PROTECTING LOCATION PRIVACY IN SENSOR NETWORKS AGAINST A GLOBAL EAVESDROPPER

by

KIRAN MEHTA

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2008

To my family and friends who have always been supportive.

ACKNOWLEDGEMENTS

I would like to thank my supervising professor Dr. Donggang Liu for constantly motivating and encouraging me, and also for his invaluable advice during the course of my master's studies. I wish to thank Dr. Matthew Wright and Dr. Nan Zhang for their interest in my research and for taking time to serve in my thesis committee.

I would also like to thank Qi and Jaideep from ISec lab for encouraging technical discussions which have always been more than helpful and made work environment full of excitement.

Most importantly I would like to thank my parents for believing in me and encouraging me to pursue my interest.

June 2, 2008

ABSTRACT

PROTECTING LOCATION PRIVACY IN SENSOR NETWORKS
AGAINST A GLOBAL EAVESDROPPER

KIRAN MEHTA, M.S.

The University of Texas at Arlington, 2008

Supervising Professor: Donggang Liu

While many protocols for sensor network security provide confidentiality for the content of messages, contextual information usually remains exposed. Such contextual information can be exploited by an adversary to derive sensitive information such as the locations of monitored objects and data sinks in the field. Attacks on these components can significantly undermine network applications. The existing techniques defend the leakage of location information only from an adversary who sees only local network traffic. However, a stronger adversary, the *global eavesdropper*, is realistic and can defeat all existing techniques. This paper first formalizes the location privacy issues in sensor networks under this strong adversary model and computes a lower bound on the communication overhead needed for achieving a certain level of location privacy. The paper then proposes two techniques to provide location privacy for monitored objects (source location privacy): *periodic collection* and *source simulation*, and two techniques to provide location privacy for data sinks (destination location privacy): *destination simulation* and *backbone flooding*. These techniques provide trade-offs between privacy, communication cost, and latency. The

analysis and simulation demonstrate that the proposed techniques are efficient and effective for source and destination location privacy in sensor networks.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Introduction

A wireless sensor network (WSN) typically comprises a large number of cheap, small, and resource-constrained sensors that are self-organized as an ad-hoc network to interact with and study the physical world [1]. Sensor networks can be used in applications where it is difficult or infeasible to set up wired networks. Examples include wildlife habitat monitoring, military surveillance, and target tracking.

Applications like military surveillance and target tracking provide incentives to adversaries to eavesdrop on network traffic to obtain valuable intelligence. Abuse of such information can cause monetary losses or endanger human lives. To protect such information, considerable effort in sensor network security has focused on providing classic security services such as confidentiality, authentication, integrity, and availability. Though these are critical security requirements in many applications, they are by no means sufficient. The communication patterns of sensors can by themselves reveal a great deal of contextual information, which can disclose the location information of critical components in a sensor network. For example, in the Panda-Hunter case [2], a sensor network is deployed to track endangered giant pandas in a bamboo forest. Each panda has an electronic tag that emits a signal that can be detected by the sensors in the network. (A sensor that detects this signal is called as a *source* sensor.) The source sensor then forwards the location of pandas to a data sink (destination) with help of intermediate sensors. Adversary may use the communication between sensors and the data sinks to locate and later capture the monitored pandas. As

another example, in military applications, the enemy can observe the communication and locate all data sinks in the field. Disclosing the locations of destinations during their communication with sensors may allow the enemy to launch calculated attacks against them and disable the network.

Location privacy is thus of utmost importance, especially in hostile environments. Failure to protect such information can completely subvert network applications. Location privacy measures thus need to be developed that can prevent the adversary from determining physical locations of *source sensors* and *destinations*.

Providing location privacy in a sensor network is very challenging. First, the adversary can easily intercept network traffic due to the use of a broadcast medium for routing packets. He can use information like packet generation time and packet generation frequency to perform traffic analysis and infer the locations of monitored objects and data sinks. Second, sensors are usually resource constrained. It is not feasible to apply traditional anonymous communication techniques for hiding the communication between sensor nodes and destinations. We need to find alternative means to provide location privacy considering resource limitations of sensor nodes.

Recently, privacy-preserving routing techniques have been developed for sensor networks. However, the performance and efficiency of most of these existing solutions are measured against an adversary capable of eavesdropping on limited portion of the network at a time. An highly motivated adversary can easily eavesdrop on the entire network and defeat all these solutions. For example, the adversary may decide to deploy his own set of sensor nodes to monitor the communication in the target network. This is especially true in a military or industrial spying context where the adversary has strong, potentially life-or-death, incentives to gain as much information as possible from observing the traffic in the target network. Given a global view of the network traffic, the adversary can easily infer the locations of monitored objects

and destinations. For example, a region in the network with high activity should be close to a destination and a region where the packets originate should be close to a monitored object.

In this paper, we focus on privacy-preserving communication methods in the presence of a *global eavesdropper* who has a complete view of the network traffic. The contributions in this paper are two-fold.

- We point out that the assumption of a global eavesdropper who can monitor the entire network traffic is often realistic for highly-motivated adversaries [3]. We then formalize the location privacy issues under such an assumption and compute the lower bounds on the communication overhead needed to achieve a given level privacy.
- We propose two techniques that prevent the leakage of location information of monitored objects: *periodic collection* and *source simulation*, and two techniques that provide location privacy for destinations: *destination simulation* and *backbone flooding*. Our analysis and simulation studies show that these appoaches are effective and efficient in providing location privacy for monitored objects and destinations.

The rest of the paper is organized as follows. The next section reviews existing algorithms for providing location privacy in sensor networks. Section 3.1 presents the network and adversary models. In Section 4.1, we formalize the location privacy issues in sensor networks and provide a model for evaluating location privacy. Section 5.1 discusses the proposed techniques for source and destination location privacy. Section 6.1 evaluates the proposed techniques via simulation study. Section 8.1 concludes this paper and points out some future directions.

CHAPTER 2

BACKGROUND

2.1 Existing Approaches

In this section, we describe previously-proposed algorithms for source location privacy and destination location privacy.

2.1.1 Source Location Privacy

Prior work in protecting location privacy to monitored objects sought to increase *safety period*, which is defined as the number of messages initiated by the current source sensor before a monitored object is traced [2].

The *flooding technique* [4] requires a source node to send out each packet through numerous paths to a destination to make it difficult for an adversary to trace the source. However, the problem is that the destination will still receive packets from the shortest path first. The adversary can thus quickly trace the source node using backtracking. This method consumes a significant amount of energy without providing much privacy in return.

Kamat et al. describes two techniques for location privacy. First, they propose *fake packet generation* technique [2] in which a destination creates fake sources whenever a sender notifies the destination that it has real data to send. These fake senders are away from the real source and approximately at the same distance from the destination as the real sender. Both real and fake senders start generating packets at the same time. This scheme provides decent privacy against a local eavesdropper. The other technique is called the *phantom single-path routing*, which achieves location pri-

vacy by making every packet generated by a source walk a random path before being delivered to the destination. As a result, packets will reach the destination following different paths. This algorithm is quite effective in dealing with a local eavesdropper.

Cyclic entrapment [5] creates looping paths at various places in the sensor network. This will cause a local adversary to follow these loops repeatedly and thereby increase the safety period. Energy consumption and privacy provided by this method will increase as the length of the loops increase.

After the preliminary version of this paper was published, several *source location privacy* techniques have been proposed to deal with global eavesdroppers. Yang et al. propose to use proxies for the location privacy of monitored objects under a global eavesdropper [6]. The network is partitioned into cells where sensors in each cell communicate with the nearest proxy. Each cell sends traffic that follows an exponential distribution to its nearest proxy. The traffic will include dummy packets if real packets are not available. The proxies filter out dummy packets and send data to destination. The proxies also send dummy packets to destination if real event packets are not available. All packets are appropriately encrypted so that adversary is not able to distinguish between real and dummy packets. *Proxy-based filtering* and *tree-based filtering* schemes are proposed to position proxies. In addition, Shao et al. propose to reduce the latency of real events [7] without reducing the location privacy under a global eavesdropper. The technique makes sure that the adversary cannot determine the real traffic based on statistical analysis.

2.1.2 Destination Location Privacy

Deng et al. described techniques to provide fault tolerance against failure or compromise of individual destination or sensor nodes [8]. They also introduced a

technique to protect the locations of destinations from a local eavesdropper by hashing the identification fields in packet headers.

Deng et al. also presented four techniques to protect the location privacy of destination from a local eavesdropper who is capable of carrying out time correlation and rate monitoring [9]. First, they propose a multiple parents routing scheme in which for each packet a sensor node selects one of its parents randomly and forwards the packet to that parent. This makes the traffic pattern between the source and the destination more dispersed than the schemes where all the packets travel through same sequence of nodes. They then introduce techniques using controlled random walk, random fake paths, and hot spots. The controlled random walk technique adds a random walk to the multiple parents routing scheme causing the traffic pattern to be more spread out and hence less vulnerable to rate monitoring. The random fake path technique is introduced to confuse an adversary from tracking a packet as it moves towards the destination, mitigating the time correlation attacks. In differential fractal propagation (DFP) technique, whenever a node transmits a real packet, its neighbor node generates a fake packet. This fake packet travels configured number of hops to confuse the adversary. They also designed a scheme for creating some areas of high activity locally in the sensor network called *hot spots*. If such an area receives a packet, the packet has high probability of traveling through the same sequence of nodes creating an area of high activity. A local eavesdropper may be deceived into believing that this area is close to a destination. However, a global eavesdropper can notice that only some packets generated by real objects pass through this hot-spots and conclude that the destination may not necessarily be close to those hot spots.

Jian et al. proposed the location privacy routing protocol (LPR) for destination location privacy [10]. The LPR algorithm provides privacy to the destination with help of redundant hops and fake packets when data is sent to the destination. Each

time a packet is forwarded to the next hop, the packet may move either closer or away from the destination. Along with the real data packets, sensors may generate fake packets that travel away from the destination to confuse the adversary.

However, these existing techniques assume a local eavesdropper. If an adversary has the global knowledge of the network traffic, it can easily defeat these schemes. For example, the adversary only needs to identify the region of high activity to locate the destination. In this paper, we will focus on privacy-preserving techniques against a global eavesdropper.

CHAPTER 3

SYSTEM MODEL

3.1 Network and Adversary Model

Although prior research has attempted to solve location privacy problems for sensor networks, prior attacker models are not strong enough to model a well-funded, motivated adversary. In this section, we describe the network and adversary models that we study in this paper.

3.1.1 Network Model

Sensor networks are a relatively recent innovation. There are number of different types of sensor nodes that have been and continue to be developed [11]. These range from very small, inexpensive, and resource-poor sensors such as SmartDust up to PDA-equivalent sensors with ample power and processing capabilities such as Stargate. Applications for networks of these devices include many forms of monitoring, such as environmental and structural monitoring or military and security surveillance.

In this paper, we consider a *homogeneous network model*. In the homogeneous network model, all sensors have roughly the same capabilities, power sources, and expected lifetimes. This is a common network architecture for many applications today and will likely continue to be popular moving forward. It has been well studied and provides relatively straightforward analysis in research as well as simple deployment and maintenance in the field.

3.1.2 Adversary Model

For the kinds of wireless sensor networks that we envision, we expect highly-motivated and well-funded attackers whose objective is to learn sensitive information such as the locations of monitored objects and destinations.

The objects monitored by the network may be critical. Any damage to such objects can cause monetary losses or issues in critical military applications. Destinations are also critical components of sensor networks. In most applications, destinations act as gateways between the multi-hop network of sensor nodes and the wired network or a repository where this information is analyzed. Unlike failure of some sensors, failure of destinations can create permanent damage to sensor network applications. Compromise of a destination will allow an adversary to gather all the information because in most applications data won't be encrypted after it is received by a destination. In some military applications, an adversary could locate destinations and make the critical sensor network non-functional by destroying them. It is thus highly critical to protect the location information of monitored objects and destinations to keep the sensor network functional and useful.

In this paper, we consider *global eavesdroppers*. For a motivated attacker, faster and more effective attacks can be done to locate monitored objects and destinations through eavesdropping on the entire network. The attacker can simply deploy his own sensor network to monitor the target network. Note that, at the current price for a BlueRadios SMT Module at $25, the attacker needs only $25,000 to build a network of 1000 nodes [12]. Further, the number of nodes can typically be smaller than the the number of nodes in the target network as they monitor wireless radio signals instead of directly sensing the environment. Thus, for even moderately valuable location information, this can be worth the cost and trouble.

Although such an eavesdropping sensor network would face some system issues in being able to report the precise timing and location of each target network event, we do not believe that these would keep the attackers from learning more approximate data values. This kind of attacker would be able to query his own sensor network to determine the locations of observed communications. He could have appropriate sensors send beacon signals that could then be physically located. He could equip his sensors with GPS to get precise location information, or use localization algorithms that avoid the costs of GPS [13, 14].

In any case, it should be feasible to monitor the communication patterns and locations of events in a sensor network via global eavesdropping. An attacker with this capability poses a significant threat to location privacy in these networks, and we therefore focus our attention to this type of attackers.

CHAPTER 4

PRIVACY MODEL

4.1 Privacy Evaluation Model

In this section, we describe our privacy evaluation model with which we quantify the location privacy of critical components in sensor network applications. In this model, the adversary deploys an *attack network* to monitor the sensor activities in the target network. We consider an adversary who can monitor transmissions of all the sensors in the network with help of his own sensor network. Each sensor i is an observation point and a tuple (i, t, e) is available to the adversary through observing each of its packet e at the time t. We assume that all the transmissions are encrypted and hence the actual useful information available to the adversary is (i, t). We assume that the network starts functioning at time $t = 0$.

4.1.1 Assumptions

In some sensor network applications, sensor nodes may not be easily accessible to the adversary. In other applications, the adversary may have physical access to sensor nodes which makes such applications vulnerable to node compromise. Having a set of compromised sensors in the network will provide advantages to the adversary. However, in this paper, we assume that an adversary does not compromise sensor nodes. We will seek solutions to the problem of node compromises in the future. We assume that we can protect the monitored objects if we can protect source sensors. *We use the terms objects and sources interchangeably.* We assume that when the location information of destinations are to be protected, the location information of

sources are known to the adversary and vice-versa via eavesdropping on the network. The set of components whose location information is to be protected is called the `protected set`. The set of components whose location information is available to the adversary is called the `available set`. We assume that all destinations and sensors are stationary but objects are mobile.

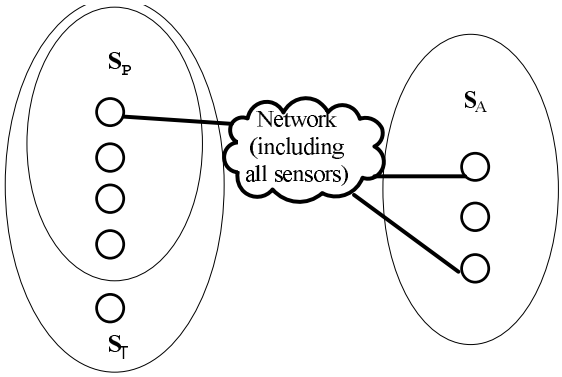
4.1.2 The Attacker

Table 4.1. Frequently used notations in the paper

b	Level of privacy in terms of bits
T	Time period for which the network has been functioning
S_P	The `protected set`
S_A	The `available set`
S_T	Set of sensors in whose range the adversary expects to find protected components at the end of time T
I	Set of all sensor IDs in the network
N	Number of sensor nodes in the network
n_b	Average number of neighbors of a sensor

Below we describe our privacy model in detail. We will first describe the privacy model considering source location privacy and then extend it to include destination location privacy. Table 4.1 lists some frequently used notations.

As shown in the Figure 4.1, a sensor network deployed for an application can be viewed as a graph $G = \{V, E\}$ where the set of vertices V is the union of the set I of sensor nodes, the set of sources, and the set of destinations. The set E of edges includes all direct communication links between sensor nodes physically close to each other. At any point in time, from the global eavesdropper's point of view, the



Set S_P	The set of components to be potected
Set S_T	The set of sensors in whose range the adversary expects to find the protected components
Set S_A	The set of components whose locations are known to the adversary

Figure 4.1. Network viewed as a graph.

network can be considered to be including a set S_P comprising of a set of sources, a set S_A which represents the destinations where the data is sent, and a set of sensors that transfer data between sources and destinations as shown in the Figure 4.1.

An adversary may eavesdrop on the entire network with the intentions of physically locating objects. Let $O_{i,T}$ be the set of all observations collected by the adversary about the node i by the time T. Thus, at time T, the knowledge that an attacker can obtain from eavesdropping on the target network is

$$O_T = \bigcup_{i \in I} O_{i,T}$$

The objective of the adversary is to identify a set $S_T \subset I$ of nodes that represents the set of sensors in whose range the adversary expects to find objects at time T. Informally, the attacker will not believe that a lone observation (i, t) indicates the presence of an object. The presence of an object should generate a *trace*, which is

a set of observations over the lifetime of the network up to time T. This means that there will be legitimate looking traffic between each object and a subset of destinations. More precisely, as shown in Figure 4.1, for each $i \in S_T$ and for a set $K \subseteq S_A$, there must exist a set $A_{i,K} \subset O_T$ that can be exactly generated due to communication of the nodes in K with an element from set S_P whose position at time T is in the signal range of node i. We call a set of observations a *candidate trace* if the adversary believes that it can be generated due to the communication between an element of set S_P and a subset of elements from set S_A.

For the attacker to determine whether a set of observations is a candidate trace, we define a *pattern analysis* function

$$f : 2^{\ddot{O}_T} \rightarrow I \cup \{\perp\}$$

where $\ddot{O}_T$ is the set of all possible observations, i.e., $\ddot{O}_T = \{(i,t)\}_{i \in I, 0 \leq t \leq T}$.

The *pattern analysis* function outputs the ID of the sensor in whose range the target is supposed to be located at the time T, if the set of observations is a candidate trace, and returns $\perp$ otherwise. For simplicity, we assume that the pattern analysis does not return fractional values, e.g. a probabilistic measure of the chance that a trace is a candidate trace or not. We say that a pattern analysis function is *perfect* if it can identify all candidate traces without error, i.e. without false positives or false negatives. In this paper, we consider a strong adversary who uses a perfect pattern analysis function. Let f_p be such perfect pattern analysis function. We have

$$S_T = \{i \mid \exists A_{i,K} \subseteq O_T, K \subseteq S_A, (i = f_p(A_{i,K})) \neq \perp\}.$$

4.1.3 Measuring Privacy

Privacy can be measured by the size of S_T. We assume that the nodes in S_T are equally likely to be close to objects. The probability of any sensor node in S_T being close to an object can be estimated by $\frac{|S_P|}{|S_T|}$. Hence, we formally define the location privacy of our system as the entropy

$$b = \sum_{|S_T|} -\frac{1}{|S_T|} \log_2 \frac{|S_P|}{|S_T|} = \log_2 \frac{|S_T|}{|S_P|}$$

We can use this notion to define the optimal privacy. Let $\ddot{S}_T$ represents the set of all possible locations for the real target at time T based on the set of all possible observations $\ddot{O}_T$, i.e.,

$$\ddot{S}_T = \{i \mid \exists A_{i,K} \subseteq \ddot{O}_T, K \subseteq S_A, (i = f_p(A_{i,K})) \neq \perp\}.$$

For simplicity, we always assume that an object can be anywhere in the deployment field at time T. We thus have $\ddot{S}_T = N$. We then define set $S_T^* = N$ for optimal privacy, representing the case that $\forall i \in I$ and a set $K \subseteq S_A$, $\exists A_{i,K} \subseteq O_T$ $s.t.$ $(i = f_p(A_{i,K})) \neq \perp$. In other words, there is a subset of observations in O_T (the set of observations made by the adversary by time T) that support the case for an object in the range of any sensor i. We thus have the optimal location privacy

$$b \leq \log_2 \frac{|S_T^*|}{|S_P|} = \log_2 \frac{N}{|S_P|}.$$

The level of location privacy is measured in terms of the number of bits. Depending on the users and applications, this can be easily modified to support different

kinds of privacy measurement models. For example, we can define high, medium and low privacy levels using appropriate values of b.

We note that the traffic in the network can cause the level of privacy to vary. The privacy would go lower if the attacker ascertains that a particular trace is no longer a candidate trace. If a candidate trace splits into two candidate traces, then the level of privacy goes up because S_T grows. The interpretation of this depends on the sensor network application and the attacker model considered. For example, if the attacker seeks to physically destroy the object being observed with a missile (instant attack), then the privacy should be taken as the minimum at any time before T. In cases where the attacker must spend time to investigate candidate locations, then the average privacy over time is adequate. We provide a snapshot of the privacy at a given time, which can be used for either purpose.

4.1.4 Privacy and Communication Costs

In the following, we explore the relationship between the level of privacy and the amount of communication overhead. To minimize the communication overhead, we need to minimize the total communication overhead required for all the candidate traces in the network.

We simplify our model to understand the effect of different policies on the overheads and location privacy in the network. We model the communication in sensor networks as a discrete time system with a granularity of Δ. Specifically, the time line is divided into a number of time intervals with equal length of Δ. The communication between sensor nodes happens at the end of each time interval, i.e., at time $\{\Delta, 2\Delta..., i \times \Delta, ...\}$. A sensor node can receive all the packets targeted to itself and will send or forward no more than one packet at any time interval. Clearly, when a sensor node receives multiple dummy packets during a given time interval, it

only needs to forward one of them to save the communication cost. Intuitively, the larger the value of Δ, the more communication cost we can save. Let α be the number of time intervals required for an event regarding an object to occur in the network. Thus there will be a `communication round` between a source and destinations every α intervals. Let K_{davg} be the average number of real destinations a source communicates with. We have the following theorem.

Theorem 1 *To achieve b-bits of source location privacy, the communication cost is at least*

$$\omega_T = \sum_{i=1}^{\frac{T}{\alpha \times \Delta}} M_s^i$$

where M_s^i is the weight of a Steiner tree connecting given $2^b \times |S_P| + K_{davg}$ nodes in the network during i^{th} round of communication between sources and destinations. T is the time for which the network has been functional. K_{davg} is the average number of real destinations a source communicates with and α is the number of intervals required for an event pertaining to an object to occur in the network.

Proof 1 *Let $S_T = \{f_1, ..., f_l\}$ be the candidate set identified by the adversary. We have $b = \log_2 \frac{l}{|S_P|}$ and thus $l = 2^b \times |S_P|$. We need to minimize the communication cost required for these l candidate traces in the network. In other words, considering a graph that represents our network, we need to find a minimum weight tree that connects $2^b \times |S_P|$ sensors and K_{davg} destinations. This is the well-known Steiner tree problem. Given an arbitrary weighted graph and a fixed set of vertices, the Steiner tree problem is the task of finding a minimum weight subtree containing all the given vertices, where the weight of the tree is the sum of the weights of the edges it covers [15]. Steiner tree is a well-studied concept in the literature. It is an NP-hard problem and many approximation algorithms have been proposed to construct it. We have used a simple approximation algorithm from [16] for our purpose. When a Steiner tree is*

being constructed, additional nodes called as Steiner points can be added to the graph. However, in practice sensor networks are dense enough and hence we assume that any location where a Steiner point could possibly be added has a sensor node already. In any case, we should get a good approximation. Weight of an edge in our case is the hop distance between the sensors representing that edge. In general, the Steiner tree problem is an NP-hard problem [15]. The lower bound on weight of Steiner tree connecting given sources and destinations would give us the minimum communication cost for sending packets from sources to destinations. Packets transmitted by sources would travel to destinations via edges in the tree and if a node in the tree receives multiple fake packets in one time interval, it can drop as many as possible to save energy. Since M_s^i is the the weight of the Steiner tree connecting $2^b \times |S_P|$ sensors and K_{davg} destinations during the i^{th} round of communication between sources and destinations, the communication overhead needed to achieve b bit of privacy can be estimated by

$$\omega_T = \sum_{i=1}^{\frac{T}{\alpha \times \Delta}} M_s^i$$

We have mentioned details about communication cost for protecting the locations of objects. Similar argument can be made for finding the communication cost for protecting the locations of destinations in a network. In this case, the set of available locations S_A would be the set of source sensors, the set of components to be protected S_P would be the the destinations, and S_T would be set of sensors identified by the adversary. The objective of the adversary would be to find the destinations instead of the objects. As locations of all objects in the network are known to the adversary via eavesdropping, fake sources are of no use. In addition, generating dummy packets will be of little use for destination location privacy since communication only happens when there are real monitored events, which we assume can be detected by

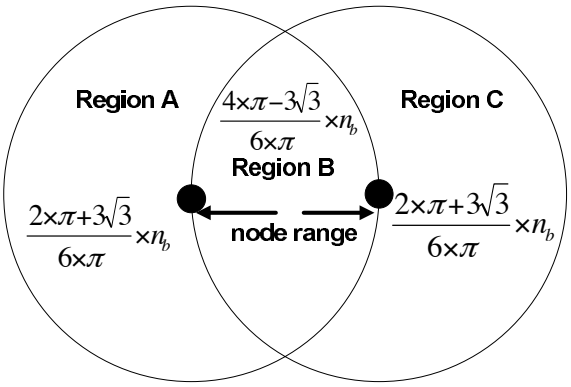


Figure 4.2. Minimum number of neighbors two adjacent sensor nodes can share.

adversaries as well. Since an intermediate node cannot drop packets from different sources, we can consider traffic for each object separately from others.

Theorem 2 *To achieve b-bits of destination location privacy, the communication cost is at least*

$$\omega_T = \sum_{i=1}^{\frac{T}{\alpha \times \Delta}} M_d^i \times |S_A|$$

where M_d^i is the weight of a Steiner tree that connects a source and K_{davg} sets each containing $\lfloor(|\frac{2^b - \frac{4\times\pi-3\sqrt{(3)}}{6\times\pi}\times n_b - 1}{\frac{(2\times\pi+3\sqrt{(3)})}{(6\times\pi)}\times n_b}|) + 1$ nodes in the network during i^{th} round of communication. T is the time for which the network has been functional and α is the number of intervals required for an event pertaining to an object to occur. K_{davg} is the average number of destinations a source communicates with.

Proof 2 *Let $S_T = \{f_1, ..., f_l\}$ be the candidate set identified by the adversary. We have $b = \log_2 \frac{l}{|S_P|}$ and thus $l = 2^b \times |S_P|$. Please note that by our assumption the adversary is aware of the locations of all objects in the network. To achieve b bits of destination privacy, a source has to communicate with at least 2^b fake destinations in addition to each real one. Thus, for the privacy of one real destination, we need to minimize the communication cost required between a source and 2^b fake destinations.*

Note that one packet from a sensor can be received by all neighbors of this sensor. Let n_b be the average number of neighbors of a sensor. The figure 4.2 shows the minimum number of neighbors shared by two adjacent sensor nodes. This means will need at least $\lfloor(|\frac{2^b-\frac{4\times\pi-3\sqrt{(3)}}{6\times\pi}\times n_b-1}{\frac{(2\times\pi+3\sqrt{(3)})}{(6\times\pi)}\times n_b}|)+1$ *packets. This is the minimum number of packets required for an event to be received by* 2^b *sensors. In other words, only* $\lfloor(|\frac{2^b-\frac{4\times\pi-3\sqrt{(3)}}{6\times\pi}\times n_b-1}{\frac{(2\times\pi+3\sqrt{(3)})}{(6\times\pi)}\times n_b}|)+1$ *sensors would need to transmit a given event so that all* 2^b *sensors and a real destination can receive it. The adversary would believe that destination could be in range of any of these* 2^b *sensors.*

Considering a graph that represents our network and the knowledge that a source communicates with K_{avg} real destinations on average, we need to find a Steiner tree that connects a source and K_{davg} sets each containing $\lfloor(|\frac{2^b-\frac{4\times\pi-3\sqrt{(3)}}{6\times\pi}\times n_b-1}{\frac{(2\times\pi+3\sqrt{(3)})}{(6\times\pi)}\times n_b}|)+1$ *nodes. Finding Steiner tree is an NP-hard problem and many approximation algorithms have been proposed to construct it. We have used a simple approximation algorithm from [16] for our purpose. Let M_d^i be the weight of such Steiner tree during the i^{th} round of communication between sources and destinations. As weight of the tree represents the minimum number of hops required to send an event from a source to destinations, for $|S_A|$ objects we get*

$$\omega_T = \sum_{i=1}^{\frac{T}{\alpha\times\Delta}} M_d^i \times |S_A|$$

Though calculating the actual Steiner tree is an NP-hard problem [15], it gives us a benchmark to compare efficiency of source and destination location privacy techniques. To the best of our knowledge, [15] is the only work till date that studies the lower bound on the weight of a Steiner tree connecting a randomly chosen set of vertices. However, the lower bound in this paper is for a graph that has edge weights chosen randomly from an exponential distribution with mean 1, which is not

applicable to our network. We thus compare our proposed techniques against the weight of Steiner trees obtained through simulation. We calculate average weight of a Steiner tree by using a simple approximation algorithm proposed in [16] whose cost ratio is not more than twice the optimal. This algorithm starts from a graph $G = \{V, E\}$ and an input set $S' \subseteq V$ of nodes to be connected. A node is randomly selected from S' and connected to the nearest node from S' via the shortest path. This forms a tree containing at least two nodes from S'. Next node from S' is selected such that the selected node is not part of the tree constructed so far. The selected node is connected to the closest node of the tree. This process continues until all the nodes from S' are in the tree. In each step to connect a new node to the existing tree, we use dijkstra's single source shortest path algorithm which calculates the shortest distances from the new node to the existing tree.

CHAPTER 5

PRIVACY PRESERVING ROUTING TECHNIQUES

5.1 Privacy-Preserving Routing Techniques

In this section, we present the proposed privacy-preserving techniques for protecting the location information of monitored objects and data sinks. In this paper, we assume that all communications between sensor nodes in the network are encrypted so that the contents of all packets appear random to the global eavesdropper. Many key pre-distribution protocols can be used for this purpose [17, 18, 19].

5.1.1 Source Location Privacy Techniques

In this section we describe two techniques to provide location privacy to monitored objects in sensor networks, *periodic collection* method and *source simulation* method. The periodic collection method achieves optimal location privacy but can only be applied to applications that collect data at a low rate and do not have strict requirements on the data delivery latency. The source simulation method provides practical trade-offs between privacy, communication overhead, and latency; it can be effectively applied in real-time applications.

5.1.1.1 Periodic Collection

As described in Section 2.1, most existing privacy-preserving techniques fail against a global eavesdropper. The primary reason is that the presence of a real object in the network will change the traffic pattern, and the global eavesdropper can easily pinpoint where the change happens. An intuitive solution to this problem is

to make the network traffic patterns independent of the presence of real objects. To achieve this, our periodic collection method has every sensor node independently and periodically send packets at a reasonable frequency regardless of whether there is real data to send or not.

To enable this, each sensor node has a timer which triggers an event every Δ seconds, as well as a first-in-first-out (FIFO) queue of size q for buffering received packets that carry real data reports. When an event is triggered by the timer, the node checks if it has any packet in its queue. If so, it dequeues the first packet, encrypts it with the pairwise key it shares with the next hop, and forwards it to that next node. Otherwise, it sends a *dummy packet*, with a random payload, that will not correctly authenticate at the next hop. Since every sensor node only accepts the packets that correctly authenticate, dummy packets do not enter the receiver's queue. When the queue at a sensor node is full, it will stop accepting new packets.

5.1.1.1.1 Privacy

The periodic collection method provides the optimal location privacy one can ever achieve in the network since the traffic pattern is entirely independent of the activity of real objects. We now present this argument more formally.

Theorem 3 *The privacy achieved by the periodic collection method is*

$$b = \log_2 \frac{N}{|S_P|}$$

Proof 3 *Since the object can be anywhere in the field at time T, we know that for any $i \in I$, there exists a candidate trace $A_{i,K} \subset \ddot{O}_T$ with $f_p(A_{i,K}) = i$ for $K \subseteq S_A$. Now, we consider a real object that can generate a candidate trace $A_{i,K}$. If such an object is present in the field, it will also produce a set of observations $A'_{i,K}$. According to the*

definition of f_p, we have $f_p(A'_{i,K}) = i$. Since the traffic does not change no matter how the object behaves in the network, we know that $A'_{i,K} \subset O_T$. This indicates that $S_T = I$. Hence, we have

$$b = \log_2 \frac{|S_T|}{|S_P|} = \log_2 \frac{N}{|S_P|}.$$

5.1.1.1.2 Energy consumption

It is generally believed that communication in sensor networks is much more expensive than computation [20]. For a privacy-preserving routing technique, its energy consumption can thus be measured by the additional communication overhead used for hiding the traffic carrying real data.

Since the network starts operation at time 0, the total number of data packets transmitted in the network can be estimated by $\frac{T \times N}{\Delta}$. Certainly, a small Δ indicates a large amount of additional traffic for our periodic collection method. This means that this method cannot handle real-time applications very well. However, we do believe that this method is practical for applications where Δ can be set large enough for a reasonable amount of covering traffic.

We now estimate the minimum communication overhead needed to achieve the optimal privacy using theorem 1 discussed in the section 4.1. The problem is to find the Steiner tree that connects all N nodes and destinations. Thus, in this case the steiner tree problem reduces to finding the weight of a minimum spanning tree of the graph [15]. The weight of a minimum spanning tree for a graph of N nodes and destinations is at least N. Thus, the minimum communication cost by the end of time T would be:

$$\omega_T = \frac{T \times N}{\alpha \times \Delta}.$$

Assume that Δ is set properly such that an object will trigger an event to the destination for each time interval Δ. In other words, we have $\alpha = 1$. This indicates that the performance of the periodic collection method is close to the optimal solution in terms of the communication overhead needed to achieve the optimal privacy.

5.1.1.1.3 Latency

Sensor networks can support a wide range of applications. Different applications may have different requirements that may affect the application of the periodic collection method in real-world scenarios. Example of these requirements include *the latency of a real event being reported to the destination* and *the network lifetime*.

Obviously, parameter Δ determines the lifetime of the network. Hence, it should be set judiciously. In general, an application that demands very low latency should set Δ to a low value. However, this would lead to high energy consumption as a large number of fake packets will also be generated. Hence, there is a trade off between energy consumption and latency depending on the value of Δ.

The queue size q is a factor which determines the number of real packets that a sensor node can buffer. This will affect how well the periodic collection method can handle the situation when real events are generated frequently in the network. Increasing the value of q will allow the queuing of more real packets and thus will help the network in forwarding more information about real objects to the destination. In other words, the number of packets dropped during the travel to the destination can be reduced. However, we have to realize that having a large q may increase the average latency of a real packet reaching the destination. This occurs because a newly received packet that carries real data may need to wait for a long time before getting forwarded in case of a large queue. We give a detailed simulation study of the effects of different values for parameter q in Section 6.1.

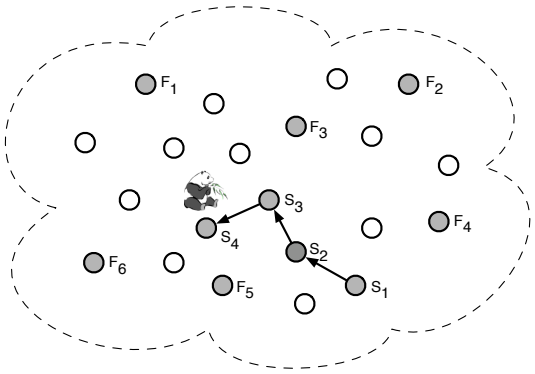


Figure 5.1. Movement pattern leaks the location of the object.

5.1.1.2 Source Simulation

Though the periodic collection method provides optimal location privacy, it consumes a substantial amount of energy for applications that have strict latency requirements. It is clearly not well-suited for real-time applications.

In the periodic collection method, every sensor node is a potential source node. To reduce energy consumption, we choose to reduce the number of potential sources in the network. In other words, we will trade off privacy with communication overhead. For this purpose, we propose to *create multiple candidate traces in the network to hide the traffic generate by real objects*. How to determine the number of candidate traces is application-dependent. In general, we expect that this number is much smaller than the size of I.

Creating candidate traces in the field is quite challenging in practice. The main problem lies in the difficulty of modeling the behavior of a real object in the field. A poorly-designed model will likely fail in providing privacy protection. For example, as shown in Figure 5.1, the behavior of an object is modeled *inaccurately* as a static object in the network. Based on this model, the candidate traces are created at

places $\{F_1, F_2, ..., F_6\}$. Sensors at each of these places will send virtual traffic to the destination, simulating a real object. However, the actual object behavior may be quite different from the model used by the defender. The adversary may notice how the object *actually* moves around in the field and use such extra knowledge to identify the possible locations of objects. As shown in Figure 5.1, the object moves from S_1 to S_4 along the path $\{S_1, S_2, S_3, S_4\}$. In this case, the traffic generated by the object exhibits a sequential pattern from S_1 to S_4. A global eavesdropper can then quickly determine that the traffic generated from places $\{F_1, F_2, ..., F_6\}$ is likely to be virtual, and the traffic generated from $\{S_1, ..., S_4\}$ is likely to be generated by a real object. As a result, the adversary can easily tell that the object is currently close to position S_4.

Fortunately, in most cases, an adversary will not have substantially more knowledge about the behavior of real objects than the defender. Even if the attacker learns about the object behavior over time, the defender will learn the same behavior and can broadcast occasional updates to the object movement model. Thus, it is often reasonable to assume that the adversary and the defender have similar knowledge about the behavior of real objects. We can then create more useful candidate traces in the field to hide real objects. Though it is challenging to model real objects, research on learning and modeling behavior of objects are quite active. We believe that it will not be a very difficult problem to obtain a reasonable behavior model for the object in the field. Such modeling of objects is beyond the scope of this paper.

5.1.1.2.1 Protocol Description

In the source simulation approach, a set of *virtual objects* will be simulated in the field. Each of them will generate a traffic pattern similar to that of a real object. Figure 5.2 illustrates the idea of this approach. In this example, objects move randomly in the

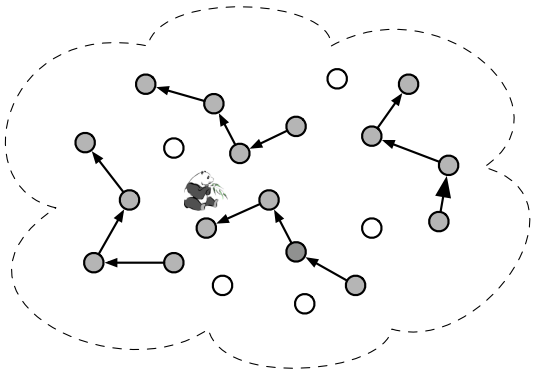


Figure 5.2. Simulating virtual objects in the field.

field. Both the adversary and the defender have a model of this random movement pattern. After network deployment, each virtual object is treated like a real object, as sensors detect it and send the object's information to the destination. The protocol works in rounds. In every round, the node simulating the fake object will randomly pick a sensor node in its neighborhood (including itself) and ask this node to simulate the real object in the next round. In this way, there will be multiple movement patterns similar to that of real objects. Figure 5.2 shows three such virtual objects that simulate real objects.

The source simulation then works as follows. Before deployment, we randomly select a set L of sensor nodes and pre-load each of them with a different *token*. Every token has a unique ID. These tokens will be passed around between sensor nodes to simulate the behavior of real objects. For convenience, we call the node holding a token the *token node*. We also assume that the profile for the behavior of real objects is available for us to create candidate traces.

After deployment, every token node will emit a signal mimicking the signal used by real objects for event detection. This will trigger the event detection process in the local area and generate traffic as if a real event was detected. The token node will

then determine who in its neighborhood (including itself) should run the next round of source simulation based on the profile for the behavior of real objects. The token will then be passed to the selected node. The delivery of the token between sensor nodes will always be protected by the pairwise key established between them.

Note that the simulation requests create additional messages that can help the attacker distinguish real objects from virtual ones. To protect against this, we require that nodes that detect the real object also send an extra message in the system each round. Alternatively, we can attach the requests to the messages to the destination, given that all messages would be received by neighbor nodes.

5.1.1.2.2 Privacy

Assume that the defender can build a model for the behavior of real objects that can always create a useful candidate trace in the network with probability P. In other words, any candidate trace created by the defender in the network will be considered as a valid candidate trace by the attacker with probability P. Let $|S_P|$ be the number of real objects in the network. We can see that the set of candidate locations S_T includes an average of $|S_P| + |L| \times P$ node IDs. As a result, the privacy provided by the source simulation approach can be estimated by

$$b = \log_2 \frac{|S_P| + |L| \times P}{|S_P|} = \log_2(1 + \frac{|L| \times P}{|S_P|}).$$

Since we assume that both the adversary and the defender have similar knowledge about the behavior of real objects, we will usually have $P \approx 1$. In this case, $b \approx \log_2(1 + \frac{|L|}{|S_P|})$.

5.1.1.2.3 Energy consumption

Since there are $|L|$ fake sources simulating the real objects in the network, the com-

munication overhead will be increased by a factor of $\frac{|L|+|S_P|}{|S_P|}$. Note that the source simulation method can be effectively used for real-time applications with privacy protection requirements. We are often required to deliver sensor reports as fast as possible, so we assume that a sensor node will immediately forward the report whenever the channel is free. Hence, Δ is very small which would mean that a sensor would not wait to queue multiple fake packets and forward only one. Let ϵ be the average number of packets required to send an event from a source to destinations that receive it, we get

$$\omega_T \approx \epsilon \times (|L| + |S_P|) \times \frac{T}{\alpha \times \Delta}.$$

This shows that the average communication overhead is approximately increased by a factor of $\frac{|L|+|S_P|}{|S_P|}$ to protect location privacy. This technique features optimal latencies.

5.1.2 Destination Location Privacy Techniques

This section presents two techniques for privacy-preserving routing in sensor networks, the *destination simulation* method and the *backbone flooding* method. The destination simulation method achieves location privacy by simulating destinations at specified locations, and the backbone flooding method provides location privacy by flooding specific portion of the network. Both techniques provide trade-offs between privacy, communication cost, and latency.

5.1.2.1 Destination Simulation

Similar to the source simulation technique, an intuitive solution for destination location privacy would be to confuse the adversary by creating virtual destinations in the network. For this purpose, we propose to create multiple candidate traces towards

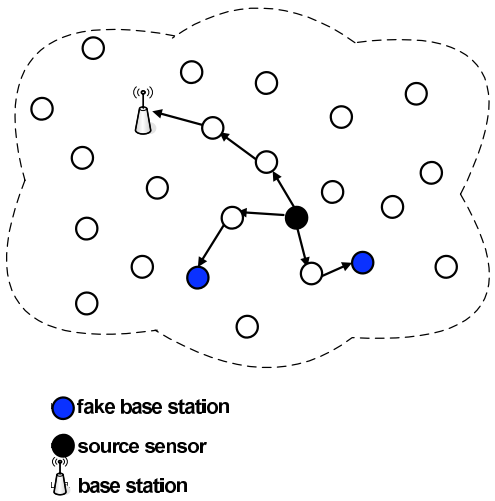


Figure 5.3. Destination simulation approach.

fake destinations in the network to hide the traffic generated for communication between real objects and real destinations. How to determine the number of candidate traces usually depends on the location privacy requirement of an application.

5.1.2.1.1 Protocol Description

In the destination simulation approach, *fake destinations* will be simulated in the field. Each of them will receive traffic similar to the traffic received by a real destination. To achieve this, we make no distinction between fake and real destinations when sensors send data. Figure 5.3 shows the basic idea of the destination simulation approach. In this figure, we show a real destination and fake destinations receiving data from an object.

During deployment, we place real destinations and select locations where fake destinations are to be simulated. A few sensors will be used as the fake destinations. It is also required that each real destination have a fake destination simulated in

its communication range. The reason for doing this will be explained later. Please note that selecting fake destinations that are too close to each other may reduce the location privacy drastically. An attack on one of them can destroy others as well. For example, if an adversary needs to destroy destinations using missile, close destinations may be destroyed in one attack. Similarly, the adversary can physically check and locate close destinations with little additional effort. In practice, the locations of fake destinations should be made as far way from each other as possible. Once the fake destinations are selected after deployment, the sensors should have routing paths to send data to locations where fake destinations are simulated. Many existing routing protocols can be used for such purpose [21, 22, 23].

During the network operations, whenever a source node has the data to be sent to a real destination, the data will be sent to the fake destination simulated in the radio range of this real destination and some other fake destinations that it is supposed to communicate with. Whenever a fake destination receives data, it broadcasts it locally in its radio range. As shown in in Figure 5.3, the blue nodes will broadcast received packet locally in their ranges so that the adversary would believe that a real destination could be in range of any neighbor of the fake destinations. Please note that we also simulate a destination in neighborhood of each real destination. When this simulated destination broadcasts received packet locally, the real destination will receive data.

5.1.2.1.2 Privacy

Let L be the set of locations where destinations are simulated other than those in the neighborhood of real destinations. Let n_b be the average number of neighbors of a sensor. Since each fake destination broadcasts every received packet locally in its range, any node in the communication range of a fake destination can be a real

destination. Assume that fake destinations are simulated at a distance of at least two hops from each other, we can see that the set S_T of candidate nodes includes $(|S_P|+|L|)\times n_b$ node IDs. As a result, the location privacy provided by the destination simulation approach can be estimated by

$$b = \log_2 \frac{(|S_P| + |L|) \times n_b}{|S_P|} = \log_2(1 + \frac{|L|}{|S_P|} \times n_b).$$

5.1.2.1.3 Energy consumption

Since there are $|L|$ places other than those in neighborhood of real destinations where the fake destinations are present in the network, the communication overhead will be approximately increased by a factor of $\frac{|L|+|S_P|}{|S_P|}$. In destination simulation approach, we consider the fake destinations to be randomly located in the network. When we need to create a large number of fake destinations to meet the high location privacy requirement, the destination simulation approach can be very expensive. The reason is that a lot of extra communication is wasted during the routing of packets to *randomly picked* fake destinations. To address this problem, we will discuss a *backbone flooding* technique in the next section.

5.1.2.2 Backbone Flooding

In the *backbone flooding approach*, we will send packets to a *connected* portion of the network, *the backbone*, instead of sending them only to a few *randomly scattered* destinations. Only the sensors, *the backbone members*, that belong to this backbone need to flood the packets so that all the sensors in the communication range of the the backbone can receive them. The real destinations are located in the communication range of at least one backbone member. Existing studies have focused on finding the minimal number of sensors that are needed to flood a packet so that entire network

can receive it [24, 25]. In our case we need to flood the packets to cover an area required to achieve the desired level of location privacy.

Certainly, an adversary with global eavesdropping capability will be able to point out locations of real destinations if he can eavesdrop on the network before the backbone formation. However, this will be very difficult for an attacker since the network deployment is often well supervised. We thus assume that the backbone is created soon after the sensor network is deployed and that the adversary does not eavesdrop on the network until the backbone is created.

5.1.2.2.1 Protocol Description

In the backbone flooding approach, we create a backbone network of $|L|$ backbone members in the given network such that the destination is within the range of at least one backbone member. The backbone network formed should cover maximum area possible to increase the search area for the adversary. However, this is an NP-hard problem [24, 25].

We present an approximation algorithm for this problem. When we say that a sensor v *covers* some neighbors, we mean that v is responsible for delivering received packets to those neighbors via local broadcast in its range. The backbone formation algorithm terminates when the backbone covers required number of sensors for desired level of privacy.

We first consider the backbone formation algorithm in case where there is only one real destination in the network. We then present simple extension for case of multiple destinations. Input to the algorithm are b (amount of privacy required) and the minimum number of uncovered neighbors m a sensor should cover to be eligible to become a backbone member. m is called the *mincover*. For a given sensor i, let $max_i(m)$ be a function that takes the number of uncovered sensors each neighbor of

i can cover and outputs the ID of the neighbor that covers the maximum number of uncovered sensors if this maximum number is greater than m. It also returns the number of sensors the newly identified sensor covers. It outputs $\perp$ otherwise. The algorithm produces a connected backbone network at each step.

We assume that each sensor has the list of its neighbors. Initially the set L that has the IDs of the backbone members is empty. We can begin with adding a sensor which has the real destination in its range. We now describe this algorithm from perspective of a sensor that sends an *election* message and then from the perspective of a sensor that receives this message. The pseudo code of the algorithm is presented in Algorithm 1

A new sensor v added to the set L will send an *election* message to carry out an election to find the number of uncovered sensors each neighbor can cover. If $max_v(m)$ outputs a valid sensor ID and the coverage of this node, the ID of this node will be added to L. The newly added sensor will then execute the same algorithm. If $max_v(m) = \perp$, v will collaborate with existing nearby backbone members to find a usable sensor. The backbone is a tree structure with backbone members as the tree nodes. Collaboration will mean getting information from parent in order to find a sensor node with at least the given mincover. This collaboration could continue to next level of ancestors if such a node is not known to immediate parent. This process, the backtracking process, continues until a sensor meeting the required constraints is found. If a sensor that can cover at least m uncovered sensors in unavailable, a sensor which covers maximum number of uncovered sensors is selected through backtracking. The beginning and termination of the backtracking process depends on the value of m. A value of $m \leq 1$ would mean that backtracking would start only if v cannot find any neighbor that can cover at least one uncovered sensor. If m has a value greater than the number of neighbors any sensor could have, then the

backtracking process would ensure that the sensor which covers the maximum number of uncovered sensors is selected. Intuitively, this would mean that an increase in the value of mincover m would help in covering more sensors with the help of less backbone members. However, more energy will be consumed to form a backbone for a large m due to more backtracking steps.

Each uncovered sensor that receives an *election* message sends an *accept* message to the sender. This sender then becomes the node's parent. When such *accept* messages are being successfully sent, all the other sensors that can overhear these messages reduce the number of sensors they can cover by the number of unique messages heard. After this is done, each sensor sends to its parent the number of sensors it can cover in a *coverage* packet. Note that we start the backbone formation from one of the neighbor of the real destination. To prevent the destination from being located at the end points of the backbone, the algorithm should be executed twice from the same neighbor of the destination. The input parameter backbone size to these two executions should add up to original input value. For example, if we need b bits of privacy, we randomly choose a pair of values $\{b_1, b_2\}$ such that $b_1 + b_2 = b$. The backbone formation algorithm is then executed twice from the same neighbor of destination once with input b_1 and then with input b_2.

Once the backbone network is setup, the source sensor just needs to communicate with the nearest member of the backbone network. After a member receives the data, it will flood the data in the backbone. The real destination can always receive every packet. It is also clear to see that from the attacker's point of view, the real destination could be anywhere in the communication range of the backbone.

Let us now turn our attention to the case where there are multiple real destinations in the network either to protect applications from single point of failure or to save energy by only sending data to the nearest destination. To protect the location

information of all the destinations, we provide a simple extension to the algorithm that we discussed above. We generate a number of random numbers $b_1, b_2, b_2, \cdots, b_c$ such that $b = b_1 + b_2 + ... + b_c$ where each b_i is an input that corresponds to a destination $dest_i$. The backbone creation process is executed in sequence for each destination $dest_i$ with the corresponding parameter b_i.

A source may need to send data to multiple destinations. when backbones are very to each other such that sum of the hops between a source and each backbone is greater than sum of the hops required to connect the backbones, one may think that backbones should be connected to each other through a path to reduce energy consumption. This may not always be true and depends on locations of destinations. Thus, if backbones are far away, the energy consumption may increase after they are connected. Further, connecting backbones may not provide the privacy specified by the user because connecting them may need more hops which may cover more sensors than needed. Hence we do not connect the backbones and, the sources consider them as independent entities.

5.1.2.2.2 Privacy

The algorithm for constructing the backbone takes b as one of the inputs and covers required number sensors to achieve privacy. The privacy achieved by this technique is thus b.

5.1.2.2.3 Energy consumption

Let h be the average hop distance from a source to the nearest backbone member. Since there are $|L|$ backbone members, every event received by the backbone will be re-broadcasted by $|L|$ sensors. Thus the communication overhead for delivering one

event will be $h+|L|$ packets. Since there are $|S_A|$ sources and each source will trigger an event every α time intervals, the overall communication cost can be estimated by

$$\omega_T = (h + |L|) \times |S_A| \times \frac{T}{\alpha \times \Delta}.$$

5.1.2.2.4 Efficiency of backbone creation

In the proposed algorithm, a newly elected backbone member will send an election message and select a neighbor that can cover the maximum number of uncovered neighbors. When $m = 1$, it will backtrack if and only if none of its neighbors can cover at least one uncovered node. Hence when $m = 1$, the algorithm will rarely need to backtrack and require only $O(|L|)$ messages.

The algorithm will require $O(|L|^2)$ messages when $m >> n_b$. This is because a newly elected backbone member will need to collaborate with all other backbone members to find a sensor (the new backbone member) that can cover maximum uncovered sensors. Detailed simulation studies are done in Section 6.1 to show the cost for different m.

5.1.2.2.5 Self-healing backbone

A physical attack or failure of some members of the backbone can prevent data packets from reaching the real destination. As shown in Figure 5.4, failure of a backbone member will partition the backbone into multiple segments and will prevent packets to reach all the backbone members. If the real destination is isolated from other backbone members, the network would be useless because the real destination will fail to receive the event reports. To overcome this problem, we need to provide means to heal the backbone whenever it is partitioned.

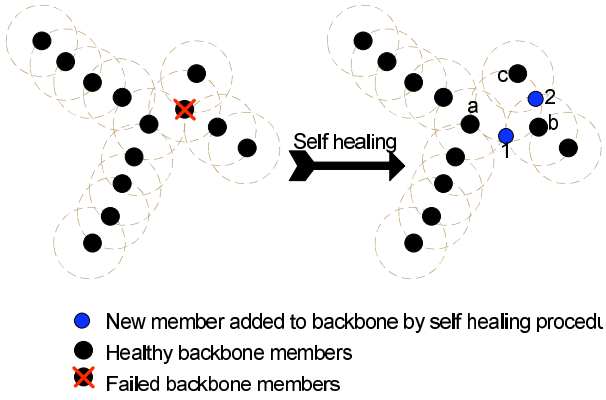


Figure 5.4. Self Healing Algorithm.

Whenever a backbone member cannot communicate with its neighbor backbone member, it should establish a path with a backbone member from the other part of the backbone. This will reduce the number of isolated segments by one. If we keep doing this, we will eventually get a connected backbone again. The number of backbone members may change depending on the number of members not working and the number of new members added for connecting the segments.

We now present the self-healing algorithm in detail. Each member has a list of all the immediate neighbor backbone members. Whenever a packet is to be sent to the backbone members and a member cannot forward packets to its neighbor backbone member, it (initiator) concludes that the backbone is partitioned. This makes the *initiator* initiate the healing process.

The algorithm needs to find all the backbone members that cannot communicate with at least one of their neighboring backbone members. To achieve this, the initiator will broadcast a healing request that can travel up to h hops, where h is a configurable parameter which depends on the kind and intensity of attack expected. For more destructive attacks we need higher value of h to reconnect the backbone. To guarantee connectivity, h can be set to a value that is greater than diameter of the network which

would mean that the packet should be flooded in the entire network. The healing request will include the identification of the data packet that the initiator could not forward to its neighbors. The healing request will be processed by sensors which cannot communicate with at least one neighbor backbone member or sensors (non-backbone members) which are neighbors of a failed backbone member. To save energy, neighbors of a failed backbone member should coordinate so that only a small number of neighbors will respond to the healing request. Among the neighbors of the backbone member that have a real destination in its range, the neighbors of the destination are given priority to respond to the healing request in case the backbone member fails. Sensors that process healing request packets are called responders. Healing request packets should not be processed by sensors that have already received the data packet with the same identification as the identification in the healing request.

All responders should reply to the initiator. The reply will travel through the shortest path that is created when the healing request travels h hops from the initiator using the path setup algorithms such as the one in [23]. The initiator connects one of the closest responders through this shortest path. The initiator sends the data packet to the responder that it failed to send earlier due to backbone failure. The responder forwards this data packet to all the backbone members it can reach. This will reduce the number of backbone segments by at least one. The responder then acts as the initiator and initiates the same healing process. The process continues until the backbone is connected.

Figure 5.4 shows an example of the healing process. When one of the backbone member fails, node a realizes this when it cannot send packet to the failed node. Node a initiates the healing process. Node 1, the neighbor of the failed node, replies to the healing request. Similarly, Nodes b and c will reply because they find that at least one of their neighbor backbone members has failed. Initiator a will connect to

node 1 as it is the closest responder. Node 1 will then act as the initiator. Nodes b and c will respond, and node 1 will connect to node b as it is the closest responder. Similarly, node b will now act as initiator and connected to node c via node 2 because node c is the only responders. Finally, the healing process will stop because node c wont find any responder. We get a fully connected backbone.

Algorithm 1 Backbone Construction

Require: Each node has list of its neighbors

```
1: procedure BACKBONE(b, m)
2:     TotalCoverage ← 1                                   ▷ first sensor in the set L
3:     Id ← GetMyId()
4:     Leader ← −1
5:     LocalCoverage ← GetNeighborCnt()
6:     while true do
7:         Msg ← GetNextMsgFromQueue()
8:         if TotalCoverage ≥ 2^b then
9:             EXIT
10:        end if
11:        if MsgType = NewMemberSelection then
12:            if CheckNewMemberId(Msg) = Id then
13:                DestId ← GetDestId(Msg)
14:                SendElectionMsg(Id, DestId)
15:                CollectVotes(Id, DestId)
16:                CollectCoverageInfo(Id, DestId)
17:                (ResultId, Coverage) ← Max_Id(m)
18:                if Valid(ResultId) = true then
19:                    TotalCoverage ← TotalCoverage + Coverage
20:                else
21:                    (ResultId, Coverage) ← CollaborateRecursivelywithParent(Coverage)
22:                    if Valid(ResultId) = true then
23:                        TotalCoverage ← TotalCoverage + Coverage
24:                    else
25:                        EXIT                ▷ Cannot find more sensors to cover
26:                    end if
27:                end if
28:                NotifyNewMember(ResultId, TotalCoverage)
29:            end if
30:        else if MsgType(Msg) = ElectionRequest then
31:            if Leader = −1 then
32:                SenderId ← GetSenderId(Msg)
33:                SendVote(SenderId)
34:            end if
35:        else if MsgType(Msg) = CoverageInfoRequest then
36:            SendCoverageInfo(LocalCoverage)
37:        else if MsgType(Msg) = AcceptMessage then
38:            LocalCoverage ← LocalCoverage − 1
39:        end if
40:    end while
41: end procedure
```

CHAPTER 6

SIMULATION MODEL

6.1 Simulation Evaluation

In this section, we use simulation to evaluate the performance of our techniques in terms of energy consumption and latency.

The Panda-Hunter example was introduced in [2], and we will use terminology from this example to describe our simulation. In this application, a sensor network is deployed to track endangered giant pandas in a bamboo forest. Each panda has an electronic tag that emits a signal that can be detected by the sensors in the network. We include 5,093 sensor nodes distributed randomly in a square field of 1000×1000 meters to monitor the location of pandas in the network. A base station is the destination for all the real traffic. Each sensor node can communicate with other sensor nodes in a radius of 50 meters, while an electronic tag attached to a panda can emit radio signals that can reach sensor nodes within 25 meters. We noticed that, on average, each sensor node has 40 neighbors and that the presence of any panda will be detected by 10 sensor nodes on average. For source location privacy techniques, we assume that the base station is located at the center of this field. For destination location privacy techniques, we randomly choose the locations of fake base stations in the field.

The proposed techniques assume a routing protocol for sensor networks, though the choice of routing protocol does not affect our results. For simplicity, we adopt a simple and widely-used routing method in many studies such as [23]. In this method, the routing paths are constructed by a beacon packet from the base station. Each

node, on receiving the beacon packet for the first time, sets the sender of the beacon packet as its parent. In this way, each node will likely select a parent that is closest to the base station.

For the purpose of simulation, we assume that the network application only needs to detect the locations of pandas and always wants to know the most recent locations. We thus have every sensor node drop a new packet if it has already queued an identical packet that was generated from the same event.

In our simulation, we assume that the adversary has deployed a network to monitor the traffic in the target network. Specifically, he is able to locate every sensor node in the target network and eavesdrop every packet this node delivers. Though the adversary may face some engineering problems in developing methods to collect these observations from its network, we do not believe that this will be a very difficult issue to address. For simplicity, we assume the adversary can always reliably collect all the observations in the network.

We ran each simulation for $6,000$ intervals of τ seconds each. We ran simulations for different lengths of queue. We selected the initial locations for pandas randomly. In the experiments, the tag attached to a panda emits a signal, which will generate an event, for detection at a rate of one per $10 \times \tau$ seconds. In addition, every panda moves from its current location (x, y) to a random location $(x \pm a_1, y \pm a_2)$ every $10 \times \tau$ seconds, where a_1 and a_2 are two random values uniformly selected between 0 and 60.

We also compare our techniques with an optimal technique that would follow a Steiner tree to route packets to the base stations. We use approximation algorithm from [16] for approximating the weight of a Steiner tree. Please see section 4.1 for description of this algorithm.

CHAPTER 7

EVALUATION

7.1 Periodic Collection

The analysis in Section 5.1 has already shown that the periodic collection method achieves optimal location privacy. In addition, the communication overhead in the network remains constant and is independent of both the number of pandas and their patterns of movement. Hence, the focus of our simulation evaluation is on the latency and the packet drop rate when there are multiple pandas in the field. The time interval in periodic collection is $\Delta = \tau$.

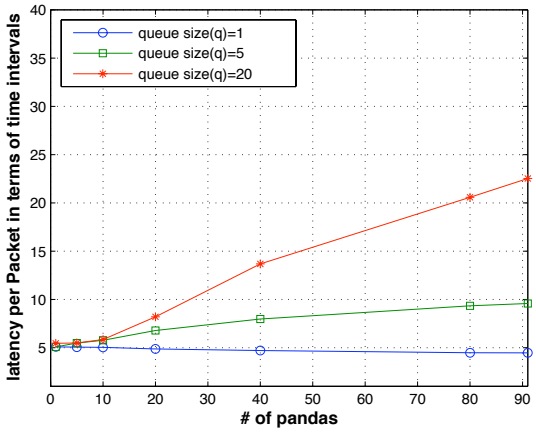


Figure 7.1. Latency vs. Number of pandas.

Figure 7.1 shows the latency of packet delivery when there are multiple pandas in the field. We can see that as the number of pandas increase, the latency increases. This is because the nodes close to the base station receive multiple reports at the same time, therefore requiring them to buffer packets. When the number of pandas grows

too large, the buffered packets start being dropped due to the limited size of queue, while the latency of the packets that do arrive at the base station becomes stable after a certain point. When the queue size q decreases, packets traveling long distances have a high probability of getting dropped, making the latency of the packets that do arrive at the base station smaller. This can be seen by a drop in the latency for smaller values of q.

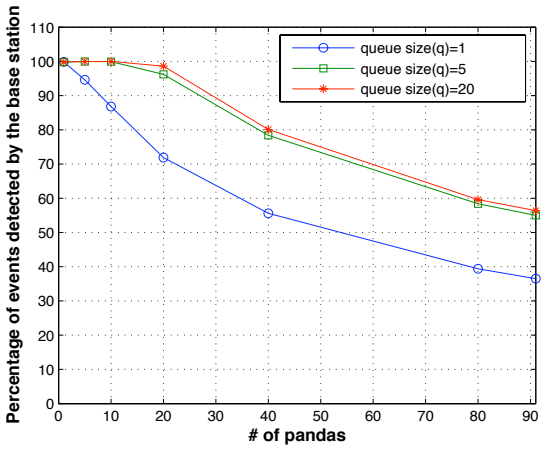


Figure 7.2. Percentage of events detected vs. Number of pandas.

Figure 7.2 shows the percentage of the detected events received by the base station. We can see that the percentage of events received decreases when there are more pandas in the field. Increasing q will certainly increase the percentage of the events forwarded to the base station. However, after a certain point, having a larger q will not offer significant benefit in terms of the packet drop rate. For example, increasing the queue size from 5 to 20 does not help much in forwarding more events to the base station. On the other hand, we have already shown in Figure 7.1 that increase q will significantly increase the latency of packet delivery. This is undesirable in some cases. As a result, we believe that q should typically be configured as small

as possible while still meeting the requirements of the event drop rate. Overall, the results in Figure 7.1 and Figure 7.2 give a guideline for configuring the queue size q to meet various requirements.

7.2 Source Simulation

According to the analysis in Section 5.1, the location privacy achieved by source simulation is determined by the number of virtual sources simulated in the network. Thus, the focus of our simulation evaluation is on how much communication cost we have to pay to achieve a given level of location privacy. We use these results to illustrate the efficiency of the proposed technique.

During the simulation, we assume that there is only one panda in the network. Multiple fake pandas are created and simulated in the field. The initial positions of the fake pandas are randomly selected. In addition, we assume that the sensor network is deployed to handle real-time applications. In other words, whenever a sensor node receives a packet, it will forward it to the next hop as soon as possible. The time interval in periodic collection is $\Delta = \tau$ and in source simulation it is $\Delta = \frac{\tau}{10}$. Thus source simulation will forward packets ten times faster than the periodic collection method. We set P to 1, which means that the adversary has the same knowledge about the panda behavior as the defender.

Figure 7.3 shows the communication overhead involved in our source simulation method to achieve a given level of privacy. We can see that the communication overhead increases as the location privacy requirement increases. This figure also includes the performance of other approaches for further comparison, which we will explain below.

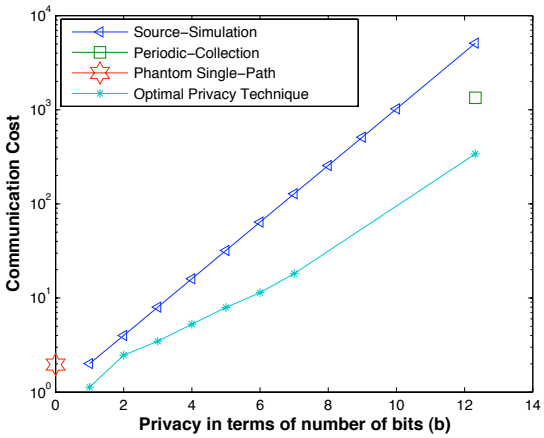


Figure 7.3. Comparison between different source location privacy schemes.

7.3 Comparison Of Source Location Privacy Techniques

We now compare the proposed source location privacy approaches in this paper with previous privacy-preserving techniques–in particular, the phantom single-path routing technique [2]. We focus on the location privacy achieved and the communication overhead introduced in the following comparison. The simulation result is shown in Figure 7.3. The performance of the phantom single-path routing is represented by a single point at the left-bottom corner of the figure, and the performance of the periodic collection method is represented by a single point on the right part of the figure.

In terms of privacy, we have already shown that none of the previous methods (including phantom single-path routing) can provide location privacy under the assumption of a global eavesdropper. In contrast, both of our methods provide location privacy against a global eavesdropper. The periodic collection method provides the highest level of privacy and is suitable for applications that collect data at a low rate and do not require real-time data delivery, while the source simulation method can

support real-time applications with practical trade-offs between privacy, communication overhead, and latency.

We compare the communication overheads through simulation. Figure 7.3 shows the communication costs involved in different methods. The simulation results are not surprising. The phantom single-path routing technique only introduces small communication overhead, while the periodic collection method involves significant but constant communication cost for a given period of time. The source simulation method is in the middle of these two schemes; it can provide practical trade-offs between privacy and communication cost. We notice that in the figure, the periodic collection method requires less communication overhead to achieve privacy of around $b = 12$ bits when compared with the source simulation method. The reason is that the source simulation method is configured to support real-time applications with time interval Δ ten times smaller than its value in periodic collection.

We can also see performance of techniques in comparison to optimal technique. For maximum privacy the periodic collection technique consumes more energy than an optimal technique because in periodic collection technique each sensor emits a packet every $\Delta = \tau$ seconds, however, in optimal technique each sensor emits a packet once every $10 \times \Delta = 10 \times \tau$ seconds as is the case with a real source (α is 10).

7.4 Destination Simulation

The analysis in Section 5.1 has shown that the location privacy achieved by destination simulation approach depends on the number of fake base stations simulated in the network. The packets generated by sources are sent to all fake and real base stations. Hence, the focus of our simulation evaluation is on the latency and the packet drop rate when there are multiple base stations in the field.

Figure 7.5 shows the latency of packet delivery when there are multiple fake base stations in the field. We can see that as the number of fake base stations increases, the latency increases. This is because more base stations will cause more traffic in the network and thus more packets to be buffered. When the number of fake base stations grows too large, the buffered packets start being dropped due to the limited size of queue, while the latency of the packets that do arrive at the base station becomes stable after a certain point. When the queue size q decreases, packets traveling long distances have a high probability of getting dropped, making the latency of the packets that do arrive at the real base station smaller. This can be seen by a drop in the latency for smaller values of q.

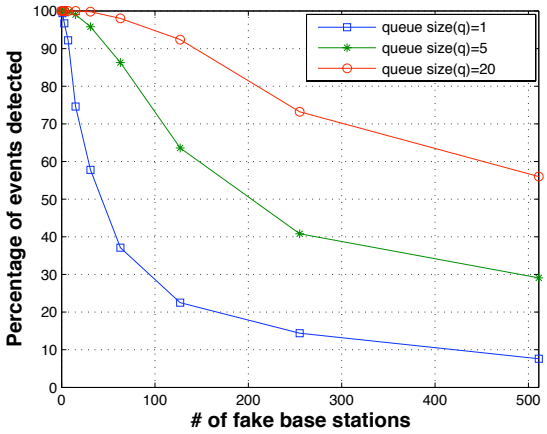


Figure 7.4. Number of fake base station vs. percentage of events detected.

Figure 7.4 shows the percentage of the detected events received by the real base station. We can see that the percentage of events received decreases when there are more fake base stations in the field. Increasing q will certainly increase the percentage of the events forwarded to the base station. However, we have already shown in Figure 7.5 that increasing q will significantly increase the latency of packet

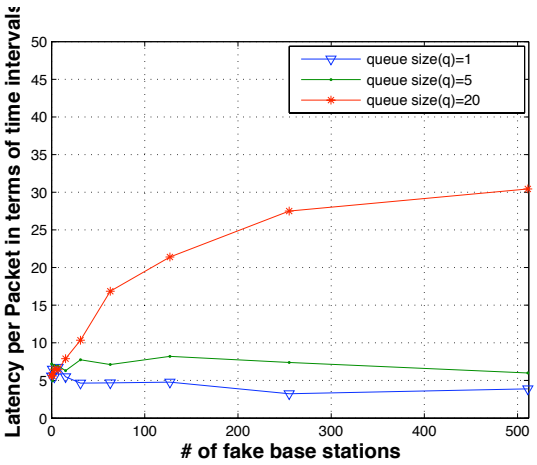


Figure 7.5. Effect of number of fake base stations on latency.

delivery. This is undesirable in some cases. As a result, we believe that q should typically be configured as small as possible while still meeting the requirements of the event drop rate. Overall, the results in Figure 7.4 and Figure 7.5 give a guideline for configuring the queue size q to meet various requirements.

7.5 Backbone Flooding

From the analysis in Section 5.1, we see that the location privacy achieved by backbone flooding approach will increase with the increase in the number of backbone members. Packets generated by a source are sent to all the backbone members. Hence, the focus of our simulation evaluation is on the latency, the packet drop rate, and energy required for backbone creation for different sizes of backbone.

Figure 7.6 shows that the increase in backbone size will cause more energy to be consumed. We can also see that increase in the value of parameter *mincover* will lead to more backtracking in backbone creation and hence utilize more energy.

Figure 7.7 shows the latency of packet delivery as the size of backbone increases. We can see that as the size of backbone increases, the latency increases. This is

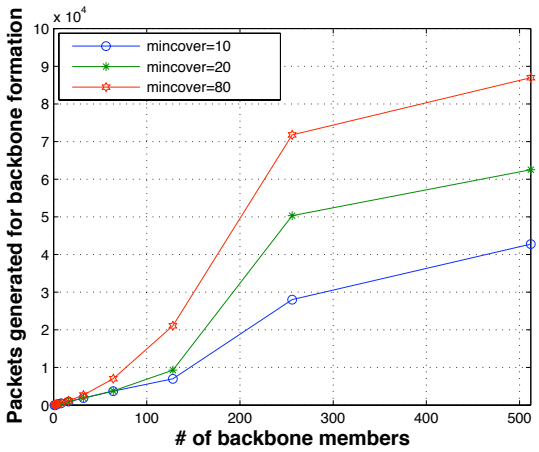


Figure 7.6. Energy consumed for creation of backbone.

because increase in the size of backbone will cause increase in the packets in the network causing buffering of packets which leads to increase in latency.

Figure 7.8 shows the percentage of the detected events received by the base station. We can see that the percentage of events received decreases when there are more backbone members in the field. Increasing q will certainly increase the percentage of the events forwarded to the base station. However, after a certain point, having a larger q will not offer significant benefit in terms of the packet drop rate. For example, increasing the queue size from 5 to 20 does not help much in forwarding more events to the base station. On the other hand, we have already shown in Figure 7.7 that increase q will increase the latency of packet delivery. This is undesirable in some cases. As a result, we believe that q should typically be configured as small as possible while still meeting the requirements of the event drop rate. Overall, the results in Figure 7.7 and Figure 7.8 give a guideline for configuring the queue size q to meet various requirements.

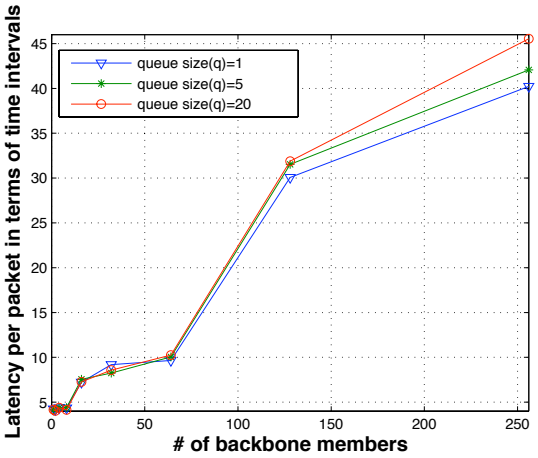


Figure 7.7. Effect of backbone size on latency.

7.6 Comparison Of Destination Location Privacy Techniques

We now compare the proposed destination location privacy approaches in this paper. We focus on the location privacy achieved and the communication overhead introduced by each technique in the following comparison. The simulation result is shown in Figure 7.9.

In terms of privacy, we have already shown that none of the previous methods can provide location privacy under the assumption of a global eavesdropper. In contrast, both of our methods provide location privacy against a global eavesdropper.

We compare the communication overheads through simulation. Figure 7.9 shows the communication costs involved in different methods. Both the techniques can provide practical trade-offs between privacy and communication cost. We notice that in the figure, the backbone flooding technique consumes minimum energy. The reason is that this method does not incur much cost to generate the traffic towards the fake base stations. A single re-broadcast of packets in the backbone creates many fake base stations.

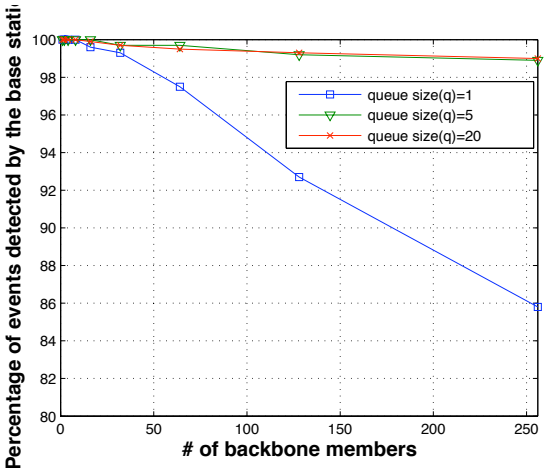


Figure 7.8. Backbone size vs. percentage of events detected by base station.

Both the optimal and backbone flooding techniques are stair graphs because a single observation can be received by all neighbors of a sensor due to local broadcast by a fake base station. All these neighbors will be considered by the adversary to be equally likely to be in neighborhood of a real base station. Hence, energy consumption will remain same for privacy in the range of 1 to $log_2(n_b)$ for a base station.

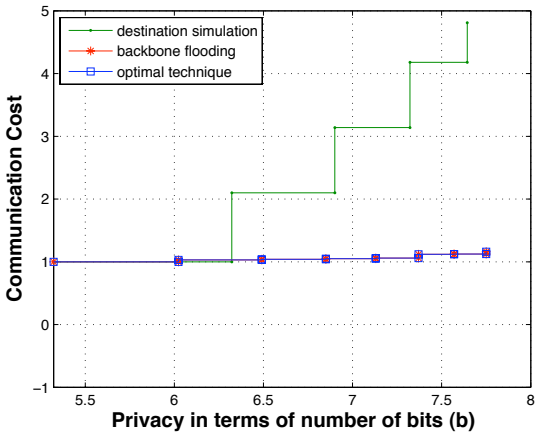


Figure 7.9. Comparison of destination location privacy techniques.

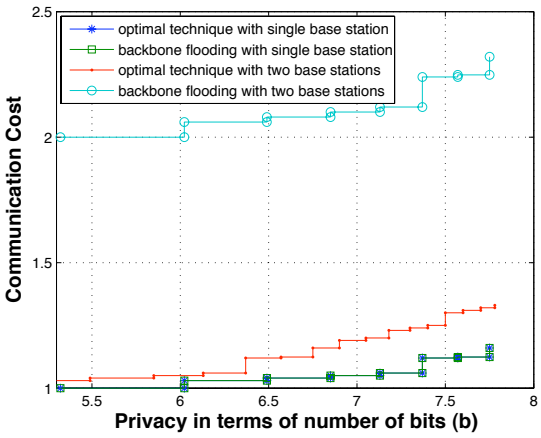


Figure 7.10. Effect of multiple base stations.

In the figure 7.10 we see the effect of multiple real base stations on communication cost for the desired level of location privacy. Each source sends every packet to both the base stations. We have considered the best case for optimal technique where the base stations are adjacent to each other. In other words, locations of real base stations are such that a set of connected nodes have these two base stations in its range and can cover at least $2^b \times 2$ sensors required for b bits of privacy. Communication cost of backbone flooding doubles when the number of base stations double because by design source communicates with each backbone independently. However, in the best case, an optimal technique would cause a small increase in communication cost because both the backbones are adjacent and source just needs to send packets to one backbone. The second backbone receives data from the first backbone which is adjacent to it.

CHAPTER 8

CONCLUSION AND FUTURE WORK

8.1 Conclusions

Prior work that studied location privacy in sensor networks had assumed that the attacker has only a local eavesdropping capability. This assumption is unrealistic given a well-funded, highly-motivated attacker. We formalized the location privacy issues under the model of a global eavesdropper, and show the minimum average communication overhead needed for achieving certain privacy. We also presented two techniques to provide location privacy to objects and destinations against a global eavesdropper. Analysis and simulation studies show that they can effectively and efficiently protect location privacy in sensor networks.

8.2 Future Work

There are a number of directions that worth studying in the future. In particular, in this paper, we assume that the global eavesdropper will not compromise sensor nodes; he only performs traffic analysis without looking at the content of the packet. However, in practice, the global eavesdropper may be able to compromise a few sensor nodes in the field and perform traffic analysis with additional knowledge from insiders. This presents interesting challenges for both of our approaches. In addition, we are also interested in the implementation of our methods in real sensor platforms and the experimental results from real sensor applications.

REFERENCES

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] P. Kamat, Y. Zhang, W. Trappe, and C. Ozturk, "Enhancing source-location privacy in sensor network routing," in *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS)*, June 2005, pp. 599–608.

[3] K. Mehta, D. Liu, and M. Wright, "Location privacy in sensor networks against a global eavesdropper," in *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, 2007.

[4] C. Ozturk, Y. Zhang, and W. Trappe, "Source-location privacy in energy-constrained sensor network routing," in *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks (SASN)*, October 2004, pp. 88–93.

[5] Y. Ouyang, Z. Le, G. Chen, J. Ford, and F. Makedon, "Entrapping adversaries for source protection in sensor networks," in *Proceedings of the International Symposium on on World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, June 2006, pp. 23–34.

[6] Y. Yang, M. Shao, S. Zhu, B. Urgaonkar, and G. Cao, "Towards event source unobservability with minimum network traffic in sensor networks," in *Proceedings of ACM Conference on Wireless Network Security (WiSec)*, 2008.

[7] M. Shao, Y. Yang, S. Zhu, and G. Cao, "Towards statistically strong source anonymity for sensor networks," in *Proceedings of IEEE INFOCOMM*, 2008.

[8] J. Deng., R. Han, and S. Mishra, "Enhancing base station security in wireless sensor networks," 2003. [Online]. Available: citeseer.ist.psu.edu/deng03enhancing.html

[9] J. Deng, R. Han, and S. Mishra, "Decorrelating wireless sensor network traffic to inhibit traffic analysis attacks," *Elsevier Pervasive and Mobile Computing Journal, Special Issue on Security in Wireless Mobile Computing Systems*, vol. 2, pp. 159–186, April 2006.

[10] Y. Jian, S. Chen, Z. Zhang, and L. Zhang, "Protecting receiver-location privacy in wireless sensor networks." May 2007, pp. 1955–1963.

[11] J. Hill, M. Horton, R. Kling, and L. Krishnamurthy, "The platforms enabling wireless sensor networks," *Commun. ACM*, vol. 47, no. 6, pp. 41–46, 2004.

[12] BlueRadios Inc., "Order and price info." http://www.blueradios.com/orderinfo.htm, accessed in February 2006.

[13] A. Savvides, C. Han, and M. Srivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *Proceedings of ACM MobiCom*, July 2001, pp. 166–179.

[14] D. Niculescu and B. Nath, "Ad hoc positioning system (APS) using AoA," in *Proceedings of IEEE INFOCOM*, April 2003, pp. 1734–1743.

[15] B. Bollobas, D. Gamarnik, O. Riordan, and B. Sudakov, "On the value of a random minimum weight steiner tree," pp. 187–207, 2004. [Online]. Available: citeseer.ist.psu.edu/482903.html

[16] H. Takahashi and A. Matsuyama, "An approximate solution for the steiner problem in graphs," *Math. Japonica*, vol. 24, pp. 573–577, 1980.

[17] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS)*, November 2002, pp. 41–47.

[18] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *IEEE Symposium on Security and Privacy (S&P)*, May 2003, pp. 197–213.

[19] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS)*, October 2003, pp. 52–61.

[20] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and D. Tygar, "SPINS: Security protocols for sensor networks," in *Proceedings of Seventh Annual International Conference on Mobile Computing and Networks (MobiCom)*, July 2001.

[21] D. B. Johnson, D. A. Maltz, Y. Hu, and J. G. Jetcheva, Internet Draft, February 2002, draft-ietf-manet-dsr-07.txt.

[22] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (AODV) routing," Internet Draft, February 2003, draft-ietf-manet-aodv-13.txt.

[23] J. Deng, R. Han, and S. Mishra, "Intrusion tolerance and anti-traffic analysis strategies for wireless sensor networks," in *2004 International Conference on Dependable Systems and Networks (DSN)*, June 2004.

[24] H. Gupta, Z. Zhou, S. Das, and Q. Gu, "Connected sensor cover: self-organization of sensor networks for efficient query execution," *IEEE/ACM Trans. Netw.*, vol. 14, no. 1, pp. 55–67, 2006.

[25] V. Paruchuri, A. Duressi, M. Duressi, and L. Barolli, "Routing through backbone structures in sensor networks," in *Proceedings of the 11th International Conference on Parallel and Distributed Systems (ICPADS)*, 2005.

BIOGRAPHICAL STATEMENT

Kiran K. Mehta was born in Pune, India, in 1981. He received his B.Sc. and M.Sc. degrees in Computer Science from The University of Pune, India, in 2002 and 2004 respectively. He received his M.S. degree in Computer Science from The University of Texas at Arlington in 2008. From 2004 to 2006, he worked with Symantec Corporation in Pune, India in Cluster Server group. His current research interest is in the area of security and privacy in sensor networks. He is a student member of IEEE.