Measures for maximal utilization of a secure Wireless Sensor Network

Deepthi Dattatray Kulkarni
Dept. of Computer Science,
Santa Clara University
ddkulkarni@scu.edu

Anisha Agarwal
Dept. of Computer Science,
Santa Clara University
anagarwal@scu.edu

Contents

Preface

We come across a lot of scenarios where we wish to monitor every occurrence of a particular event remotely. For example, consider a battlefield wherein the footsteps of the enemy contingent is to be tracked. Or, lets take a more humane scenario of a forest wherein a team wants to analyze the movement patterns of a particular herd of animals for research purposes. In such a scenario, all they need to do is spread out sensors around the area to be tracked and monitor the signals from these sensors. The network formed by all these numerous sensors is called a Wireless Sensor Network.

Our project concentrates on extending the lifetime of a sensor network. In the scenario of the forest mentioned previously, consider you would want to track the sounds of a particular animal for a particular period of time. And after sometime, you want to increase the duration of sensing as your experiment is not yet complete. Our project handles exactly the same kind of situations. Another example would be about a scenario where the a sensor network is to be deployed such that only a part of the network has to be active initially, and later the rest of them should get activated. For example, take a scenario where in an intruder is trying to jam the signals of the sensor nodes. In order to bypass him and communication continue, what can be done is deploy several nodes and activate only a part of them which transmit at a particular frequency. In case the intruder is successful in jamming those signals, then other sensor nodes can be activated and programmed to send signals at a different frequency. In such a case, the intruder will not come to know about the signal transmissions with this new frequency. Thus the intruder can be kept at bay.

Acknowledgment

We would like to thank Michael Sirivianos, one of the authors of the paper titled 'Non-Manipulatable Aggregator Node Election Protocols for Wireless Sensor Networks', currently with Duke University, for his quick pointers about his research paper and for the fact that we will have to take corporate permission from NEC Europe Ltd to use his research code as foundation for our work. He also gave valuable pointers towards various standard conferences about Wireless Sensor Networks.

Abstract

Wireless Sensor Networks is an emerging topic wherein nodes are organized into clusters for better scalability and efficiency. The nodes elect one of them as their leader who takes care of certain cluster related tasks. As the operations of the leader role are highly power-consuming, the role of the leader needs to be switched among the nodes in the network periodically. In hostile environments, security of the cluster and of the leader election itself is a critical issue. The algorithm presented in [1] offers a highly secure leader election to take place. However, the drawback of that algorithm is that the clusters lifetime is almost fixed at the time of cluster deployment. If the cluster reaches its end time with a few of its nodes still being active and functional, it would be a serious waste of resources and efforts to redeploy a new network if needed. Hence we have come up with an enhancement to the algorithm proposed in [1] to increase the lifetime of the cluster if the nodes in the cluster are still active.

1. Introduction

A sensor network is a kind of distributed systems wherein the nodes are highly resource-constrained in terms of capability, power, communication, etc. The nodes are specifically designed to perform a small set of tasks until they lose their power. The functionality of the sensor network is implemented at the operating system level, instead of having it as a separate application, since the overhead of having an OS and running a sensor node operations on top of the OS is higher. Hence the functionality of the sensor network nodes is a part of OS. Also as stated by Andrew S. Tenenbaum in his book [10], *operating systems security and network security are so intertwined that it is really impossible to separate them,* this project is considered to be very closely related to our operating system class.

In this report, section 2 describes the theoretical background of our project, which includes the basics of Wireless Sensor Networks (WSN)

2. Theoretical background

A WSN consists of spatially distributed autonomous sensors to cooperatively monitor physical events such as movement of intruders, or environmental conditions, such as temperature, vibration, pressure, motion or pollutants. A sensor node typically consists of a small micro-controller to do the required sensing, a wireless communicating device, and an energy source such as a small battery [5]. The critical issues with the sensor nodes are: limited battery power, computation power and storage, and smaller communication bandwidth. In some information sensitive applications, communication security and secure leader election are highly desirable. In [1] and [2], the authors have dealt mainly with the the process of electing cluster leader securely. Communication security of out of scope of our project. This paper only deals with extending the lifetime of a sensor cluster.

For a cluster to be functional, just the presence of active sensor nodes is not sufficient. They also need to have a leader who can manage them. Hence leader election is a an important part of our project.

2.1 Leader Election

There can be several criteria to select a leader, but the main requirement is that all the nodes in a cluster must consider the same criteria while electing a leader. Only then, all nodes can arrive at a consensus on the leader. Both [1] and [2] propose algorithms such that all nodes consider the residual energy in

the nodes as the criteria while electing their leader. The probability of each candidate node i.e., every node whose energy level is atleast equal to the minimum required threshold value, becoming the leader is equal.

The algorithm proposed in [1] makes use of two security tools*:*

- ***One-way hash function:*** A one-way key chain is represented as *{K0,K1,K2, ...,KR}* where each key is calculated using a one-way hash on its previous node. Given a valid key *Ki*, the authenticity of the next key *Ki*+1 can be verified by using the hash function on *Ki*+1 and comparing the result with Ki. In the algorithm proposed in [1], this key-chain concept is used to evaluate the authenticity of the the election values received by the other nodes in a cluster.
- ***Blundo's key pre-distribution:*** This algorithm is used for pairwise key distribution between each pair of node. In the algorithm suggested in [1], before deployment of a particular cluster, the base station (server) generates a t-degree symmetric polynomial f(i,j) where i and j are node IDs and f(i,j) = f(j,i). The server assigns f(i,*) to each node during deployment, and later, to establish pair-keys between nodes i and j, nodes i and j, calculate f(i,j) and f(j,i) respectively.

2.2 Existing protocol description

The leader election is a performed in the following steps.
All these steps have been implemented in out simulation unless otherwise mentioned.

Step 1 - Initialization

- While deploying a node, the server will be provided with:
  a. Two one-way key chains, YES and NO chains. NO chain tells the nodes in the cluster that the corresponding node has no energy left to perform the role as a cluster leader. Therefore, that node only includes a key chain commitment *Ni,*0 and a NO key *Ni,*1. The YES chain tells that the corresponding node has sufficient energy to act as the cluster leader. Therefore, it includes a key chain commitment *Yi,*0 and a number of YES keys *{Yi,1, Yi,2, ..., Yi,R}*, where *R* is the maximum number of times leader election can take place in a cluster.
  b. A unique ID: it is simply the hash of the two key chain commitments of the of a node. $i = H(Yi,0||Ni,0)$
  c. The Blundo's keying function f(*,*) for pairwise key establishment that successfully authenticates the node IDs to each other. Hence once the cluster is formed all the nodes will have the same view of the cluster i.e, same list of node IDs, *Linit*.

- All nodes exchange their key chain commitments with each other that are verified using the hash function described previously. After authenticating the other member nodes, each node will run a shuffle algorithm just to shuffle the member list *Linit* to create a member list *Lcandi* that is random in nature.

Step 2 - Anti-spoofing announcement

- Each cluster member will check its residual energy. If it is less than a certain energy threshold *Eth* required to be eligible for a leader candidate-ship, then it broadcasts a NO, informing the other nodes that it is not left with enough energy to serve as the leader. On the other hand, if a node's residual energy is equal to or greater that *Eth* then it will broadcast the next key in its YES chain to announce its willingness to be the next cluster leader. On receiving a NO from a node, all the other nodes will update their *Lcandi* by deleting its node ID from the list. However

that node still keep performing it basic operation of sensing and communicating data to the leader until its entire battery power is exhausted.

Step 3 - Distributed Decision Making

- Since clusters can be deployed in a hostile environment such as highly stormy weather, communication failures between nodes are likely to occur . Hence, a node will try to send its key chain values to other nodes by broadcasting for a maximum of α times.
- If a node is not sending its values during β consecutive times, then the node is assumed be inactive and it will be permanently removed from the candidate list.
- After doing the above updates to the candidate list, the first active node in the current candidate list will be chosen to be the next leader of the cluster.

2.3 Research regarding the above algorithm and our enhancement

The authors claims that new nodes can be added to an existing cluster. The only prerequisite is that the new node should meet the requirements of certain cluster formation protocol and have sufficient residual energy to participate in the leader election process. To become a member, the new node needs to perform the following steps:

- establish pairwise keys with other members in cluster.
- broadcast its two key chain commitments
- collect the released NO keys and the fresh YES keys from the other cluster members

Once the already existing cluster members gets an updated list of cluster members including the new node added, they need to just run the shuffle algorithm again to get the new candidate list. And starting from the next iteration of node election, the new node just now added can also participate in the leader election process.

The disadvantage of this algorithm is that the cluster will run only a fixed number of iterations of leader election, R, which is predefined during the cluster formation phase. No more leader election iterations means that the system will virtually get useless even if many nodes are active. Considering the facts that leaders utilize more of their residual energy when compared to the rest of the nodes, and the leader is elected randomly, the residual energy within the nodes will vary, especially after adding newer nodes. Hence some nodes have to be having residual energy even after all the R iterations of leader election are completed. This situation is a waste of nodes, which is not desirable. Another important issue is since all the nodes become active as soon as they are deployed, all the nodes exhaust their battery considerable faster. Hence we have come up with a novel idea to handle this scenario and we have not yet found any paper which concentrated on this aspect of the sensor networks.

In the algorithm presented in [1], all the deployed nodes are active right from the time the they are deployed, i.e, they will be sensing data as long as they are alive and also they are part of the leader election process as long as possible. The disadvantage of this scheme is that all the nodes continue deplete their energies concurrently and hence the network goes down fast. In our enhancement we are going to deploy the same number of nodes, but the activation of the nodes is done in two steps. The key features of our enhancement are:

1) After deployment of nodes, we would be activating only a few of them. The rest are marked as dormant.
2) The number of keys generated will still remain as R.

3) The dormant nodes will be passive nodes which keep listening to the broadcast messages and keep updating their candidate lists accordingly, and also listen to any message which commands them to become active.
4) We have another passive entity in the network called the controller. It performs the following roles
    - It keeps listening to all the leader election broadcast messages, runs the same algorithms as the other active nodes, maintains a Global Candidate List, which is updated during every round of leader election. However, is does not perform functions related to the other nodes such as sensing, or broadcasting data. It does not broadcast its messages not does it take part in leader election. It is just a passive listener of the leader election messages and always has the latest list of candidate and dormant nodes. This takes the advantage of one of the key features of the protocol, distributed synchronization of the candidate list
    - When no leader is elected for a particular round, it means that either there was some communication problems or all the nodes are dead in the same sensing round at once. The controller keeps tracks of whether the leader election took place or not by monitoring its own copy of the candidate list. If one is not elected, then it signals the network to start leader election again.If a leader is not elected it means that no nodes with enough energy are present in the network. The controller then will signal few of the dormant nodes to become active. As there is no leader elected for that round all the sensors will just cache any information they have to send, this is to conserve energy. There is no extra communication required to get the nodes integrated as all the nodes would already have their keys and the candidate list on the dormant nodes is in sync with other nodes in the network. In the following round the new nodes will participate in the leader election. In spite of that if leader is still not elected then the above step is repeated. This will continue till no more dormant nodes are available to activate and cluster is still unable to elect a leader. The exact set and number of nodes to be activate is application dependent, for example the node selection itself could be geo-spatial based where by the controller can ensure that required number of nodes are present in each region of the deployment. The details about the node selection is beyond the scope of this project

Later in the report we prove that by these enhancements our algorithm keeps the network active for a longer time, allowing more data to be sensed and more number of leader election rounds to take place.

3. Hypothesis

3.1 Positive hypothesis

- More data sensing possible
- More number of leader elections rounds
- Increased cluster lifetime

3.2 Negative hypothesis

- None

4. Methodology

4.1 Collection of input data

The input to the algorithm is a configuration file with the following details such as Maximum number of iterations, maximum number of nodes in the cluster, the number of nodes to be activated initially, etc.

4.2 Algorithm

Our simulator is coded in C++. We have used all possible object-oriented concepts to keep our code more extensible.
Our algorithm has three phases:

1. Deployment phase
2. Leader Election phase
3. Sensing phase

Deployment phase:

- A user specified number of nodes are generated with a user defined number of keys.
- A unique ID is assigned to all the nodes, using a pseudo-random function. There is an option to generate either sequential keys or random keys. Debugging and understanding the algorithm is easier with sequential keys.
- Each node has a status – 'ACTIVE' or 'DORMANT'. The status of the nodes to be added later is set to DORMANT. The status of the rest of the nodes will be set to ACTIVE.
- Key assignment and ID generation is done for all the nodes, irrespective of whether they are going to be alive now or to be added later.

Leader election phase:

- After each node has the list of ID of every other node in the network, L(init), a shuffle algorithm is run to introduce randomness in L(init). A candidate list is generated L(candi). As explained earlier, the controller also maintains the latest candidate list and dormant node list.
- Every node loses certain fixed amount of energy every time it broadcasts a byte of message, and at one point its energy goes below a threshold value. Then it sends a NO key and keeps itself out of leader election. There after it is removed from the candidate list.
- If an Yes was sent by a node. then its hash value is calculated of the incoming key and compared with the previous key corresponding to that node. If they match then the node continues to be present in the latest candidate list.
- Once the all the broadcasts are done, leader is elected on a round-robin fashion.
- The simulator comes into picture when no leader is elected for a particular round. Then either restarts the election process or activates the dormant nodes as described previously.

Our simulator has all the steps of the paper's algorithm mentioned previously, namely, Initialization, Anti-spoofing and Distributed Decision Making. However the only step we have left ignored is Blundo's key distribution, and we directly assigning node IDs to *Linit.* The reasons are mainly two; Firstly, the author in [1] mentions that this is a fool-proof mechanism. Hence proving this need not be a part of our project.. We are more concentrating on increasing the lifetime of the project. Secondly, this is a term project and we have implemented the entire simulator. Hence we did not find time to implement it.

4.3 Generating the output

The program is executed with a configuration file. It generates the simulation output for both the cases – the algorithm existing in the paper, and our enhancement to that algorithm.

4.4 Testing the hypothesis

As per our experiments, we could prove that all our hypothesis were right. Below are the details:

Maximum number of nodes to be deployed: 5
Maximum interations of leader election: 500
Minimum number of active nodes to be deployed initially: 3
Sense data size: 200 bytes
Energy loss per byte: 200 microjoules

	Existing algorithm	Enhancement
Sensing time of the cluster(sec)	660	960
Duration of simulation(sec)	720.26	1020.26
Number of election iterations	13	18

5. Summary and Conclusion

This project suggested and proved an efficient algorithm to maximize the utilization of a WSN. The improvement was proved with respect to three areas: Sensing time of the cluster, Duration of the cluster and the number of leader election iterations with the same number of keys assigned in both the cases – exiting protocol and our enhancement.

6. References

[1] Qi Dong, Donggang Liu, '*Resilient Cluster Leader Election for Wireless Sensor Networks'*. In IEEE, 2009

[2] Michael Sirivianos, Dirk Westhoff, Frederik Armknecht, Joao Girao, '*Non-Manipulable Aggregator Node Election Protocols forWireless Sensor Networks'*, in ICST WiOpt, Jan. 2007.

[3] Meng-Yen Hsieh, Yueh-Min Huang, Han-Chieh Chao, '*Adaptive security design with malicious node detection in cluster-based sensor networks'*. In Elsevier, 2007

[4] D. Liu, "*Resilient cluster formation for sensor networks*" In Proceedings of the 27th International Conference on Distributed Computing Systems (ICDCS), 2007

[5] Wikipedia

[6] compilation by Jorge Nuevo, '*A Comprehensible GloMoSim Tutorial*', March 4, 2004

[7] S. Vasudevan, B. DeCleene, N. Immerman, J. Kurose, and D. Towsley, "*Leader election algorithms for wireless ad hoc networks*," in Proceedings of DARPA Information Survivability Conference and Exposition, 2003, pp. 261– 272.

[8] D. Liu, P. Ning, and W. Du, "*Detecting malicious beacon nodes for secure location discovery in wireless sensor networks*," in Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS), 2005, pp. 609–619.

[9] S. Vasudevan, N. Immerman, J. Kurose, and D. Towsley, "*Leader election algorithms for wireless ad hoc networks*". Umass Computer Science Technical Report 03-01.
[10] Andrew S. Tenenbaum, '*Modern Operating Systems, 4th Edition'*