $temp\_pad$ to $\widetilde{pad}$ until $\widetilde{pad}$ is of the same length as the data
$\widetilde{x_i} = x_i \, XOR \, \widetilde{pad}$
Replace $x_i$ with $\widetilde{x_i}$ in $p(x_i)$ and send
```

Refer to Algorithm 1 for details on the embedding algorithm. This algorithm calculates a MAC over the static part of the packet. Multi-hop routing needs to change the packet header to properly route the packet, so only the part of the packet that does not change is used, as shown in Figure 4.1. The calculated MAC is appended to the data and the secret key shared between the sender and the receiver is used to create a time synced key. For example, at time t, every t^{th} bit of the key is dropped. This ensures that an attacker has to be time synced with the network in order to break the encryption. Time synchronization is handled similarly to [20]. It is important to note that our algorithm will work without time synchronization, but time synchronization provides for dynamic adjustment to the keys, thus providing added security. The next step is to calculate the length of the substring and the starting position of the substring. This study calculates α by taking the secret key modulo the length of the MAC in bits. β is calculated by taking the key modulo the length of the MAC minus α plus 1; this ensures that the substring will exist. The substring is of length α bits, so the last α bits for the source string cannot be the starting point of the substring. The final steps involve generating the one-time pad by repeatedly concatenating the substring to a target string until the target string is of the same length as the static portion of the packet to be secured. In most cases that will be the payload plus some portion of the packet header as shown in Figure 4.1. The study then



Address (2) Bytes	Msg Type (1) Byte	Group ID (1) Byte	Data Length (1) Byte	Source Address (2) Bytes	Original Address (2) Bytes	Sequence Number (2) Bytes	Hop Count (1) Byte	Type (1) Byte	Reading (2) Bytes	Parent Address (2) Bytes	MAC (4) Bytes	CRC (2) Bytes

Fig. 4.1. Multihop packet structure. The fields shaded gray are protected by the MAC calculation.

encrypts the data that are transmitted by XORing the generated one-time pad to the value x_i and then transmits the packet.

4.2. Detection Of A One-Time Pad

The detection process works very similarly to the embedding process. This is necessary in order to find the embedded pad, remove it, and return to the original sensor value. The detection and removal of the one-time pad is done by the base station, because only the base station shares the secret key with the embedding sensor node.

Because sensor networks communicate via wireless mediums, there exists a natural probability of collisions, dropped packets, or corrupted packets. Any of these problems will be detected by this protocol. If the MAC or the payload of a packet is corrupted, then the MAC calculated by the receiver will not match the one from the sender.

In order to retrieve the original data, the detection process described in Algorithm 2 is used. This study uses the MAC calculated by the sender and stored in the packet to calculate the $\widetilde{pad}$ and remove it from the packet. Then the MAC is calculated just like the sender, and the two $MACs$ are compared. If they match, then processing the packet continues.

4.3. Example

The following is an example of Algorithm 1 executed at node X and Algorithm 2 executed at the base station, along with the intermediate steps. X senses an event, does the embedding, and transmits the data to the base station. Then the base station does the detection.

Step 1: Embedding at Node X. X senses an event and generates the value x_X = 00011110 10011011. X calculates the MAC with the secret key k_{ij} using SkipJack in the CBCMAC mode, as implemented by [8] over x_X.

Algorithm 2 Basic Detection Algorithm: **wm_detect**

```
Require: $k_{ij}$ packet $p(\widetilde{x_i})$ of sensor value
Ensure: $p((x_i)$ the original packet with the original sensor value $x_i$
 1: Take the $MAC$ calculated by the sender
 2: $k_{time} = k_{ij}$ modified and time synced
 3: $\alpha = k_{time} \mod b(MAC)$ + 1
 4: $\beta = k_{time} \mod (b(MAC) - \alpha + 1)$
 5: $temp\_pad =$ substring of $MAC$ starting at $\beta$ for $\alpha$ bits
 6: $\widetilde{pad} =$ repeatedly concatenate $temp\_pad$ to $\widetilde{pad}$ until $\widetilde{pad}$ is of the same length as the
    static portion of the packet.
 7: $x_i = \widetilde{x_i}\ XOR\ \widetilde{pad}$
 8: calculate $MAC$ over $p(x_i)$ and compare to $MAC$ received with packet
 9: if The two $MAC$'s match then
10:    Packet was not altered
11: else
12:    Packet was altered
13: end if
```

$MAC_{x_X,k_{ij}} =$ 01100000 01111100 11011111 11001010.

Algorithm1:Line3 α = k_{time} mod 32 + 1 = 00101001

Algorithm1:Line4 β = k_{time} mod (32 - α + 1) = 2

Algorithm1:Line5 $temp\_pad$ = subsequence of MAC starting at 29 for 2 = 11

Algorithm1:Line6 $\widetilde{pad}$ = 11111111 11111111

Algorithm1:Line7 $x_i(old)$ = 00011110 10011011 $x_i(new)$ = 11100001 01100100

Step 2: Detection at the base station. BS receives the value of x_i = 11100001 01100100 from node X with the stored MAC = 01100000 01111100 11011111 11001010.

Algorithm2:Line3 α = k_{time} mod 32 + 1 = 00101001

Algorithm2:Line4 β = k_{time} mod (32 - α + 1) = 2

Algorithm2:Line5 $temp\_pad$ = subsequence of MAC starting at 29 for 2 = 11

Algorithm2:Line6 $\widetilde{pad}$ = 11111111 11111111

Algorithm2:Line7 $x_i(old)$ = 11100001 01100100 $x_i(new)$ = 00011110 10011011

Algorithm2:Line8 BS calculates the MAC over $x_i(new)$ = 01100000 01111100 11011111 11001010.

Algorithm2:Line9 MAC's match, data was not altered.

5. SECURITY ANALYSIS

In order to analyse the security of our proposed algorithm, we look at the message authenticy and message integrity as well as the confidentiality provided by the OTP.

5.1. Message Integrity And Authenticity

The message integrity of the one-time pad protocol is based on the security of the MAC. The MAC used in the simulations is a CBC-MAC implemented by TinySec. It uses a MAC of 4 bytes. Using a 4 byte MAC, there is a 1 in 2^{32} chance that an attacker is able to create a MAC and have it be an exact match. Overall, after about 2^{31} such attempts an attacker should have made a match. The assumption is that the MAC can only be verified by sending it to an authorized receiver. The attacker notes if the message is validated by the receiver. The attacker would have to send 2^{31} messages, on average. In traditional networks such a number is not much, however in sensor networks it will provide a level of security that is sufficient. Most radios channels in sensor networks operate at 19.2kb/s and it would take an attacker about 8 months to send 2^{31} messages of 23 bytes each. This exceeds the lifetime of most sensor networks. Additionally, our protocol uses a new key for each transmission, therefore an attacker cannot learn anything about future transmissions from the past.

5.2. Confidentiality

The security of the OTP depends on the security of the key k_{time}. Since a new key is generated at each transmission the security of the protocol depends on the security of the key derivation function (KDF) as described in the IEEE Standard Specifications for Public-Key Cryptography [6]. Under the assumption that the KDF is secure the problem reduces to the ability of an attacker to randomly create a one-time pad and have it match. The size of the one-time pad is equal to the size of the data to be protected, we assume 5 bytes throughout this work. That is equal to a 1 in 2^{64} chance of randomly having the right string or an average of 2^{32} tries until a correct one is found. At almost 16 months that is well over the lifetime of the typical sensor network.

6. ANALYTICAL ANALYSIS

To analyze the suitability of the algorithms for sensor networks, we consider cost of running the algorithm in terms of computational complexity and in terms of energy. We also show the correctness of the algorithms.

6.1. Cost Analysis

The embedding and the detection algorithms are very similar. The calculation of the MAC using an appropriate algorithm takes $O(b(x_i))$ time, where $b(x_i)$ is the bit size of the input, not the size of the network. This study uses the SkipJack algorithm in the analysis. All other lines have complexity $O(1)$. Thus, the running time of the complete embedding and detection algorithms is $O(b(x_i))$.

While the program space is of concern in sensor programming, this research is primarily concerned with the size of the memory necessary to perform the computation depending on the input size $b(x_i)$. The algorithm neither increases nor decreases the size of the measured value. The memory requirements for any of the variables used in the algorithm is rather small; however, the requirements of the SkipJack algorithm are large in comparison. The memory requirements of the SkipJack algorithm are independent of the size of the input, but because it has been implemented for TinyOS as part of the TinySec project, it is known that the size is small enough to work [8].

6.2. Correctness

Claim 6.1: The embedding algorithm 1 correctly embeds a one-time pad in the sensed value.

proof A sensor value x_i of $b(x_i) = 1$ is given. The calculation of the MAC in line 1 of Algorithm 1 results in a MAC of 4 bytes. The calculation of α and β on lines 3 and 4 of algorithm 1 are as follows. The calculation of α will result in a value of $[1, b(MAC))$. The calculation of β will result in a value of $[0, b(MAC) - \alpha + 1)$. Because α is subtracted from the bitsize of the MAC, it is confirmed that the substring $temp\_pad$ will in fact exist. Then $temp\_pad$ is repeated until $b(temp\_pad)$ = $b(x_i)$. Thus, the one-time pad $\widetilde{pad}$ is the same length as the data, and by XOR $\widetilde{pad}$ with the reading, the data are securely encrypted.

It is hypothesized that for all l less then m, such that $|x_i| = l$ it implies that the one-time pad used to secure the reading is of the same length as the reading. In other words, $\forall l, l \leq m$, $|x_i| = l \Rightarrow$ successful embedding of the one-time pad.

Now, it will be shown that $|x_i| = m$ has proper embedding $\Rightarrow$ $|x_i| = m+1$ has proper embedding. By adding an additional bit to x_i, the MAC calculated will be different, but the values of α and β still result in a value of $[1, b(MAC))$ and $[0, b(MAC) - \alpha + 1)$, respectively. The calculation of $temp\_pad$ on line 5 of algorithm 1 will still result in a proper substring of the MAC and thus a proper one-time pad, and hence the embedding is done properly.

Claim 6.2: The detection algorithm 2 detects the embedded one-time pad in a received value and restores the sensed value.

proof Again, the smallest permissible length of the values of x_i is one. A sensor value x_i with $|x_i| = 1$ is given. The calculations of α and β on lines 3 and 4 of Algorithm 2 using the MAC received as part of the packet result in the values of $[1, b(MAC))$ and $0, b(MAC) - \alpha + 1)$, respectively. If the MAC was not modified during transmission, then this guarantees that the same α and β will be calculated. Naturally, it also implies that the same $temp\_pad$ will be extracted. Because the size of the reading is fixed in the network, the same $\widetilde{pad}$ will be generated and thus can successfully be removed from the packet. The calculation of the MAC is over the same data as at the embedding location, and it will only result in a successful outcome if the data were not modified during transmission.

It is hypothesized that for all l less then m, such that $|x_i| = l$ it implies that the embedding can be found during detection and the original value can be restored. In other words, $\forall l, l \leq m$, $|x_i| = l \Rightarrow$ successful detection and removal of the one-time pad.

Next it will be shown that $|x_i| = m$ has proper detection $\Rightarrow$ $|x_i| = m + 1$ has proper detection. By adding an additional bit to x_i, the MAC calculated will be different, but the values of α and β will still result in a value of $[1, b(MAC))$ and $[0, b(MAC) - \alpha + 1)$, respectively. The calculation of $temp\_pad$ on line 5 of Algorithm 1 will still result in a proper substring of the MAC and thus a proper one-time pad. Thus, the detection is done properly.

6.3. Energy Use Analysis

Law, Doumen, and Hartel showed that while there are some difference in the number of clock cycles an instruction will take, the differences are not statistically significant [9]. Therefore, the computational complexity of an algorithm can be directly translated to energy consumption, assuming that the energy per CPU cycle is fixed. In [4] the authors showed that the SkipJack cipher uses 15925 cycles. The algorithms in the current research add an additional 20 instructions per algorithm. Because the average instruction takes 400 cycles, these algorithms add an additional 8000 cycles for a total of 23925 cycles for the application [11]. The MICA2 motes use 4 nJ per cycle [7]. Thus, the expected energy consumption for the embedding and detection algorithms is 95700 nJ. The average AA battery contains 1000 Joules. Because the MICA2 motes operates on two AA batteries, the energy consumption of both the embedding and detection algorithms is appropriate for sensor networks.

7. SIMULATION

PADS was simulated using the TinyOS operating system and its simulator TOSSIM [10]. It was compared to other protocols using a fixed topology for each simulation. In all simulations, every node in the network generates and sends a packet about every five seconds. The comparison included routing with non-secure AODV, with the PADS technique, and with TinySec [8]. Each simulation ran for a simulated 10 minutes, and for each network size the simulation was repeated five times. The average of the five simulations was used for the evaluation. For each set of simulations, five sizes of networks were compared: 15 nodes, 30 nodes, 45 nodes, 60 node and 75 nodes. Three different sets of simulations were performed. The first set compared same sized messages; all messages had a size of 23 bytes no matter the protocol. We ran simulations using the same size message to study the performance, as communication costs mainly depend on the size of the message. Since larger messages would require more energy to transmit any difference in performance can be attributed to sources other then message size. This allowed us to remove any doubt regarding what contribution the message size has on the performance. In order to achieve this, messages in non-secure AODV had payloads of seven bytes, PADS had payloads of three bytes and TinySec used payloads of two bytes. In the second set of simulations we are using the three protocols using messages with a two byte payload resulting in messages of 18 bytes, 22 bytes and 23 bytes for non-secure AODV, PADS and TinySec, respectively. The latency (average time it takes a packet to reach the base station), the throughput of bits per second (bps), and the average energy used per node were evaluated. We calculated the standard error for each measure. Two measures are considered to be statistically different if their error bars do not overlap.

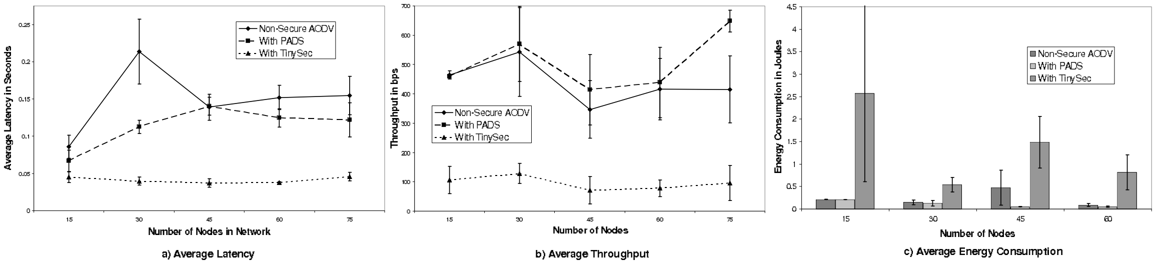
7.1. Comparison Of Protocols With Total Message Size Of 23 Bytes

In order to see the impact the protocol has on network performance, we simulated using the three algorithms (PADS, TinySec and non-secure AODV) while limiting messages to 23 bytes. In this set of experiment each message has 23 bytes, resulting in payloads of 2 bytes for TinySec, 3 bytes for PADS and 7 bytes for non-secure AODV. Any differences in performance can be attributed to the protocols as the size of the messages is the same.

7.1.1. Latency

It is important to know what delay creating and applying the one-time pad adds. Each packet was timestamped when sent in the application layer at the sending node and again when received in the application layer at node 0. The time it took for a packet to travel from the application layer in the sending node to the receiving point in the application layer was measured. The difference is the travel time of a packet.

Figure 7.1a) shows the average latency per packet for various network sizes. It is easy to see that the latencies for non-secure AODV and the PADS protocol are similar. The only statistically significant difference between PADS and non-secure AODV is at 30 nodes. One of the simulations for the non-secure network had congestion, resulting in several hundred messages with latencies well above 1 second, while the average for the rest of the simulations was closer to 0.14 seconds. This caused the average latency to be so high. However, it seems that TinySec has an average latency that is much better then either of the two other protocols. A closer look at the data revealed that TinySec, on average sends 16 times as many messages as are being received. Figure 7.2a) shows the ratio of Sent Message to Received Messages, the failure rate, for the three protocols. It is worthy to note, that the ratio is similar for PADS and for networks with non-secure AODV. TinySec, on the other hand quickly reaches ratios of more then five messages send for each message received. The reason is that TinySec, due to the lengthy encryption/decryption processes, at each node drops messages, especially at nodes which are utilized as relay nodes.



a) Average Latency

b) Average Throughput

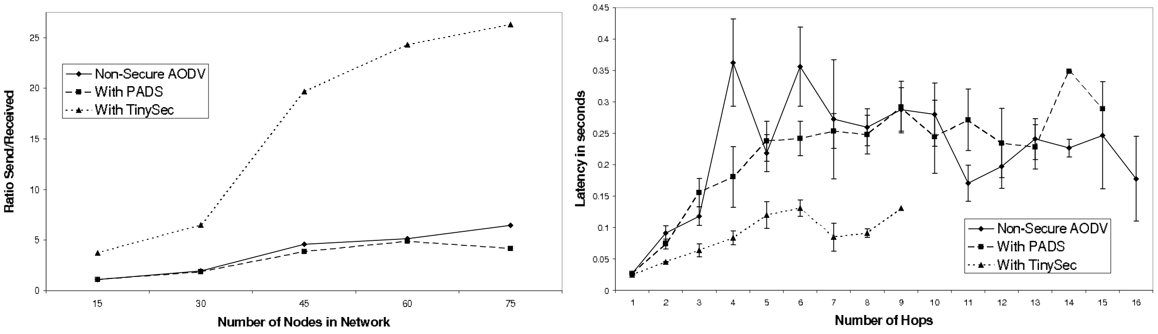
c) Average Energy Consumption

Fig. 7.1. Messages of 23 Bytes for Various Protocols

Another important metric is the average latency per total number of hops traveled. It is expected that the latency increases as a message has to travel multiple hops. Figure 7.2b) shows the average latency per hop. Note how the slop of the increase is similar for PADS and networks with non-secure AODV. At 4 and 6 hops, respectively, the simulations for non-secure AODV resulted in a lower number of received messages, which inflates the average latency for those hop counts. At 11 the network with non-secure AODV had very little congestion resulting in about 15% of the messages having average latencies below 0.1 seconds. This reduced the overall average latency for messages with that hop count. At 14 hops, PADS had a particularly congested network, resulting in a higher then expected average latency. TinySec seems to have an average per hop latency much better then either of the other protocols. Considering the low success rate of received messages as well as the fact that despite using a fixed topology in each simulation, there are no simulations using TinySec with hop counts greater than 9 hops, the overall performance of TinySec is much worse then PADS.

7.1.2. Throughput

Another important consideration in a sensor network is the throughput. Because the usefulness of sensor networks lies in independently sensing and sending large amounts of data, any technique must be able to sustain high data rates. Again, PADS is compared to



b) Ratio Sent/Received

b) Per Hop Latency

Fig. 7.2. Messages of 23 Bytes for Various Protocols

no security and TinySec encryption and authentication. As is evident from Figure 7.1b), the throughput of PADS is close to that of a network with non-secure AODV. The only noteworthy difference is for networks of 75 nodes. Of the 5 simulations, 2 had very high failure rates, resulting in only a few hundred received messages when almost 9000 were sent. This causes a lower than expected throughput. On the other hand, TinySec encryption and authentication results in greatly reduced throughput.

7.1.3. Energy Cost

The amount of energy available in sensor networks is very limited. Any sensor network protocol needs to be energy aware. Since the energy available to sensor networks is limited, the energy consumption of any application is critically important. PADS is compared to non-secure AODV and to TinySec using encryption and authentication. During simulation each network sends different amounts of messages. In order to have an accurate picture of the energy usage in a network and what the true cost of sending a data message is, we calculate the average amount of energy used per data message received. Figure 7.1c) shows the average amount of energy used per node per message sent for various network sizes. The difference for networks with 45 nodes is because the networks with non-secure AODV had high failure rates on average resulting in a high energy consumption per message received. It is important to note, that TinySec always uses significantly more energy then the other protocols no matter the network sizes. The reason for the much higher energy consumption per message for TinySec is that we are measuring the energy used per message received at the base station and the failure rate for TinySec is much higher than the other networks.

7.2. Comparison Of Protocols With 2 Byte Payload

In most applications of sensor networks, the payload remains stable and depending on other protocols used, such as AODV for multi-hop routing and TinySec for security, the total message size may vary. In this set of experiments, each message generated has a payload of 2 bytes, resulting in message sizes of 18 bytes for networks with non-secure AODV, 22 bytes for PADS and 23 bytes for TinySec.

7.2.1. Latency

The average latency for the three protocols is shown in Figure 7.3a). The average latency increases at a similar pace for the PADS protocol as well as for non-secure AODV. On average the PADS networks with 30 nodes experienced more congestion which lead to a higher latency. Again TinySec seems to outperform both the network with non-secure AODV as well as the proposed PADS protocol. However, when considering the low success rate for TinySec, performance is no longer adequate.

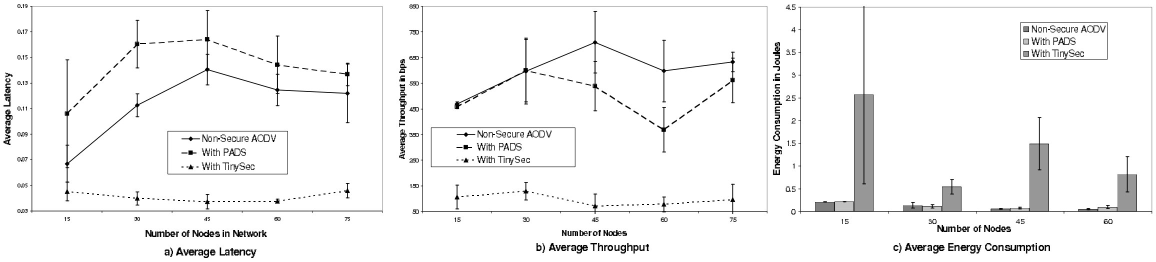


Fig. 7.3. Messages with 2 Byte Payload for Various Protocols

7.2.2. Throughput

Figure 7.3b) shows the comparison of the throughput in bps. As expected, the PADS protocol and non-secure AODV behave in a similar fashion. The only statistically significant difference between PADS and non-secure AODV is at 60 nodes. This is because two simulations produced less then 800 messages while the three other simulations produced more then 1800 messages. Hence the congestion decreased the average bps for PADS networks with 60 nodes. The performance of TinySec is much worse then either of the other two protocols since TinySec received many fewer messages then either of the other two protocols.

7.2.3. Energy Cost

The average energy consumption per message sent was measured and compared for different size networks for each of the three protocols where each message had a 2 byte payload. Figure 7.3c) shows the result of this simulation. TinySec's energy consumption per message is much higher then the other two protocols, this stems from the fact that in TinySec much fewer messages are received. There are no statistically significant differences between the PADS protocol and non-secure AODV.

8. CONCLUSION AND FUTURE WORK

This paper has shown that data in sensor networks can be securely routed through the network. By embedding a one-time pad to encrypt the payload of a message, the actual value transmitted is distorted so that any eavesdropper will not be able to use the information. Additionally, it is possible to ensure that the data were generated by a trusted source because the construction of the one-time pad will fail if any messages have been corrupted or were injected. The correctness of the two algorithms presented has been proven and an analysis of the time and space complexity required by the algorithms has been given. The simulations have shown that the additional work required by these algorithms is negligible and the performance rivals that of a network without security. The one-time pad protocol outperforms a network with encryption and authentication using TinySec. The throughput of this approach is as good as one without security and better then with Tinysec. Future work will include applying the one-time pad protocol to data aggregation and data fusion.

9. REFERENCES

[1] B. Deb, S. Bhatnagar, and B.i Nath. A topology discovery algorithm for sensor networks with applications to network management. DCS Technical Report DCS-TR-441, Rutgers University, May 2001.

[2] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *Mobile Computing and Networking*, pages 263–270, 1999.

[3] S. Ganeriwal, R. Kumar, and M. B. Srivastava. Timing-sync protocol for sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 138–149, New York, NY, USA, 2003. ACM Press.

[4] G. Guimaraes, E. Souto, D. Sadok, and J. Kelner. Evaluation of security mechanisms in wireless sensor networks. *icw*, 00:428–433, 2005.

[5] W. Rabiner Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 174–185. ACM Press, 1999.

[6] IEEE. IEEE standard 1363–2000. Standard Specifications for Public Key Cryptography, August 2000.

[7] Crossbow Technology Inc. MPR400/410/420 MICA2 Mote. Datasheet 2005.

[8] C. Karlof, N. Sastry, and D. Wagner. Tinysec: A link layer security architecture for wireless sensor networks. In *Second ACM Conference on Embedded Networked Sensor Systems (SensSys 2004)*, November 2004.

[9] Y. Law, J. Doumen, and P. Hartel. Benchmarking block ciphers for wireless sensor networks (extended abstract). In *1st Int. Conf. on Mobile Ad-hoc and Sensor Systems*, page electronic edition, Fort Lauderdale, Florida, Oct 2004. IEEE Computer Society Press, Los Alamitos, California.

[10] P. Levis. Tossim: A simulator for tinyos networks.

[11] P. Levis, D. Gay, and D. Culler. Active sensor networks. In *Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI '05)*, Boston, MA, USA, May 2005.

[12] P. Levis, N. Lee, M. Welsh, and D. Culler. Tossim: accurate and scalable simulation of entire tinyos applications. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 126–137, New York, NY, USA, 2003. ACM Press.

[13] D. Malan, M. Welsh, and M. Smith. A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography, 2004.

[14] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar. SPINS: Security protocols for sensor networks. In *Seventh Annual International Conference on Mobile Computing and Networks (MobiCOM 2001)*, Rome, Italy, July 2001.

[15] B. Przydatek, D. Song, and A. Perrig. Sia: secure information aggregation in sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 255–265. ACM Press, 2003.

[16] P. Von Rickenbach and R. Wattenhofer. Gathering correlated data in sensor networks. In *DIALM-POMC*, pages 60–66, October 2004.

[17] R. Sion, M. Atallah, and S Prabhakar. Resilient rights protection for sensor streams. In *VLDB*, pages 732–743, 2004.

[18] W. Stallings. *Cryptography and network security: principles and practice*. Pearson Education, third edition, 2002.

[19] H. Vogt. Exploring message authentication in sensor networks. In *Proceedings of ESAS 2004 (1st European Workshop on Security in Ad Hoc and Sensor Networks)*, LNCS, Heidelberg, Germany, August 2004. Springer-Verlag.

[20] H. Xu, L. Huang, Y. Wan, and B. Xu. Accurate time synchronization for wireless sensor networks. In Xiaohua Jia, Jie Wu, and Yanxiang He, editors, *MSN*, volume 3794 of *Lecture Notes in Computer Science*, pages 153–163. Springer, 2005.

[21] S. Zhu, S. Setia, S. Jajodia, and P. Ning. An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks. In *IEEE Symposium on Security and Privacy*, pages 259–271. IEEE Computer Society, 2004.

[22] S. Zhu, S. Setia, and S.l 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*, pages 62–72. ACM Press, 2003.

IV. SECURE HIERARCHICAL DATA AGGREGATION IN WIRELESS SENSOR NETWORKS

Julia Albath and Sanjay Madria
Missouri University of Science and Technolgy
Department of Computer Science
julia.albath@acm.org, madrias@mst.edu

Abstract—**Communication in wireless sensor networks uses the majority of a sensor's limited energy. Using aggregation in wireless sensor network reduces the overall communication cost. Security in wireless sensor networks entails many different challenges. Traditional end-to-end security is not suitable for use with in-network aggregation. A corrupted sensor has access to the data and can falsify results. Additively homomorphic encryption allows for aggregation of encrypted values, with the result being the same as the result when unencrypted data was aggregated. Using public key cryptography, digital signatures can be used to achieve integrity. We propose a new algorithm using homomorphic encryption and additive digital signatures to achieve confidentiality, integrity and availability for in-network aggregation in wireless sensor networks. We prove that our digital signature algorithm which is based on the Elliptic Curve Digital Signature Algorithm (ECDSA) is as secure as ECDSA.**

I. INTRODUCTION

Wireless sensor networks (WSN) are self-organizing networks of small, battery powered sensors used to monitor the environment for events such as forest fires, pollutant levels or enemy troop movements. A large number of small, battery powered computing devices with built-in radios are spread over the area to be monitored. Upon activation, these sensors self-organize into a multi-hop network, which connects to the users via a powerful base station in order to achieve a common goal [1]. As each sensor surveys the area within its sensing range, the data is sent towards the base station along a multi-hop path. A WSN

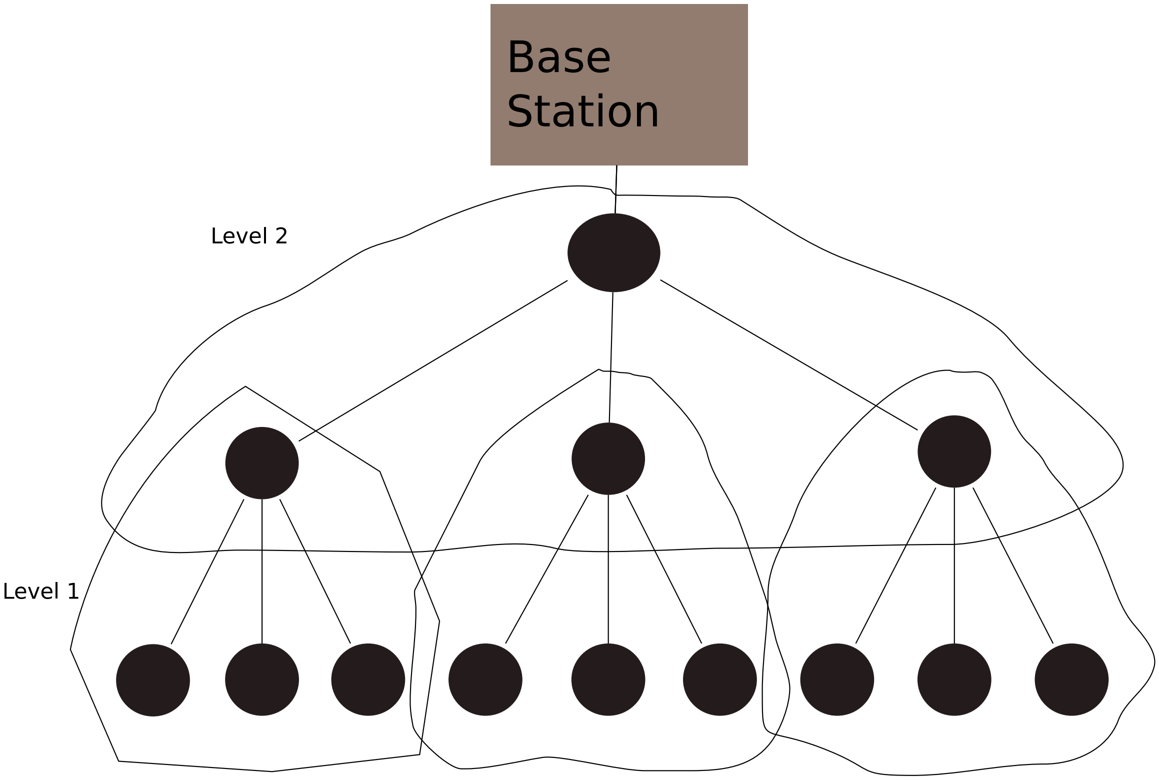


Fig. I.1. Motivating Example

is able to remotely cover a large sensing area since these low-cost sensors organize into a multi-hop network without human assistance.

Since sensors are typically battery powered and a WSN contains thousands of sensors, replacing the batteries is not a possibility. In terms of energy usage, communication is much more expensive than any internal computations [2]. In in-network aggregation, intermediate results are calculated along the multi-hop path whenever two or more messages are routed along the same path. Depending on the routing structure, energy savings may be by as much as eight times [3]. Consider the network depicted in Figure I.1. Without aggregation, a total of $9 + 12 + 13 = 34$ messages are sent; with aggregation, only $9 + 3 + 1 = 13$ messages are sent. Through the sample network in Figure I.1 it is clear that in-network aggregation greatly reduces the number of messages sent in a WSN, which leads to large energy savings.

Security in WSNs includes confidentiality, integrity and availability. Confidentiality in sensor networks is accomplished by preventing outsiders from eavesdropping on transmissions. This is generally achieved by encrypting the relevant parts of a packet. Integrity in general means that the receiver is assured that the packet was not tampered with or the message altered in some way. By ensuring availability we mean that the data is available in a timely fashion so that it is useful to the user. Availability in sensor networks is of great concern to the user of the network. Unfortunately, many existing security primitives cannot be used in sensor networks, either because the computing power of the sensors is too limited or the additional work created by the protocols causes excessive network traffic [4].

Sensors in the network can become corrupted due to the environment such as water, wind or sand acting on the sensor. In hostile environments, a sensor may deliberately be corrupted by an attacker. A corrupted sensor may appear to participate in the mission of the network but falsify sensor readings, improperly apply an aggregation function, exclude legitimate messages from the aggregate result or create a fictitious result. A sensor corrupted by an attacker may behave in this way in order to get the base station to accept an incorrect result that is favorable to the attacker. Hence in order to securely aggregate data in a sensor network, we must not only provide protection against eavesdroppers, but we should also prevent intermediate sensors from having access to the data.

Homomorphic encryption schemes are one possibility of ensuring secure aggregation, as they allow data aggregation to be performed on encrypted data. Encryption and decryption operations are computationally very expensive and time consuming. In link-layer cryptography [5], the data is encrypted by the sender, decrypted at intermediate nodes, the aggregation function is applied and the result is encrypted again before being sent to the next hop; this can lead to overflowing queues. In homomorphic encryption certain aggregation functions such as sum and average can be calculated on the encrypted data, reducing the workload of the sensors in the network significantly. The data is encrypted and sent toward the base station, while sensors along the path apply the aggregation function on the encrypted data. The base station receives the encrypted aggregate result and decrypts it. In section II we describe the scheme of homomorphic encryption in detail. Homomorphic encryption schemes provide security against eavesdroppers and protect the aggregate result from being known by intermediate, possibly corrupted sensors.

Integrity of the aggregate result can easily be achieved on a hop-by-hop basis in wireless sensor networks. Achieving end-to-end integrity while allowing for data aggregation provides us with new challenges. We need to clarify the meaning of integrity when data aggregation is applied. In aggregation integrity implies that any aggregate result is made up of only legitimate data without inclusions or additions, and that corrupted sensors cannot interfere with operations of the aggregation. We want to be able to assure the base station that the aggregate result it receives is a fair representation of the network state.

In this paper we introduce a novel way to provide confidential and integrity preserving aggregation in wireless sensor networks. In section III we propose the use of homomorphic encryption in WSN in order to achieve:

- A solution for confidentially calculating the SUM and AVERAGE in a wireless sensor network. Our algorithm is present in Section IV.
- A solution for integrity preserving data aggregation in wireless sensor networks. We are using an additively digital signature algorithm based on ECDSA to achieve integrity of the aggregate result. We provide security analysis of the algorithm in Section V.

II. BACKGROUND

In sensor networks, data aggregation provides energy savings. The lifetime of most networks is limited and it is important for protocols to be energy-efficient. Combining multiple payloads into one message or combining data by allowing for in-network calculation of aggregates leads to these energy savings [6].

In many proposed applications of wireless sensor networks, the networks generate large amounts of data in a continuous stream, making it difficult for a user to sift useful information from these masses of data [7]. Sensors may generate a reading of their environment every three seconds until the mission is completed; for a 100 sensor network, this would generate 300 messages every three seconds, or 6000 messages every minute. In-network aggregation can take this amount of data and combine it into one or more aggregated results every minute.

Calculating aggregate results such as SUM or AVERAGE is of special interest to sensor networks. Wireless sensor networks are designed to provide large amounts of data, which is a snapshot of the environment at one point in time. Combining several readings by calculating the AVERAGE or the SUM increases the accuracy of these readings [8].

Security in sensor networks requires new approaches due to the limitations of sensors and their limited computing power. Since a sensor network uses radio as the communication medium, all communications are inherently insecure. Anybody tuned to the same channel is able to eavesdrop on the transmissions. Many sensor network applications demand secure communications. Encryption is the preferred way to provide for a secure communication channel. Encryption ensures that only the sender and the intended receiver can read the message contents [9]. Traditional link-layer cryptography is an important part of an overall security strategy for sensor networks.

TinySec [5], an implementation of link-layer cryptography for TinyOS, has two operational modes: hop-by-hop (HBH) and end-to-end (ETE). ETE provides total security, as only the sender and the receiving base station are able to know the content of the message. Unfortunately, this also means that aggregation is not possible, because the intermediate nodes cannot access the payload. TinySec uses the SkipJack cipher, which does not allow for calculating of aggregate functions such as SUM or AVERAGE on encrypted data. When TinySec is used in HBH mode instead, the message is decrypted at each hop, and aggregation is possible. Due to the amount of time required for the encryption and decryption operations, the queues in a WSN can overflow, leading to dropped packets. Other drawbacks are that TinySec is based on private key cryptography, which leads to problems such as key distribution, key management, and that digital signatures are not possible [10].

In private key cryptography, both parties use the same key. Deciding when and how two sensors agree on which key to use is a big challenge. A private key cryptography approach also means that a sensor needs to store one key for every other sensor it wishes to communicate with. In WSN, the topology of the network can change, and protocols need to be flexible enough to allow for two previously unassociated sensors to begin communicating securely. One possible solution is a network-wide key, but the obvious problem with this approach is that only one corrupted sensor is required to compromise communications in the entire network.

Since all parties in private key cryptography use the same key, digital signatures are not feasible. In order to achieve integrity Message Authentication Codes (MACs) are used.

MACs are used to prove that the message has not been tampered with. Since sender and receiver share the same key, it cannot be proven who sent the message. A public key cryptography approach addresses many of these problems [11]. Public key cryptography allows for the application of digital signatures. Digital signatures provide integrity and repudiation. Only the party in possession of the private key can create a particular signature. When a message with a signature is received, the corresponding public key is used to verify the signature. Once the signature is verified, the receiver can be certain that the integrity of the message has not been breached. The receiver is also certain that only the sender in possession of the private key could have created that signature.

Homomorphic encryption is a cryptographic technique which allows calculations to be performed on aggregate data. Specifically, a homomorphic encryption scheme allows the following property to hold:

$$enc(a \oplus b) = enc(a) \oplus enc(b).$$

This means that in order to calculate the SUM of two values, we can apply some function to their encrypted counterparts and then decrypt the result of the SUM operation. Clearly, considering the cost of encryption and decryption, homomorphic encryption is useful in wireless sensor networks, because homomorphic encryption would allow for the calculation of SUM and AVERAGE on encrypted data. The data would be encrypted at the sensor node, the SUM or AVERAGE would be calculated as the aggregate result follows a path to the base station, and the final result would be decrypted at the base station. Any eavesdropper would be unable to gather information from the transmissions. Any corrupted sensor could not know the aggregate result. An example of an homomorphic cryptography scheme is the elliptic curve ElGamal system [12]. The EC ElGamal system is additively homomorphic because the following property holds:

$$enc(a + b) = enc(a) + enc(b).$$

Elliptic curve cryptography (ECC) employs the points on an elliptic curve over a finite field K. The required algorithms for elliptic curve cryptography can easily be implemented, even on small devices such as sensors [13]. Elliptic curve cryptography uses the analog of the discrete logarithm problem (DLP), also known as the elliptic curve discrete logarithm problem (EC-DLP). The DLP over elliptic curves is believed to computationally much more difficult then DLP over finite fields of the same size [10].

Homomorphic encryption does not provide integrity. Since we are using public key elliptic curve cryptography we will use digital signatures to provide integrity. Digital signature schemes are not homomorphic. That is two signatures generated on two different messages cannot be combined to verify the sum of the messages. We propose the use of an encryption scheme which will allow for homomorphic signature generation and verification.

III. APPROACH

We propose to use the EC-ElGamal system for homomorphic encryption in wireless sensor networks. In [11] the authors evaluated several public key homomorphic encryption schemes for use on sensors. The authors in [14] use an implementation of the EC-ElGamal algorithms for homomorphic encryption and storage in sensor networks. The EC-ElGamal system is thus clearly suitable for wireless sensor networks. Instead of applying EC-EG (EC-ElGamal) for persistent data storage, we propose to use it for homomorphic data encryption during transmission. The work in [11] provides for data confidentiality only. We propose the use of elliptic curve digital signatures to provide message integrity and integrity of the aggregate in addition to data confidentiality.

We will now provide an explanation of the example in Figure III.1. Each node generates a reading. The reading is signed with the aggregate signature protocol using the node's private key; this is shown as $Sig(x)$. Each node homomorphically encrypts the reading with the base stations' public key; this is shown as $Enc(x)$ in Figure III.1. The node sends the secured reading, the signature and its public key to its parent. After receiving messages from all its children, the parent combines the messages into one. The parent sums the secured readings, the signatures and the public keys. If the parent also contributes a reading, that reading is treated like any other reading. These are shown as $SUM - ENC$, $SUM - SIG$ and $SUM - KEY$ in Figure III.1. This is process is repeated by each parent along the path to the base station.

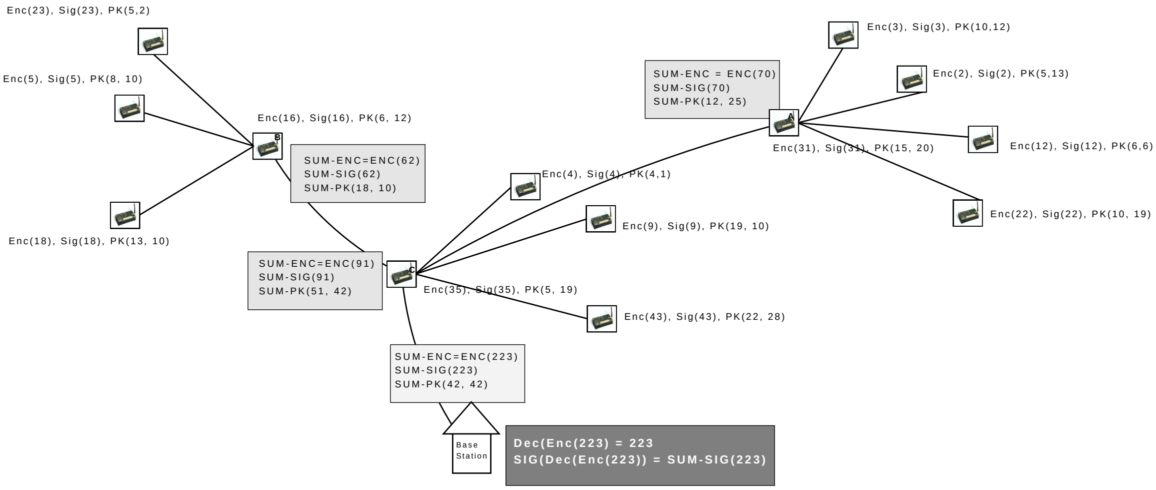

Enc(23), Sig(23), PK(5,2)
Enc(5), Sig(5), PK(8, 10)
Enc(16), Sig(16), PK(6, 12)
B
SUM-ENC=ENC(62)
SUM-SIG(62)
SUM-PK(18, 10)
Enc(18), Sig(18), PK(13, 10)
SUM-ENC=ENC(91)
SUM-SIG(91)
SUM-PK(51, 42)
C
Enc(35), Sig(35), PK(5, 19)
SUM-ENC=ENC(223)
SUM-SIG(223)
SUM-PK(42, 42)
Base Station
Dec(Enc(223) = 223
SIG(Dec(Enc(223)) = SUM-SIG(223)
Enc(4), Sig(4), PK(4,1)
Enc(9), Sig(9), PK(19, 10)
Enc(43), Sig(43), PK(22, 28)
SUM-ENC = ENC(70)
SUM-SIG(70)
SUM-PK(12, 25)
A
Enc(31), Sig(31), PK(15, 20)
Enc(3), Sig(3), PK(10,12)
Enc(2), Sig(2), PK(5,13)
Enc(12), Sig(12), PK(6,6)
Enc(22), Sig(22), PK(10, 19)

Fig. III.1. Homomorphic Encryption Example

The base station decrypts the received message. The sum of the readings was homomorphically encrypted with the base stations public key. This allows the base station to decrypt the readings. Only the base station which is in possession of the matching private key is able to decrypt the readings. This is shown as $Dec(Enc(x))$ in the figure. Each node signed its messages, and these signatures were combined along the way. The base station can now verify the sum of the signatures given the sum of the public keys. The aggregate signature protocol ensures that only readings from legitimate sensors are included in the aggregate.

IV. ALGORITHMIC DETAILS

We first describe the details for the algorithm executed at the sensors. Each sensor is pre-loaded with the appropriate elliptic curve parameters, the base stations' public key and a network wide random integer. The integer is used to generate a new k at set intervals. This ensures that the signatures are additive and secure against attacks. At the start of each round, each sensor choses a private key and computes the appropriate public key. Choosing a private key is straightforward and requires the sensor to pick an integer in the field of the elliptic curve. The public key is generated by multiplying the base point T with the private key; the result is another point on the curve. A new public/private key pair is necessary during each round of processing because it would only take two signatures for a malicious node to determine another node's private key. If a sensor signs the same reading with the same key, another sensor would be able to determine the private key. In most sensor applications, it's likely that a sensor would generate the same message several times. Each sensor computes R, which is the base point T multiplied by the current random integer k. Additionally, each sensor computes the multiplicative inverse of $k \bmod p$. Each sensor can now generate its unique signature s_i. After the signature has been generated, the sensor proceeds to homomorphically encrypt its reading x_i. The sensor first maps its reading onto the elliptic curve. After the mapping the reading is encrypted using the EC-IES algorithm [15].

If the sensor receives messages from other nodes for forwarding, it combines them according to the algorithm. The signature scheme is designed such that all signatures can be combined via simple arithmetic. This makes the amount of work required from a parent very small and thus well suited for wireless sensor networks.

Algorithm 1 Sensor Algorithm

```
Require: Elliptic Curve Parameters D = (q, FR, a, b, T, p , h), sensor reading $m_i$, private
    key $z_i$, base station public key $Q$, a network wide random integer $k$
Each sensor computes $z_i * T = (x, y)$, its public key.
Each sensor computes $R = (r(x), r(y)) = k * T$.
Each sensor computes $k^{-1} \bmod p$.
Each sensor computes $s_i = k^{-1}(m_i + z_i * r(x)) \bmod p$.
Each sensor's signature for the message $m_i$ is $s_i$.
Each sensor maps its reading $m_i$ onto the elliptic curve $D$.
Each sensor generates ciphertext $\overline{m_i} = enc(m_i)$
if Sensor is a parent then
    The sensor combines the signatures into $s = \sum s_i)$
    The sensor combines all ciphertexts into one ciphertext $\sum m_i$
end if
```

We will now describe the base station's algorithm. The base station receives the sum of the signatures, the sum of the appropriate public keys and the homomorphically encrypted aggregate result. The base station can now verify that the same sensors that contributed to the aggregate also signed their inputs and that signature is included in the combined signature. The base station first decrypts the aggregate result using its private key. Additionally, the base station needs to reverse the mapping from the point on the elliptic curve to the aggregate result. To verify the signature, the base station calculates a point on the curve using the received signature, the decrypted aggregate result and the integer k. If the x-coordinate of the point calculated is the same as r(x), the signature is verified. The base station is now assured that no data not generated by a legitimate sensor was included in the aggregate.

Algorithm 2 Base Station Algorithm

```
Require: Elliptic Curve Parameters D = (q, FR, a, b, T, p , h), sum of encrypted sensor readings $m = \sum \overline{m_i}$, sum of the signatures $s = \sum s_i$, base station private key $q_i$, sum of public keys $Z$, a network wide random integer $k$
Decrypt ciphertext $\sum \overline{m_i} = \sum m_i$
Map reading $m$ from the elliptic curve $D$ into plaintext.
Compute $R = (r(x), r(y)) = k * T$.
Compute $w = s^{-1} \bmod p$.
Compute $u_1 = mw \bmod p$.
Compute $u_2 = r(x)w \bmod p$.
Compute $X = u_1 T + u_2 Z$.
Compute $v = X(x) \bmod p$.
if $v == r$ then
    The signature verified
end if
```

The algorithm described securely calculates the SUM of the readings in a wireless sensor network. In order to securely calculate the AVERAGE in a wireless sensor network, the base station needs a count of the number of points included in the SUM. With the knowledge of how many sensors contributed to the aggregate, the AVERAGE can be calculated.

V. SECURITY ANALYSIS

Our signature algorithm is an extension of the ECDSA. ECDSA is assumed to be secure. ECDSA has been shown to be secure under the assumption that the underlying group is generic and that a collision resistant hash function has been used.

Theorem 5.1: 1 The signature produced by summing the individual signatures will only verify if the contributing individual signatures were produced by a valid node and the appropriate public key was included in the sum of public keys.

We will now prove that the combined signature will only verify if the individual signatures contributed are signatures generated by valid nodes and are valid signatures. The value k is a randomized, synchronized integer used by all nodes in the network. We do not need to send $r(x)$ with each signature, as the base station is able to compute $r(x)$. Therefore the unique part of each node's signature is s_i.

Lemma 5.2: The sum of the public keys equals the public key generated from the sum of the private keys.

Proof: We have the sum of the public keys, $Z = Z_A + Z_B + \ldots$. Let's say that the sum of the private keys, $z = z_A + z_B + \ldots$. Since $Z_A = z_A * T, Z_B = z_B * T, \ldots$ we know that

$$\begin{aligned} Z &= z_A * T + z_B * T + \ldots \\ &\equiv (z_A + z_B + \ldots) * T. \end{aligned}$$

Therefore $Z = zT$. In other words, the sum of the public keys equals the public key generated from the sum of the private keys. □

Lemma 5.3: The sum of signatures produced creates a valid signature equivalent to a signature produced using the sum of the private keys on the sum of the messages.

Proof: We know that $m = m_A + m_B + \ldots$ and $s = s_A + s_B + \ldots$, as per Algorithm 1. Then

$$\begin{aligned} s &= k^{-1}(m_A + z_A r(x)) + k^{-1}(m_B + z_B r(x)) + \ldots \\ &\equiv k^{-1}((m_A + z_A r(x)) + (m_B + z_B r(x)) + \ldots) \\ &\equiv k^{-1}((m_A + m_B + \ldots) + (z_A + z_B + \ldots) r(x)) \\ &\equiv k^{-1}(m + (z_A + z_B + \ldots) r(x)). \end{aligned}$$

Using Lemma 5.2, it follows that $s = k^{-1}(m + zr(x))$. □

We can now prove Theorem 5.1, that the signature verification of $v == r$ will only work if each signer contributed the signature and the matching public key to the aggregate.

Proof: From Lemma 5.3, we know that $s = k^{-1}(m + zr(x))$. Rearranging we get

$$\begin{aligned} k &\equiv s^{-1}(m + zr(x)) \\ &\equiv s^{-1}m + s^{-1}r(x)z \\ &\equiv wm + wr(x)z \\ &\equiv u_1 + u_2 d \mod p. \end{aligned}$$

We also know that $X = u_1T + u_2Z$ and that $Z = zT$; it follows that

$$\begin{aligned} X &= u_1T + u_2Z \\ &\equiv u_1T + u_2(zT) \\ &\equiv (u_1 + u_2 z)T \end{aligned}$$

Thus $(u_1 + u_2z)T \equiv kT$ and $r == v$ as required for the signature verification. □

VI. RELATED WORK

Secure data aggregation schemes have been of interest to researchers. The earliest approaches focused on confidentiality of the data against a single aggregators. Algorithms which prevented or detected multiple aggregators colluding to deceive the base station were also introduced.

A new protocol for provably secure data aggregation in wireless sensor networks was proposed in [16]. The algorithm guarantees the detection of aggregate modification by the aggregator, except for those cases where the aggregator injects data into the aggregate. The algorithm supports any arbitrary tree structure and is resilient to any number of malicious nodes. The algorithm focuses on the use of the SUM operator, but would also work with MEDIAN, COUNT and AVERAGE. The algorithm forces a commitment from the adversary at intermediate nodes. Each sensor also verifies that its data was properly added to the aggregate. Our algorithm works with any single-path routing protocol, and will securely calculate the SUM and AVERAGE. Compared to the algorithm in [16], our proposed algorithm provides for a greater reduction in energy savings due to a reduced number of messages sent.

The algorithms introduced in [17] achieve concealed data aggregation. Concealment means that the data and the aggregates are not readable for anyone who is not in possession of the proper key. The algorithm uses privacy homomorphism to achieve data hiding while still allowing for data aggregation. The algorithm provides for data confidentiality only; the authors refer to other papers regarding solutions that provide data integrity and authenticity. The algorithm uses symmetric keys, while our work uses a private/public key approach.

The protocol introduced in [14] is not meant to provide data authentication and message integrity during transmission. Rather it is meant to provide persistent, secure data storage in sensor networks. It provides for secure data replication to ensure data availability in case of node failure. It also introduces secure data aggregation due to restricted storage space. Our work uses a similar public/private key homomorphic encryption protocol to ensure secure data aggregation during transmission.

The work in [11] provides a survey of possible homomorphic public key encryption schemes suitable for wireless sensor networks. The authors provide a list of desirable properties of a homomorphic public key encryption scheme for wireless sensor networks and evaluate the various candidates based on that list. The authors conclude that EC-OU (Elliptic Curve Okamoto-Uchiyama) and EC-EG (Elliptic Curve ElGamal) are the two algorithms most suitable for use as homomorphic public key encryption schemes. For our implementation we have chosen EC-IES, a variant of EC-EG.

For the work in [18], the authors propose a new additively homomorphic stream cipher suitable for wireless sensor networks. The proposed algorithms use a symmetric key stream. In addition to the issues related to symmetric key use, the base station needs to know exactly which sensor did or did not contribute data to the aggregate. Without this information the base station will be unable to decrypt the result. Only data confidentiality is provided with this algorithm. Our algorithm provides both data confidentiality and message integrity without the limitations of symmetric key cryptography.

As far as we can determine, this is the first work which used additively digital signatures in wireless sensor networks.

VII. CONCLUSION AND FUTURE WORK

In this paper a novel algorithm is presented to address the problem of secure data aggregation in wireless sensor networks. We apply a homomorphic encryption algorithm to the messages to achieve data confidentiality while allowing in-network aggregation. An additively digital signature algorithm based on ECDSA is used to achieve integrity of the aggregate. We showed that the signature algorithm is as secure as ECDSA. Future work will include implementing this algorithm in TinyOS/TOSSIM [19].

REFERENCES

[1] L. Clare, G. Pottie, and J. R. Agre, “Self-organizing distributed sensor networks,” *SPIE-The International Society for Optical Engineering*, pp. 229–237, 1999.

[2] R. W. Heinzelman, J. Kulik, and H. Balakrishnan, “Adaptive protocols for information dissemination in wireless sensor networks,” in *MobiCom ’99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*. ACM Press, 1999, pp. 174–185.

[3] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-efficient communication protocol for wireless microsensor networks,” in *HICSS ’00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8*. Washington, DC, USA: IEEE Computer Society, 2000, p. 8020.

[4] J. Albath and S. Madria, "Practical algorithm for data security (pads) in wireless sensor networks," in *MobiDE '07: Proceedings of the 6th ACM international workshop on Data engineering for wireless and mobile access*. New York, NY, USA: ACM Press, 2007, pp. 9–16.

[5] C. Karlof, N. Sastry, and D. Wagner, "Tinysec: A link layer security architecture for wireless sensor networks," in *Second ACM Conference on Embedded Networked Sensor Systems (SensSys 2004)*, November 2004.

[6] B. Krishnamachari, D. Estrin, and S. B. Wicker, "The impact of data aggregation in wireless sensor networks," in *ICDCSW '02: Proceedings of the 22nd International Conference on Distributed Computing Systems*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 575–578.

[7] K. Kalpakis, K. Dasgupta, and P. Namjoshi, "Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks," *Comput. Networks*, vol. 42, no. 6, pp. 697–716, 2003.

[8] C. Y. Chong, S. P. Kumar, and B. A. Hamilton, "Sensor networks: evolution, opportunities, and challenges," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1247–1256, 2003.

[9] M. Anand, Z. Ives, and I. Lee, "Quantifying eavesdropping vulnerability in sensor networks," in *DMSN '05: Proceedings of the 2nd international workshop on Data management for sensor networks*. New York, NY, USA: ACM Press, 2005, pp. 3–9.

[10] A. J. Menezes, *Elliptic Curve Public Key Cryptosystems*. Norwell, MA, USA: Kluwer Academic Publishers, 1994.

[11] E. Mykletun, J. Girao, and D. Westhoff, "Public Key Based Cryptoschemes for Data Concealment in Wireless Sensor Networks," *IEEE International Conference on Communications ICC*, 2006.

[12] J. M. Adler, W. Dai, R. L. Green, and C. A. Neff, "Computational Details of the VoteHere Homomorphic Election System," 2000.

[13] D. J. Malan, M. Welsh, and M. D. Smith, “A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography,” *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*, pp. 71–80, 2004.

[14] J. Girao, D. Westhoff, E. Mykletun, and T. Araki, “Tinypeds: Tiny persistent encrypted data storage in asynchronous wireless sensor networks,” *Ad Hoc Networks*, vol. 5, no. 7, pp. 1073–1089, September 2007.

[15] A. Liu and P. Ning, “Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks,” in *7th International Conference on Information Processing in Sensor Networks (IPSN 2008)*, April 2008, pp. 245–256.

[16] H. Chan, A. Perrig, and D. Song, “Secure hierarchical in-network aggregation in sensor networks,” in *CCS ’06: Proceedings of the 13th ACM conference on Computer and communications security*. New York, NY, USA: ACM Press, 2006, pp. 278–287.

[17] D. Westhoff, J. Girao, and M. Acharya, “Concealed data aggregation for reverse multicast traffic in sensor networks: Encryption, key distribution, and routing adaptation,” *IEEE Transactions on Mobile Computing*, vol. 5, no. 10, pp. 1417–1431, 2006.

[18] C. Castelluccia, E. Mykletun, and G. Tsudik, “Efficient aggregation of encrypted data in wireless sensor networks,” in *The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, July 2005, pp. 109–117.

[19] P. Levis, N. Lee, M. Welsh, and D. Culler, “Tossim: accurate and scalable simulation of entire tinyos applications,” in *SenSys ’03: Proceedings of the 1st international conference on Embedded networked sensor systems*. New York, NY, USA: ACM Press, 2003, pp. 126–137.

BIBLIOGRAPHY

[1] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*. New York, NY, USA: ACM Press, 2002, pp. 88–97.

[2] W. R. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*. ACM Press, 1999, pp. 174–185.

[3] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8*. Washington, DC, USA: IEEE Computer Society, 2000, p. 8020.

[4] B. Deb, S. Bhatnagar, and B. Nath, "A topology discovery algorithm for sensor networks with applications to network management," in *In IEEE CAS workshop, September 2002*, 2002.

[5] Y. Chen and A. Liestman, "Approximating minimum size weakly-connected dominating sets for clustering 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 Press, 2002, pp. 165–172.

[6] B. Das and V. Bharghavan, "Routing in ad-hoc networks using minimum connected dominating sets," in *ICC (1)*, 1997, pp. 376–380.

[7] S. Guha and S. Khuller, "Approximation algorithms for connected dominating sets," in *ESA '96: Proceedings of the Fourth Annual European Symposium on Algorithms*. London, UK: Springer-Verlag, 1996, pp. 179–193.

[8] D. Dubhashi, A. Mei, A. Panconesi, J. Radhakrishnan, and A. Srinivasan, "Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons," in *SODA '03: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2003, pp. 717–724.

[9] M. Chen and J. Wu, "EECS: an energy efficient clustering scheme in wireless sensor networks," *Performance, Computing, and Communications Conference, 2005. IPCCC 2005. 24th IEEE International*, pp. 535–540, 2005.

[10] F. Kuhn and R. Wattenhofer, "Constant-time distributed dominating set approximation," in *In Proc. of the 22nd ACM Symposium on the Principles of Distributed Computing*, 2003.

[11] F. Grandoni, J. Knemann, A. Panconesi, and M. Sozio, “Primal-dual based distributed algorithms for vertex cover with semi-hard capacities,” in *PODC ’05: Proceedings of the twenty-fourth annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*. New York, NY, USA: ACM Press, 2005, pp. 118–125.

[12] L. Jia, R. Rajaraman, and T. Suel, “An efficient distributed algorithm for constructing small dominating sets,” *Distrib. Comput.*, vol. 15, no. 4, pp. 193–205, 2002.

[13] A. Amis, R. Prakash, T. Vuong, and D. Huynh, “Max-min d-cluster formation in wireless ad hoc networks,” in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1, 2000, pp. 32–41 vol.1.

[14] Y. Fernandess and D. Malkhi, “K-clustering in wireless ad hoc networks,” in *In Proceedings of the second ACM international workshop on Principles of mobile computing*, 2002, pp. 31–37.

[15] S. Kutten and D. Peleg, “Fast distributed construction of k-dominating sets and applications,” in *PODC ’95: Proceedings of the fourteenth annual ACM symposium on Principles of distributed computing*. ACM Press, 1995, pp. 238–251.

[16] C. Karlof, N. Sastry, and D. Wagner, “Tinysec: A link layer security architecture for wireless sensor networks,” in *Second ACM Conference on Embedded Networked Sensor Systems (SensSys 2004)*, November 2004.

[17] B. Przydatek, D. Song, and A. Perrig, “Sia: secure information aggregation in sensor networks,” in *SenSys ’03: Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM Press, 2003, pp. 255–265.

[18] A. Mahimkar and T. Rappaport, “Securedav: a secure data aggregation and verification protocol for sensor networks,” in *Global Telecommunications Conference, 2004. GLOBECOM ’04. IEEE*, vol. 4, 2004, pp. 2175–2179 Vol.4.

[19] L. Hu and D. Evans, “Secure aggregation for wireless networks,” in *SAINT-W ’03: Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT’03 Workshops)*. Washington, DC, USA: IEEE Computer Society, 2003, p. 384.

[20] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar, “SPINS: Security protocols for sensor networks,” in *Seventh Annual International Conference on Mobile Computing and Networks (MobiCOM 2001)*, Rome, Italy, July 2001.

[21] H. Chan, A. Perrig, and D. Song, “Secure hierarchical in-network aggregation in sensor networks,” in *CCS ’06: Proceedings of the 13th ACM conference on Computer and communications security*. New York, NY, USA: ACM Press, 2006, pp. 278–287.

[22] S. Lee, C. Kim, and S. Kim, “Constructing energy efficient wireless sensor networks by variable transmission energy level control,” in *Computer and Information Technology, 2006. CIT ’06. The Sixth IEEE International Conference on*, 2006, pp. 225–225.

[23] D. Wei and A. H. Chan, “Clustering algorithm to balance and to reduce power consumptions for homogeneous sensor networks,” in *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*, 2007, pp. 2723–2726.

[24] R. Krishnan and D. Starobinski, “Efficient clustering algorithms for self-organizing wireless sensor networks,” *Ad Hoc Networks*, vol. 4, no. 1, pp. 36–59, January 2006.

[25] C. Duan and H. Fan, “A distributed energy balance clustering protocol for heterogeneous wireless sensor networks,” in *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*, 2007, pp. 2469–2473.

[26] O. Younis and S. Fahmy, “HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks,” *Mobile Computing, IEEE Transactions on*, vol. 3, no. 4, pp. 366–379, 2004.

[27] R. Karp, “Reducibility among combinatorial problems,” in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds. Plenum Press, 1972, pp. 85–103.

[28] F. Kuhn, T. Moscibroda, and R. Wattenhofer, “Unit disk graph approximation,” in *DIALM-POMC ’04: Proceedings of the 2004 joint workshop on Foundations of mobile computing*. ACM Press, 2004, pp. 17–23.

[29] J. Wu and H. Li, “Domination and its applications in ad hoc wireless networks with unidirectional links,” in *ICPP ’00: Proceedings of the Proceedings of the 2000 International Conference on Parallel Processing*. Washington, DC, USA: IEEE Computer Society, 2000, p. 189.

[30] V. Mhatre and C. Rosenberg, “Design guidelines for wireless sensor networks: communication, clustering and aggregation,” *Ad Hoc Networks*, vol. 2, no. 1, pp. 45–63, 2004.

[31] M. Younis, M. Youssef, and K. Arisha, “Energy-aware routing in cluster-based sensor networks,” in *MASCOTS ’02: Proceedings of the 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS’02)*. Washington, DC, USA: IEEE Computer Society, 2002, p. 129.

[32] S. Banerjee and S. Khuller, “A clustering scheme for hierarchical control in multi-hop wireless networks,” in *IEEE INFOCOM*, April 2001, pp. 1028–1037.

[33] T. T. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to algorithms*. Cambridge, MA, USA: MIT Press, 1990.

[34] B. Liang and Z. Haas, "Virtual backbone generation and maintenance in ad hoc network mobility management," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, 2000, pp. 1293–1302 vol.3.

[35] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: accurate and scalable simulation of entire tinyos applications," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*. New York, NY, USA: ACM Press, 2003, pp. 126–137.

[36] O. Younis and S. Fahmy, "An experimental study of routing and data aggregation in sensor networks," *Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on*, pp. 8 pp.–, Nov. 2005.

[37] A. Boukerche, X. Fei, and R. B. Araujo, "An optimal coverage-preserving scheme for wireless sensor networks based on local information exchange," *Computer Communications*, vol. 30, no. 14-15, pp. 2708–2720, October 2007.

[38] G. Guimaraes, E. Souto, D. Sadok, and J. Kelner, "Evaluation of security mechanisms in wireless sensor networks," *icw*, vol. 00, pp. 428–433, 2005.

[39] C. T. Inc., mPR400/410/420 MICA2 Mote. Datasheet 2005.

[40] B. Krishnamachari, D. Estrin, and S. B. Wicker, "The impact of data aggregation in wireless sensor networks," in *ICDCSW '02: Proceedings of the 22nd International Conference on Distributed Computing Systems*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 575–578.

[41] C. Castelluccia, E. Mykletun, and G. Tsudik, "Efficient aggregation of encrypted data in wireless sensor networks," in *The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, July 2005, pp. 109–117.

VITA

Julia G. M. Albath was born in Bochum, Germany. In May 2003, she received her B.S. in Mathematics from Illinois State University, Normal, Illinois, USA. She worked for State Farm Insurance Cos. for four years starting in 1999. In 2003 she left and started her graduate studies at the Missouri University of Science and Technology, Rolla, Missouri, USA. In December 2008 she received her Ph.D. in Computer Science from the Missouri University of Science and Technology, Rolla, Missouri, USA.

She has published conference papers, some of which are listed with the references of this research.

Julia G. M. Albath has been a member of the Institute of Electrical and Electronics Engineers (IEEE) since 2003. She has been a member of the Association for Computing Machinery since 2003.