# Barrier Coverage Construction without Barrier Breach in Wireless Sensor Networks

Senior Project

In partial fulfillment of the requirements for
The Esther G. Maynor Honors College
University of North Carolina at Pembroke

By

Luke L. Fleming
Department of Mathematics and Computer Science
May 7, 2021

| | |
|---|---|
| Luke L. Fleming<br>Honors College Scholar | Date |
| Joong-Lyul Lee, Ph.D.<br>Faculty Mentor | Date |
| Joshua Kalin Busman, Ph.D.<br>Senior Project Coordinator | Date |

## Acknowledgements

I would like to thank everyone at UNCP that helped me get to this point, including not only my mentor Dr. Lee, but also all the faculty and staff that helped me along the way. I have truly had a great experience at UNCP, and I hope that continues throughout my time at graduate school.

## Abstract

In wireless sensor networks, the barrier coverage detects objects crossing a protected area or monitors an area of interest. It is an important application in the wireless sensor networks. In such a wireless sensor network application, sensor nodes are randomly deployed along the boundary of the monitoring area due to cost issues and construct multiple barrier coverage in order to maximize the network life time. These multiple barriers are operated according to the sleep wakeup schedule. In this application, a new security problem which is the barrier breach problem occurs in the sleep wakeup schedule In this work, we propose a new barrier coverage construction algorithm without a barrier breach.

Barrier Coverage Construction without Barrier Breach in Wireless Sensor Networks

## Introduction

Wireless Sensor Networks (WSNs) have been utilized for various applications in various fields by diversity of technology due to the development of technology and the low price of devices. As one of these applications, the sensing technology of a device can detect someone's entry and exit in a protected area. Intrusion detection and border surveillance systems are critical applications in military operations and homeland security. Acheving barrier coverage is important in the construction and monitoring of the area of interest seamlessly (Kumar et. al., 2007a). The sensor placement strategy directly affects the barrier coverage. Forming the most ideal barrier coverage is to place manually the area of interest along the border line. However, this is almost impossible to apply to many applications, such as areas in hostile environments and hard-to-reach areas. Therefore, sensor nodes are randomly deployed along the boundary of the monitoring area by airplane, and it constructs multiple barrier coverages in order to maximize the network lifetime. These multiple barriers are operated according to the sleep-wake up schedule because of network lifetime and battery lifetime issues. In Figure 1, two-barrier coverage is constructed by a sleep-wake up schedule. Barrier B1 is working to monitor in order to detect intruders at time T1, and barrier B2 is in a sleep state in order to save the battery energy at time T1. After battery depletion of barrier B1, barrier B2 starts to work for monitoring instead of barrier B1 at time T2. In figure 1 below, we can see that intruder P1 and P2 can enter the protected area without detection. When B1 is working, P1 and P2 can move to location L1, and when B2 is working at time T2, P1 and P2 can enter these protected areas without detection. However, P3 and P4 cannot enter without detection. This new security problem is the barrier breach problem which occurs in the sleep-wake up schedule (Kim et. al., 2012). In this research, we propose a new barrier coverage construction algorithm which does not have a barrier breach using logical vector, which was never discussed before. By using a logical vector (Fidge, C. 1987, Saipulla et. al., 2013), this algorithm can confirm that there exists a barrier breach. Therefore, this algorithm can construct a barrier coverage without barrier breach.

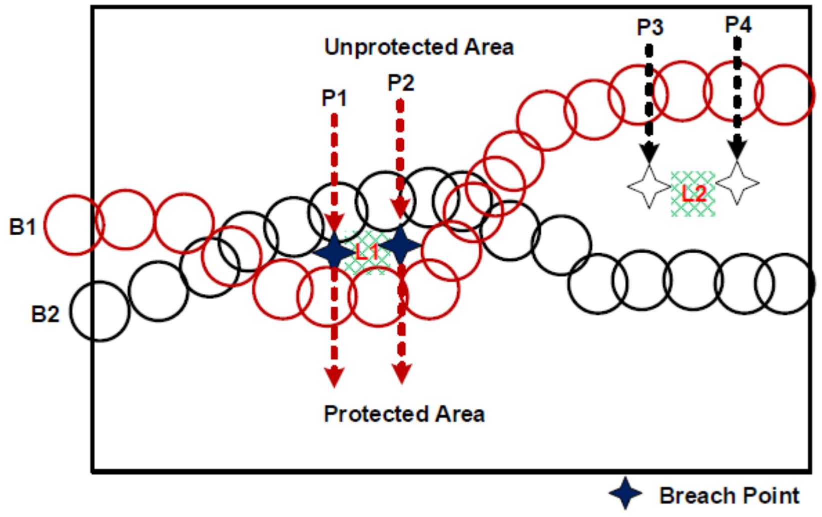

Figure 1: An example showing a barrier breach problem in barrier coverage

## Literature Review

Wireless Sensor Networks (WSNs) are, “interconnected sensor nodes that communtcate wirelessly to collect data about the surrounding environment. (Patil & Chen, 2017)” Common uses for WSNs include monitoring motion, temperature, humitity, sound, detection of intruders, and air pressure. For our application, we are mainly focusing on the detection of intruders. The main problem with WSNs are that the sensor nodes in the network are limited in power, computational capabilities, and memory. This means that these sensor nodes are prone to failure. The main failure we will focus on is a dead battery; we are trying to extend the lifetime of the network to more than one battery cycle if possible, without compromising the security of the system. A sleep-wakeup schedule will need to be established in order to extend this lifetime. However, there are issues with this method of extending the lifetime, namely the Barrier Coverage Problem, and the Barrier Breach Problem, which I will cover next.

The Barrier Coverage Problem tries to answer a unique question in WSNs; How many sensors are necessary to cover an area, and where should they be placed to be the most efficient? The Barrier Coverage Problem is unique in that it doesn't require the entire area to be covered by wireless sensors. Instead, we form a "barrier" of sensors (Kumar et. al., 2007a). It is only enough if penetrating objects are detected at some point (Mostafaei et. al., 2017). We classify barrier coverage into two distinct categories: weak barrier coverage and strong barrier coverage. Weak barrier coverage requires detecting intruders moving along congruent crossing paths. Strong barrier coverage requires detecting intruders moving along arbitrary paths. This means that, for weak barrier coverage, an intruder could be detected on a straight line congruent crossing path. For strong barrier coverage, these intruders could be detected on any path through the entirety of the system, even if that path changes directions. For example, a three-barrier covered area would have strong barrier coverage, because at some point the intruder would be detected by at least three sensor nodes, due to these barriers consisting of sensor nodes, whose ranges form a barrier across the entire width of the region.

What does it mean for barriers to be k-barrier covered? The following is a definition by Kumar et. al.: “A given belt region is said to be k-barrier covered with a sensor network if all crossing paths through the region are k-covered, where a crossing path is any path that crosses the width of the region completely (2007a).” This is in contrast to an area that has full coverage, where all of the area is covered. In other words, if an area was three-barrier covered, then it would consist of three lines of sensors, each forming a “barrier” across the entire width of the region. The intruder would have to be detected at some point in their trajectory across the area. This is why it is not a requirement to cover the entire area in sensors; however, this is also where vulnerabilities come in, such as the inability to detect an intruder during the phase of switching an active barrier in a sleep-wakeup schedule, which is where the Barrier Breach Problem lies.

The Barrier Breach Problem exists because of the sleep-wakeup schedule. After all, we are trying to extend the lifetime of the network. However, there are major benefits of using a sleep-wakeup schedule, including maximizing the lifetime of the network. For example, if each sensor node had a battery lifetime of one unit of time, and the area was three-barrier covered, then a sleep wakeup schedule could be established so that the network would last up to three units of time, not taking into consideration of any potential breaches that could occur. Unfortunately, breaches can occur in this situation, depending on the position of each barrier of sensors and each sensors battery lifetime.

Specifically, one problem is barriers that cross each other. Although some of these crossing barriers are not penatrible, others are, especially if there is a gap between the barriers that are crossing. This gap would entail that the intruder would wait in that gap until the active barrier switches, breaching the set of barriers. The order in which the barriers are active is also important, because if the intruder can get to the gap in between the two barriers without being detected, then the system is breached (Kim et. al., 2012). Fortunately, there are several algorithms that have attempted to solve this problem.

Stint, Pharari, Max-Flow Edge Eraser, Ordered Cielings, and the dispatching of mobile units are among some of the algorithms that were used to attempt to solve this problem. While all of these are effective for preventing breaches in some applications, they still fail in many others. The Stint and Pharari algorithms proposed by Kumar et. al. are used to develop the map of sensor nodes in a given area, and assign a barrier to them. Then, only one barrier is active at a time. The Stint algorithm is designed for systems whose sensor nodes begin with the same battery life; on the contrary, the Pharari algorithm is designed for those systems with different battery levels (Kumar et. al., 2007b). The Stint algorithm is used as a base for all the other algorithms mentioned above.

The Max-Flow Edge Eraser algorithm proposed by Kim uses Stint to generate the nodes with edges, and removes any edges along the path that cross each other, eliminating the possibility of a breach in the system. However, removing edges may compromise the efficiency of the system, and could result in a lifetime of the network of less than what could have been achieved otherwise, due to a lesser amount of barriers (Kim et. al., 2016). Alternatively, the Ordered Ceilings algorithm proposed by Cobb uses stint to generate the sensor nodes, but then uses its own algorithm to form its own barriers, by obtaining ceilings in order from top to bottom throughout the traversal of the network. As one ceiling is removed, a new ceiling is formed, which yields a new barrier, and the cycle continues until there are no more barriers that can be formed. Excluding Stint, because it was used as an upper bound, Ordered Ceilings proved to be more efficient than Max-Flow Edge Eraser at finding the maximum number of barriers. In fact, Ordered Ceilings can find more barriers than Max-Flow Edge Eraser, while doing it at a significantly lower complexity (Cobb, 2015).

All of the algorithms above dealt with stationery sensor nodes. However, a method proposed by Cheng & Weng dispatches mobile sensor nodes to those areas where a breach could occur in the system (in other words, the gap that is vulnerable to a breach between crossing barriers). There is an algorithm on which mobile elements get dispatched; it depends on their battery life, as well as their location. These mobile sensors move through the system as the active barrier is changing, ensuring that a breach is impossible (2017).

## Method

Our method will consist of using the Max Flow Problem to determine the maximum lifetime of a wireless sensor network. Here, we will use Stint to generate the sensor nodes in a given area, and we will use Stint to form the barriers needed. These barriers will be connected by edges, which will have a cost of 1, or a one unit of time battery life. The Max Flow problem will determine the maximum amount of flow we can send through the system, which will be equivalent to our maximum lifetime of the network.

The purpose of the Max Flow Problem is to maximise the flow of a material through a transportation network. The Max Flow problem consists of a source node, and a sink node. A source node is the only vertex on the graph with no entering edges. A sink node is the only vertex in the graph with no leaving edges. For our application, we will consider the source and sink to be outside the area of sensor nodes. The objective of this algorithm is to find all possible paths from the source to the sink and connect them with edges, which will form our barriers (Levitin, 2012). From there, we will use the algorithm to find the maximum flow, or the maximum time the area can be protected for. Each of the edges will have a capacity, which is the max amount of material that can be sent from node to node through a link in the network represented by an edge.

The Ford-Fulkerson method is used to incorporate a flow augmenting path, which is a path that can add additional flow to the system. The algorithm uses a search algorithm like Breadth-First Search to find these augmenting paths. If an augmenting path is found, the max flow is updated. Otherwise, the flow is deemed optimal. It is important to note that not only are there forward edges in the system, backward edges can be used to create a flow augmenting path (Levitin, 2012). In other words, think of a undirected graph. Any path through that system starting from the source and ending at the sink can be used. The direction of these arrows indicate what to do. If there is a forward edge, add to that edges capacity; however if that edge is a backward edge, take away from that edges capacity. A forward edge is where the tail is listed before the head along a flow- augmenting path ( 1 →…i → j… →n), while a backward edge is where the head is listed before the tail along a flow- augmenting path ( 1 →…i ← j… →n) (Levitin, 2012).

The Max Flow Problem has some efficiency issues that need to be addressed, in which a cycle of paths with the same edge being used with only 1 unit of capacity is acting as both the forward and backward edge, depending on the augmenting path found. Due to the capacity of this edge being 1, we can only add 1 and subtract 1 at each step, creating an inefficient pattern. We know that the result of the graph will have a maximum flow of U units, but the algorithm has to perform these calculations that can prove to be costly (Levitin, 2012). This term is named Efficiency Degradtion, and it is important to note here in case we find paths like this in the experiment.

## Findings

For our experiment, we used the Java programming language to construct the Max Flow algorithm. It is a very complex program that uses a graph generated by the user as the input, and then calculates the maximum flow of that map using the methods mentioned above. This program works beautifully, and accurately calculates the maximum flow of the map given by the user. However, it will not display the actual graph; this is still in the works for later research. We have the goal of using this program to aid us in providing a potential barrier construction algorithm that does not contain a breach in the system, and maximises the lifetime of the network.

## Conclusion

In this work, we researched how we can construct barrier coverage without barrier breach problem and implemented a Java program to simulate a barrier coverage construction algorithm.

For future work, we will implement a Java program to simulate the Stint algorithm, and make it compatible with our Max Flow algorithm, so that we can produce a new system that will maximize the lifetime of the network without compromising the security of it. Also, we will need to modify the Max Flow algorithm so that it takes an input directly from the stint algorithm that will be implemented. The Max Flow algorithm we have implemented in Java has substantial potential to provide us with the best possible outcome for the security and lifetime of the sensor network.

## Bibliography

Cheng, C. & Wang, C. (2017). The barrier-breach problem of barrier coverage in wireless sensor networks. *IEEE Communication Letters. 21*(10), 2262-2265. https://doi.org/10.1109/LCOMM.2017.2694432

Cobb, J. A. (2015). Improving the lifetime of non-penetrable barrier coverage in sensor networks. *2015 IEEE 35th International Conference on Distributed Computing Systems Workshops.* 1-10. https://doi.org/10.1109/ICDCSW.2015.13

Fidge, C. (1988) Timestamps in message-passing systems that preserve the partial. ordering. *Proceedings of the 11th Australian Computer Science Conference, 10*, 56–66.

Kim, D., Kim, H., Li, D., Kwon, S., Tokuta, A. O. & Cobb, J. A. (2016). Maximum lifetime dependable barrier-coverage in wireless sensor networks. *Ad Hoc Networks, 36*(1), 296-307. https://doi.org/10.1016/j.adhoc.2015.08.004

Kim, D., Kim, J., Li, D., Kwon, S., & Tokuta, A. O. (2012). On sleep-wakeup scheduling of non-penetrable barrier-coverage of wireless sensors. *2012 IEEE Global Communications Conference (GLOBECOM)* 321-327. http://dx.doi.org/10.1109/GLOCOM.2012.6503133

Kumar, S., Lai, T.H. & Arora, A. (2007a). Barrier coverage with wireless sensors. *Wireless Networks, 13*, 817–834. https://doi.org/10.1007/s11276-006-9856-0

Kumar, S., Lai, T. H., Posner, M. E., & Sinha, P. (2007b). Optimal sleep-wakeup algorithms for barriers of wireless sensors. *2007 Fourth International*

*Conference on Broadband Communications, Networks and Systems.* 327-336. https://doi.org/10.1109/BROADNETS.2007.4550452

Levitin, A. (2012). Introduction to the design & analysis of algorithms (3rd ed.). Pearson.

Mostafaei, H., Shojafar, M., Zaher, B., & Singhal, M. (2017). Barrier coverage of WSNs with the imperialist competitive algorithm. *Journal of Supercomputing*, *73*(11), 4957–4980. https://doi-org.proxy181.nclive.org/10.1007/s11227-017-2067-x

Patil, H. K. & Chen, T. M. (2017). Chapter 18 – Wireless sensor network security: the internet of things. *Computer and Information Security Handbook, 3*, 317-337. https://doi.org/10.1016/B978-0-12-803843-7.00018-1

Saipulla, A., Westphal, C., Liu, B., Wang, J. (2013) Barrier coverage with line-based. deployed mobile sensors. Ad Hoc Networks, 11(4),1381-1391.