UNIVERSITY OF SHARJAH

Distributed Fault Tolerant Target Tracking in Face-based Wireless Sensor Networks

by

Ammara Razzaq

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE

IN

Computer Science

AT

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF SHARJAH

SUPERVISOR
Dr. Ahmed M. Khedr
CO-SUPERVISOR
Prof. Zaher Al Aghbari

SHARJAH, UNITED ARAB EMIRATES

December, 2018

UNIVERSITY OF SHARJAH

DEPARTMENT OF COMPUTER SCIENCE

Author: **Ammara Razzaq**

Thesis Title: **Distributed Fault Tolerant Target Tracking in Face-based Wireless Sensor Networks**

Department: **Computer Science**

Degree: **MSc.**

Defense Date: 20 December 2018

_____

Signature of Author

UNIVERSITY OF SHARJAH

DEPARTMENT OF COMPUTER SCIENCE

The undersigned hereby certify that they have read and recommend to the graduate studies Council for acceptance of a thesis entitled "Distributed Fault Tolerant Target Tracking in Face-based Wireless Sensor Networks" by Ammara Razzaq in partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Ahmed Khedr

Supervisor: Dr. Ahmed M. Khedr

Co-Supervisor: Prof. Zaher Al Aghbari

Internal Examiner: Dr. Sohail Abbas

External Examiner: Dr. Ouns Bouachir

December 2018

Acknowledgements

First of all, I would like to acknowledge the continuous support of my thesis advisors, Dr. Zaher Al Aghbari and Dr Ahmed M. Khedr, who assisted me throughout my research in preparing this thesis.

I would like to thank my parents for always being there for me and believing in my strengths.

I would like to thank my friends, who motivated me throughout the hardest times of my research. ...

ABSTRACT

Target tracking in face-based wireless sensor networks (WSNs) is an important area of research. WSNs have limited batteries, which are hard to replace due to their deployment in inaccessible areas. Hence the goal of any WSN protocol is to design energy efficient mechanisms that prolong the lifetime of WSNs. Existing work in face-based WSNs utilizes face structure built using Gabriel Graph (GG) or Relative Neighborhood Graph (RNG). However, the existing face structures can lead to high energy consumption as all the nodes in network become part of topology. Due to high energy consumption, the nodes will die quickly causing the partitioning in the network. The objective of this study is to build a new distributed face topology by putting certain nodes to sleep to reduce the energy consumption and prolong the lifetime of network. A distributed target tracking algorithm is designed to run on the proposed face structure which tracks the target accurately while reducing the energy consumption of network. To further prolong the lifetime of network, a distributed fault tolerance protocol is proposed by using the sleeping nodes in the network as backup nodes. When a node's battery reaches a critical level it selects a nearby sleeping node to replace itself to minimize the interruption in tracking of mobile target. The comparison of proposed target tracking protocol on new face structure with existing target tracking protocols shows that, our design achieves better energy efficiency and improves the lifetime of network while maintaining the performance of tracking. At high speeds, our work achieves significant improvement in accuracy of tracking. By using the sleeping nodes as replacement nodes the lifetime of network is further enhanced.

Key Words: Wireless Sensor Networks, Distributed Target Tracking, Face Topology, Fault Tolerance

الملخص

يعد التتبع المستهدف في شبكات استشعار الوجه اللاسلكية (WSN) أحد المجالات المهمة للبحث. تحتوي WSNs على بطاريات محدودة ، والتي يصعب استبدالها بسبب نشرها في مناطق يتعذر الوصول إليها. وبالتالي فإن الهدف من أي بروتوكول WSN هو تصميم آليات كفاءة الطاقة التي تطيل عمر شبكات WSN. يستخدم العمل الحالي في WSNs التي تعتمد على الوجوه بنية الوجه التي تم إنشاؤها باستخدام Gabriel Graph (GG) أو Relative-Neighborhood-Graph (RNG). ومع ذلك ، يمكن أن تؤدي هياكل الوجوه الموجودة إلى استهلاك عالي للطاقة حيث تصبح جميع العقد في الشبكة جزءًا من الهيكل. بسبب استهلاك الطاقة المرتفع ، سوف تموت العقدة بسرعة مما يتسبب في التقسيم في الشبكة. الهدف من هذه الدراسة هو بناء طبولوجيا وجه موزعة جديدة عن طريق وضع عقدة معينة للنوم لتقليل استهلاك الطاقة وإطالة عمر الشبكة. تم تصميم خوارزمية تتبع الهدف الموزعة للتشغيل على هيكل الوجه المقترح الذي يتتبع الهدف بدقة مع تقليل استهلاك الطاقة في الشبكة. لمزيد من إطالة عمر الشبكة ، يقترح بروتوكول الاستعادة الموزع لإزالة الأخطاء باستخدام العقد النائمة في الشبكة كعقدة احتياطية. عندما تصل بطارية العقدة إلى مستوى حرج ، فإنها تحدد عقدة نوم قريبة لاستبدال نفسها لتقليل الانقطاع في تتبع هدف الجوال. توضح مقارنة بروتوكول التتبع المستهدف المقترح مع بروتوكولات التتبع المستهدفة الموجودة أن تصميمنا يحقق كفاءة أفضل للطاقة ويحسن عمر الشبكة مع الحفاظ على أداء تتبع موضوع. بسرعات عالية ، يحقق عملنا تحسيناً كبيراً في دقة التتبع. من خلال استخدام عقدة النوم كعقد بديلة ، يتم تحسين عمر الشبكة.

الكلمات المفتاحية: شبكات الاستشعار اللاسلكية ، تتبع الأهداف الموزعة ، طوبولوجيا الوجه ، التسامح مع الخطأ

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Study Background

Wireless sensor networks (WSNs) consist of hundreds and thousands of low power and low cost sensor nodes which are deployed in a random or deterministic manner to monitor different environmental attributes [1]. Sensor nodes are equipped with computation and communication capabilities. Sensor nodes sense the environment, process the sensed information and communicate with each other to forward the data to sink node. Sink node is responsible for gathering the data from all sensor nodes. A sink node can be static or mobile. In case of static sink node, sensor nodes will forward the data to sink using a multihop routing protocol. A mobile sink moves in the network to gather the data from representative/ beacon nodes or cluster heads in cluster based WSNs. Sensor nodes in WSN are usually densely deployed i.e. neighbor nodes may be very close to each other [2].

WSNs have many constraints that need to be dealt with while developing an application. Energy is the biggest resource constraint in WSNs. Hence, the need is to develop

energy efficient algorithms that consume minimum energy and perform accurately. Minimum energy consumption will lead to prolonged network lifetime. Energy is consumed during computation at sensor nodes and communication between sensor nodes. Computation and communication costs are the main reason for energy depletion in sensor nodes. The goal is to reduce these costs and increase the lifetime of sensor network [3]. Sensor nodes are prone to failure due to harsh environmental conditions, malicious attacks on the network, hardware failure such as battery depletion or software failure such as wireless channel fluctuations [4]. A WSN should be able to recover from failures in a timely manner to keep performing continuously and accurately.

A topology in wireless sensor networks is the grouping of nodes to collect data cooperatively and in an energy conserving manner. Mainly three types of network topologies exist in WSNs, *tree* [5], *cluster* [6] and *face* [7] topology. Many distributed tracking protocols have been designed for different network topologies such as, cluster based [8], tree based [9] and face based [10] target tracking protocols. The goal of these target tracking applications is to gather data about one or multiple targets and estimate their positions in the network [11].

In face topology, network is divided into a set of polygons called faces. A face consists of minimum three nodes. A face structure is a planar graph where nodes act as vertices of the graph and edges between nodes represent the communication links. A planar graph is a graph which has no crossing edges [12]. Gabriel Graph (GG) and Relative Neighborhood Graph (RNG) are two well-known planar graphs which are used to build face structure.

WSNs are usually densely deployed, which means there are many nodes in each area of the network [13]. In dense WSNs the number of sensors per unit area is large and it can go as high as 20 nodes/m^3 which can cause many issues related to redundancy, radio contention and scalability [14] . In dense networks, it is not always necessary to keep all

the node active. Some nodes can be put to sleep because they are contributing almost the same as the other nodes in the network. In the existing face topology, all the nodes in the network become part of topology which may result in very small faces in case of dense networks. A too small face size can cause irregular signal patterns affecting the accuracy of application running on the network [15]. The overall energy consumption of the network will also be high as all the nodes will be part of topology.

In general, in target tracking applications, a node detects the presence of target by sampling the sensed signals (e.g., light, sound, image or video), then these readings are used to estimate the trajectories of one or more targets and notify the sink node [16]. The main goal of a target tracking protocol is to design an energy efficient mechanism that tracks the target accurately and prolongs the lifetime of WSN [17]. In face-based target tracking protocols, once the target is detected, two or more number of node collaborate with each other to track the target. Once the target moves away from these node, new set of nodes are selected to track the target. One of the nodes from the set will be beacon node that will be responsible for communicating with the tracker. The beacon nodes keep track of the last known target location or next beacon node in case the target has moved away. A mobile tracker moves in the network, communicates with the beacon nodes in order to capture the target in the network.

In [10-18], object tracking protocols have been developed in face-based WSNs. However, none of these works have tried to modify the topology of the network to adapt to new situations such as, identification of certain nodes to put to sleep to minimize energy consumption. In [13], a slight change in topology is made but it is very similar to the original face topology. In these approaches, all the nodes in network take part in the topology formation, which is energy consuming specially in dense networks. Due to the small faces, objects moving at high speed will leave the face quickly increasing the chances of target loss. The main focus of these protocols have been on designing the prediction protocols to predict the future location of target and only awaking the nodes in predicted

region to save the energy. However, in prediction based protocols, a target can be easily lost if it does not move towards the predicted region. In this case, a target recovery mechanism will be required to recover the target by activating as many number of nodes as required. This can lead to excessive energy consumption.

In this study, a new face structure is proposed in which extra nodes are put to sleep before building the face topology. The resulting network will have generally bigger faces which will solve the problem of signal irregularities. Due to sleep nodes in the network the overall energy consumption of face structure will be less and the life time of network will be improved.

To test the performance of the proposed face structure a new target tracking protocol is designed to compare with existing face based tracking protocols. This will improve the energy efficiency and lifetime of the network while maintaining the performance of tracking protocol. The nodes in the resulting topology will be fully connected means there will always be a single/ multi-hop path available from one node to another node in the network.

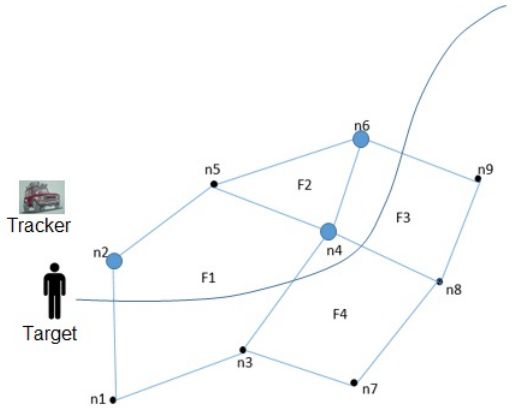


FIGURE 1.1: Target tracking in face topology

An example of face-based target tracking is shown in Figure. 1.1. Target moves in a random path inside network. As the target enters the network, it is detected by node $n2$, which is the nearest node to the target. Node $n2$ becomes the representative node

and starts tracking the target. As the target moves away from node $n2$, another node that is now nearest to the target becomes representative node. The target is continuously tracked by representative nodes until it leaves the network. A representative node is responsible for communicating with tracker/sink node to give the information of target.

Another important factor in the performance of WSN protocols is their ability to deal with faults in the network. Any fault that prevents a node from sensing or transmitting is considered a failure. Faults in sensor nodes can cause error in sensing, processing and communicating data [18]. Distributed protocols requires self healing to deliver a desired level of functionality in the presence of faults [19]. Faults in the network mainly occur due to node, network or sink problems [20]. Faults may occur in the network due to several reasons such as misplacement of nodes, battery depletion, breakage of links due to wireless channel fluctuations. Harsh environmental conditions affect the radio of sensor nodes. Since the battery of sensor nodes is limited, it will deplete after a certain amount of time causing the node to fail. Sensor readings may become incorrect when the battery of sensor node reaches a certain level [21].

Faulty nodes cause the WSN to be partitioned, causing holes in the network, and disrupting the performance of the network as a consequence. In target tracking applications, partitioning of the network will result in loss of tracking and the accuracy of the tracking protocol will be reduced.

Although fault tolerance has been studied in detail in general, it is not extensively studied in face based WSNs. In face based WSNs, mainly target recovery protocols have been designed to recover the target in case of target loss. Fault tolerance is a significant requirement of WSNs. The goal of a fault tolerance protocol is to keep the network performance high for as long as possible by fixing the faults in the network in a timely manner [22].

1.2 Statement and purpose of the problem

The current face structures use all the nodes in network as part of topology which is not an energy efficient approach, especially in dense WSNs, where smaller faces lead to reduced accuracy of target tracking protocols. Generally, current face structures don't have any fault tolerance support and node failures lead to network partitioning further reducing the accuracy of tracking protocols.

The main goal of this work is to design and implement an energy efficient, life prolonging mobile target tracking system by designing a new face structure. Another goal of this study is to design a distributed fault tolerant protocol to recover from fault in the network due to battery depletion. The purpose of this study is to design the distributed fault tolerant target tracking system in following steps:

1. Designing a new distributed energy efficient face structure for WSNs which has the following characteristics.
   (a) The face structure is built using certain nodes in the network and other nodes are put to sleep.
   (b) The coverage of edges in the network is ensured, which is an important parameter in target tracking protocol.
   (c) The resulting network is a planar graph i.e. no two edges cross each other.
2. Designing a distributed target tracking system for WSN which makes use of the property of the face structure that the edges are covered.
3. Designing a distributed fault tolerant target tracking protocol using the sleep nodes in the network to prolong the lifetime of WSN.

1.3 Study questions

1. How to design a new distributed face structure by putting certain nodes to sleep while ensuring the coverage of edges in the network?

2. How to design a new face structure that is better than the existing face structure in terms of energy consumption and lifetime?

3. How to design an accurate and energy efficient distributed target tracking protocol be designed for the new face structure?

4. How to design a distributed fault tolerant protocol be designed by using the sleeping nodes in network to wake-up later to replace a dying node to improve the lifetime and maintain the performance of the network?

1.4 Significance of the study

This study exploits the possibility of building a new face structure in order to save the energy and prolong the lifetime of network. For now, the face structure that is built using Gabriel Graph or Relative Neighborhood Graph uses all the nodes in the network. The significance of this study lies in using the less number of nodes is building the face structure and putting the remaining nodes to sleep while maintaining the coverage of edges in the network. A new target tracking protocol is designed to track the target accurately while conserving the energy of the network. The sleep nodes are used later in designing a fault tolerance protocol to further improve the lifetime of the network.

1.5 Definition of terms

- **Neighbor Node:** A node within the communication range of node.
- **Sensing Neighbor:** A node within the sensing range of node.
- **Edge Node:** A node with which an edge is made. The edge represents the communication link between nodes.
- **Sleep Node:** The node whose radio is turned off. It is neither transmitting, nor listening.
- **Active Node:** A node whose radio is turned on. It is actively listening to the incoming messages.
- **Boundary Node:** A node at the boundary of the network.
- **Fault:** Weakness of the system that causes an error in network.
- **Fault Tolerance:** Continuously maintaining WSN optimal performance despite having faulty nodes. [23]

- **Fault Detection:** Detecting faulty functionality in a system by self diagnosis or cooperative diagnosis.
- **Fault Recovery:** Restoring the correct functionality after the fault detection by repairing or replacing the failed component.
- **Topology:** The logical or physical organization of the nodes in the network.
- **Duty Cycle:** The time period where a node alternates between being awake and being sleep.
- **Network Lifetime:** The amount of time where a network will remain active, either containing at least one node or containing the minimum number of nodes.
- **Neighbor List:** The nodes in the network that can be reached from a node at one hop distance.
- **Base Station:** The sink node that collects network management data and application data.
- **Planar Graph:** A graph in which no two edges cross each other.
- **Target:** An unauthorized entity moving in the network.
- **Tracker:** A mobile sink moving in the network trying to capture the target.
- **Beacon node:** Sensor nodes in the network responsible for communicating with tracker.
- **Coop node:** Sensor nodes tracking the target upon request of beacon node.

1.6 Limitations of study

- All sensor nodes in network are homogeneous.
- Each sensor can compute its location using a GPS or non-GPS technique. [24–26]
- There is no fault in processing and transmitting/receiving neighboring measurements as well as the proper execution of algorithms.
- Sensor nodes are randomly deployed in the network.
- A sensor can detect the location of target using a target localization algorithm.
- The designed object tracking protocol is for single target tracking.

1.7 Thesis Organization

In the first chapter study background of the problem that is investigated in this thesis, motivation for this research and problem statement are presented. In Chapter. 2, related work regarding face structure, target tracking and fault tolerance in face-based WSNs is reviewed. Proposed work is presented in Chapter. 3. In Chapter. 4, the performance of proposed protocol is evaluated through extensive experiments. Finally, conclusion and future work is given in Chapter. 5.

Chapter 2

Literature Review

2.1 Face Architecture

In Gabriel Graph (GG) based face structure [12], an edge is valid between two nodes if there is no node inside the circle formed by the edge as a diameter, otherwise the edge is invalid.

In Relative Neighborhood Graph (RNG) based face structure [27], an edge exists between two nodes if there is no node inside the intersection area of their communication circles.

For a detailed description of how a face structure is built, refer to Appendix A.

In both these structures, all the nodes in network become part of topology. In dense WSNs, where nodes lie very close to each other, the resulting topology will consist of large number of faces which will be very energy consuming.

2.2 Target Tracking

In face based tracking, object is tracked through faces in the network. One or more nodes in every face will be responsible for tracking the target. A mobile sink/tracker may be chasing the target in the network by getting information through representative nodes in each face.

Many object tracking protocols have been designed using face structure in WSNs. These protocols can be divided into prediction based and non-prediction based tracking protocols. The prediction based protocols [10, 28–34] aim to track the target using as less number of nodes as possible and focus on developing a prediction mechanism to predict the future location of the target. The nodes in the predicted region are awaken to continuously track the target hence they are more energy efficient, but they can lose the target very easily if it doesn't move to the predicted area and a target recovery mechanism will need to be invoked to rediscover the target.

In non-prediction based protocols [35, 36], a target is usually trapped by one or more faces, to track the target. The target will be tracked in whichever direction it moves, because it is covered from all sides. Hence no prediction is required to predict the future position of target. Non-prediction mechanisms are more efficient than prediction based mechanisms as they don't lose the target easily. However, they can be more energy consuming depending on how many nodes are tracking the target at a time.

There is a trade-off between accuracy and energy consumption of a target tracking protocol, hence it is crucial to find an optimal solution.

All the face based tracking protocols mentioned in this literature review are distributed in nature. From building face topology to running target tracking protocol, every thing is done in a distributed manner. Most of these works use mobile sink/tracker to track the target. Tracker moves in the network and communicates with beacon node.

Mobile sink sends a flooding request in the network to detect the target. Initial beacon node that detects the target and is the nearest node to that target responds to tracker. All beacon nodes keep information of its next beacon node. When the tracker reaches the initial beacon node it gets information about next beacon node and moves towards the beacon node. The tracker keeps on following the beacon nodes until it captures the target in the network.

In [35], a dynamic object tracking (DOT) solution is provided using face architecture. This paper uses the spatial neighborhood discovery algorithm from [37] to track the target. Sensor node that is nearest to the target acts as beacon node and wakes up all its spatial neighbor nodes to cooperatively track the target, in order to avoid losing target tracks. However, waking up all the spatial neighbor nodes consumes a lot of energy. In this work, no prediction mechanism is applied to predict the future position of the target. This was the first paper that used face structure for target tracking. Before this, face structure was mainly used in routing protocols. After this paper, the focus has been on two main things. 1) using prediction to wake up nodes in future location instead of waking up all spatial neighbors and 2) recovery mechanism instead of flooding the whole network.

Another non-prediction based approach called "face track" is presented in [36]. In this approach, only the face in which target is present is used to track the target. The target crossing a face is detected using a brink detection algorithm. Upon brink detection, the forward polygon is informed that the target is approaching and the nodes in forward polygon wake up and wait for the target to enter the face. The brink detection is a two step process, first square brink is detected and the previous polygon is informed, then rectangular brink is detected and forward and previous polygons are informed.

In [31], the focus is on using minimum number of sensor nodes to track the target. The node that is near to the target becomes monitor node and selects a backup node to

cooperatively track the target. The monitor node also selects a future monitor node by using the proposed prediction mechanism.

In [28], only the beacon node and one of its immediate neighbor nodes cooperatively track the target. Beacon node is responsible for communicating with the tracker. A beacon node determines the next beacon node based on a prediction mechanism. If both nodes fail to to detect the target, then it is assumed that the target may have changed its direction, so all the face neighbor nodes are requested to track the target.

In Face-based Object Tracking Protocol (FOTP) [10], a hexagon based prediction mechanism is designed to reduce the number of awaking nodes. Instead of waking up all the spatial neighbors, this algorithm proposes a hexagon based prediction mechanism to predict the future location of target. The prediction region is limited to 1/6 of hexagon to awake few nodes and save the energy. The spatial nodes are activated only to recover the lost target. One drawback of this approach is that, target can easily be lost since prediction generates future location based on only the current and prediction location. If the target is moving fast, it will be lost by the time spatial neighbors are activated, and then all the sensor nodes will need to wake up to recover the target. Secondly, random soldier nodes are used to detect the entrance of target. The target may not be detected at all until it comes in the vicinity of one of the soldier nodes.

In [30], a Prediction Based Object Tracking approach (POOT) along with two target recovery schemes are presented. To detect the presence of target in network, the source uses the flooding mechanism. The target can be detected by more than one nodes, and the nearest node to the target takes the responsibility of tracking the target. When the target is detected, its future location is predicted using a linear prediction method. Then the nodes in the current face and the future predicted face are activated to track the target. When the target reaches the next predicted face, the nearest node to the target in that face takes responsibility of tracking the target. If the target does not enter the predicted

face, it is lost. To recover the lost target, two target recovery schemes, Time-efficient Object Recovery Scheme ($TORS$) and Communication-efficient Object Recovery Scheme ($CORS$) are proposed. These schemes consume less energy as compared to flooding the whole network to recover the target.

In [29], an object tracking approach based on prediction of target trajectory is proposed. Least square method (LSM) is used to predict the future location of target. The prediction scheme is used when the object reaches close to the border of the residing face. The nodes in the future face are awaken and the nodes in previous face go to sleep to save energy. To recover the lost target, time-efficient object recovery scheme ($TORS$) [30] is used to search the area where object was lost.

Uptil now, all the face based tracking protocols have not focused on building a new planar topology for better energy efficiency and prolonging the lifetime of WSNs.

2.3 Fault Tolerance

In the literature, fault tolerance has been studied in various aspects. In [22] fault tolerance techniques are divided into curative and preventive techniques. A curative technique is the one in which the fault tolerance is applied after the fault has occurred and detected in the network. The purpose of a curative technique is to restore the connectivity or coverage of network topology [38, 39]. A preventive technique is applied before the occurrence of fault to maintain the topology of network by predicting that a fault is going to happen a priori. The failed node is replaced with a backup node by waking up the backup node when a fault is predicted or by moving some nodes to the location of faulty node.[40, 41].

Fault tolerance techniques can also be categorized based on how many nodes are required to detect a fault. If a node can detect a fault in itself, it is called self diagnosis technique. If more than one nodes need to cooperate with each other, its called a cooperative technique [42]. Faults such as depletion of battery and broken links can be detected by a node itself. The depletion of battery can be determined by monitoring current battery voltage. The broken links can be detected when a node doesn't receive any messages from a neighbor in a certain amount of time.

Another categorization of fault tolerance techniques is centralized vs. distributed [43]. In a centralized fault tolerance approach, the nodes will have to send their data to base station and the base station/sink node will implement the fault tolerance mechanism. In a distributed fault tolerant protocol, nodes can detect faults and recover from faults locally without any communication with a central station. A distributed cluster based self healing fault tolerance protocol is presented in [43]. In case of battery level reaching below a threshold value, sensor node sends a notification message to cluster head. the cluster head declares this node as sleeping node, removes it from other lists and update the topology.

Although fault tolerance has been extensively studied in WSNs, not much work has been done in face based WSNs.

In [36], a fault tolerance protocol is presented which suggests merging two or more faces into one single face. Merging of faces is done once the fault is detected in the network. Hence, this approach is curative in nature.

Some other face based target tracking protocols presented target recovery protocols in case of target loss. Target loss can happen as a result of faults in the network like node failures, link failures etc. A target recovery mechanism tries to find the target by gradually increasing the number of active nodes where the target was last found. Most works that work with prediction based target tracking protocols, focus on target recovery mechanism in case of loss of tracking of target. In [31] a target recovery mechanism is given in which the neighbor sensors close to the monitor node try to relocate the target. If the target is not found all the nodes in the face cooperate to find the target. If the target is still not detected, all the neighbor sensors in close vicinity of the face try to recover the target. If all tries fail, the network return to the initial state where every node in the network try to locate the target.

Another target recovery mechanism is proposed in [34] in case of target loss. If the target does not move to the predicted face, the target is assumed to be missing and the target recovery mechanism is triggered. Firstly, the previous face is awaken to check if the target is still there. If the target is not found, the vicinity faces are awaken one by one until the target is found. Lastly, if the target is not in any of the vicinity faces, the entire network will be awaken to find the target.

Chapter 3

Methodology

The proposed face structure is built in a distributed manner using distributed scheduling algorithm. In a distributed slot scheduling algorithm [44], every sensor node computes its own time slot schedule in a distributed manner to avoid collisions. The proposed face structure algorithm requires that each node makes edges at individual time, hence a distributed scheduling algorithm is used for this purpose.

At first, the nodes will gather the information of their communication neighbors using neighborhood discovery protocol. Then each node will make edges, at its assigned time slot. A node will make edges with nodes that are farthest in sensing ranges and put the closer nodes to sleep by running the proposed edge making protocol. Once all the nodes have made edges, the result is the proposed face structure where nodes that made edges have become part of topology and remaining nodes have gone to sleep. Next every node runs a face discovery protocol to discover the spatial neighborhood nodes. Once, all these steps are done, network is ready for target tracking.

The target tracking protocol makes use of the fact that all the edges are covered. Once the target is detected in a face, it cannot escape the face without being detected by

the nodes in a face. The target is tracked by the face in which the target is present and as the target moves in the network, it is tracked by the faces that it crosses.

Since the battery of sensor nodes is limited, once the battery of a node is depleted, it will die and its edges will break causing partition in the network. The proposed fault tolerance protocol makes use of the sleep nodes in the network to replace the dying node to avoid partitioning the network and keep the network alive and performing for longer periods of time.

3.1 Neighborhood Discovery Protocol

In this protocol the information of the neighboring nodes is gathered by sending broadcast packets by each node. A broadcast packet is a packet which is transmitted without any recipient node address and all the nodes in communication range will hear the broadcast [45]. According to the neighborhood discovery protocol , the recipient nodes will save the sender node as a communication neighbor in their memory. A communication neighbor is one-hop neighbor of a node with which a direct communication link exists.

The neighborhood discovery protocol in algorithmic form is given in Algorithm. 1. The update_Neighbor_Table function stores the information of communication neighbors in memory along with its id and location.

Algorithm 1 Neighborhood Discovery Protocol

Input: $\emptyset$
Output: Set of communication neighbors
1: INITIALIZATION: send Broadcast Discovery packets
2: **if** Received Broadcast Discovery Packet **then**
3: update_Neighbor_Table()
4: **end if**

3.2 Boundary Discovery of the Network

Once the neighbor nodes are discovered the next step is to discover the boundary of the network to ensure that boundary nodes are not put to sleep and the boundary of the network remains intact. The boundary of the randomly deployed WSN can be discovered using a distributed boundary discovery protocol [46, 47]. We need to discover the minimum perimeter coverage of the network i.e. the minimum number of nodes that are covering the perimeter of the network will be declared as boundary nodes. For this purpose, minimum perimeter coverage protocol from [48] is used. An example of the boundary in a network is shown in Figure. 3.1. The nodes in grey colour represent the boundary of the network.

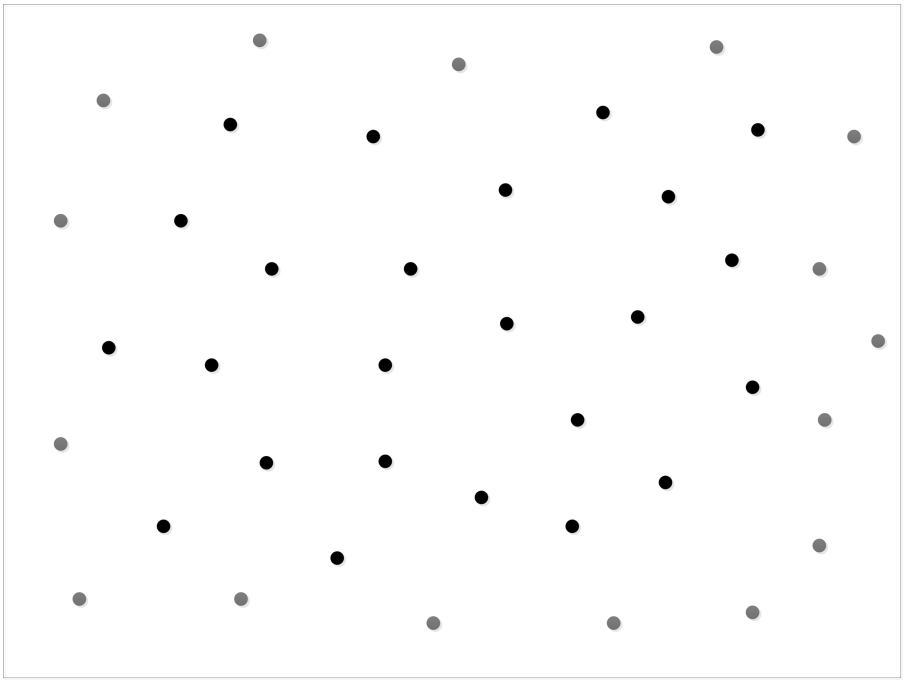


FIGURE 3.1: Boundary of the Network Example

3.3 Distributed Slot Scheduling

Each node runs the edge making algorithm where it makes edges with some nodes in the sensing range and puts some node to sleep. Since some nodes go to sleep, the edge making process of one node will affect the edge making process of other nodes. The nodes that go to sleep should not make edges and the nodes that have been made edges with, will not be put to sleep. A distributed scheduling algorithm [49] is used to schedule the time slots for each node in which it will run the edge making algorithm in a distributed manner as shown in Figure. 3.2.

FIGURE 3.2: Distributed Slot Scheduling

3.4 Proposed Face Structure

In the proposed face structure, each node makes edges according to the proposed edge making criteria. Every node works at its own allotted time assigned using the distributed scheduling algorithm. Once all the nodes have made edges, the resulting network will be a face structure in which some nodes have been put to sleep for conserving energy.

Mathematical notations used in proposed face structure are given in Table. 3.1.

Notation	**Meaning**
A_N	Active Node
S_N	Sleep Node
B_N	Boundary Node
AE_N	Active node that has made edges
IS_N	Invalid Sleep Node

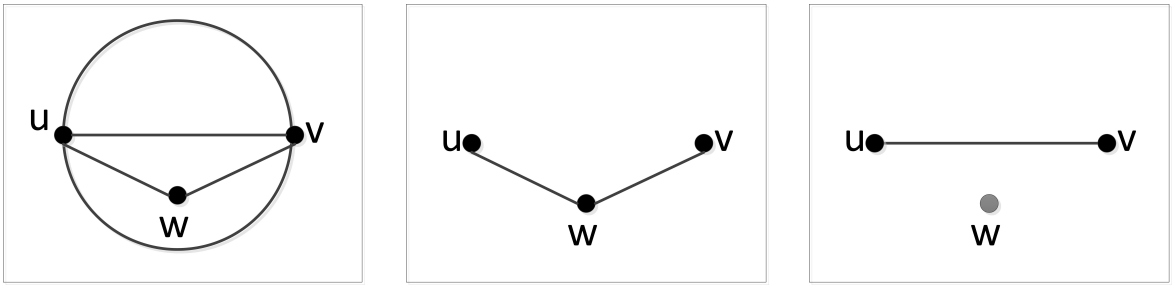
TABLE 3.1: Mathematical Notations for Face Structure

3.4.1 Edge Making Algorithm

Edge making algorithm is used by nodes in the network for building the proposed face structure. A node tries to make edges with the nodes that lie inside its sensing range. Since a node knows the information of the nodes that lie inside the communication range and sensing range is always less than the communication range, it selects the nodes that lie inside the sensing range and sort them based on the distance from farthest to nearest nodes. Then, the node checks the validity of the edges with each node in the sorted order, starting from farthest node. If there is a node inside the circle formed by the edge as diameter, the node will check if it can be put to sleep. If the node is a boundary node B_N, it will not be put to sleep, because making a boundary node sleep will result in reduced boundary of the network.

If the network is represented as a graph $G = (V, E)$, nodes in the network represent the vertices V of the graph and communication links between nodes represent the edges E. Initially, every node can communicate with all of its neighbors hence the network is a fully connected graph as shown in Figure. 3.3a. In Gabriel Graph, edges that don't fulfill the criteria of Gabriel edges are removed. In Figure. 3.3b, edge e_{uv} is removed as node w lies inside the circle formed by the edge as diameter. e_{uw} and e_{vw} are kept as they fulfill the criteria of Gabriel edges. In proposed face structure as shown in Figure. 3.3c, if node u makes edges first, it has two neighbors v and w. It will pick the farthest node first i.e. v and check the validity of edge with this node. The node w comes inside the circle formed by the edge uv as diameter. If node w is not a boundary node, it is put to sleep and edge uv is valid, which is the case in Figure. 3.3c. Otherwise, if node w is a boundary node, edge e_{uv} would be invalid.

An example of a node making edges with all its sorted neighbors according to edge making criteria is shown in Figure. 3.4a. Node $n1$ has five neighbors inside sensing range $[n2, n3, n4, n5, n6]$, given in sorted order as $[n3, n4, n2, n5, n6]$. $n1$ checks the validity

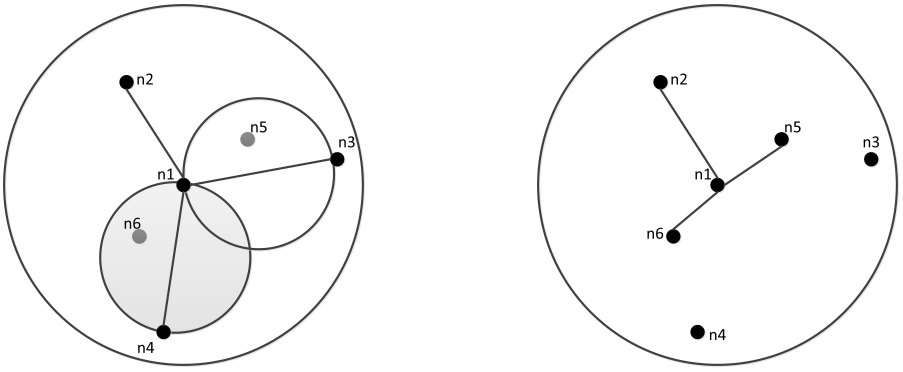


(A) Fully Connected Graph (B) Gabriel Graph (C) Proposed Graph

FIGURE 3.3: Comparison between Gabriel and Proposed Structure

of edge with $n3$, $n5$ is lying inside the circle formed by the edge $e_{n_1n_3}$, and it is not a boundary node, hence it is put to sleep, and an edge is made. Then, $n1$ checks the validity of edge with n_4. n_6 is lying inside the circle formed by edge $e_{n_1n_4}$ as diameter and it is not B_N hence it is put to sleep and edge $e_{n_1n_4}$ is termed valid. Next node in the sorted list is n_2, and there is no node inside the circle formed by edge $e_{n_1n_2}$ as diameter, hence it is also valid. Next in the sorted list are n_5 and n_6, which are both asleep.

In comparison, Gabriel edges are shown in Figure. 3.4b for the same set of nodes. Node n_1 checks the validity of edges with each node one by one. Edge with n_3 is invalid because n_5 comes inside the circle formed by edge $e_{n_1n_3}$ as diameter. Similarly, edge with n_4 is invalid due to node n_6.



(A) Edge Making for Proposed Face Structure (B) Edge Making for Gabriel Face Structure

FIGURE 3.4: Comparison between Gabriel and Proposed Edges

Initially, there were only active nodes (A_N) and boundary nodes (B_N) in the network. Once nodes have started making edges, there will be two more types of nodes in the network, sleep nodes (S_N) and nodes that have made edges (AE_N). These nodes broadcast their information so that their neighbors know know their current status. The sleep nodes will not make edges.

When further nodes make edges, they will check if a node in their sensing range is a sleep node and they will not make edges with sleep nodes. It is also possible that S_N or AE_N lie inside the circle formed by the edges as diameter. In case of S_N, edge will be valid because it is already a sleep node. The valid edge criteria is shown in Figure. 3.5. If S_N or A_N lie inside the circle formed by the edge e between nodes u and v, the edge e is valid. In case of A_N, it will be put to sleep. S_N does not need to be put to sleep because it is already asleep.

In case of AE_N, edge will be termed invalid, because it is a node that has already made edges. If it is put to sleep, its edges will be broken. If its edges had put some other nodes to sleep, they will have to be awoken. It can be called *reverse effect* in which the process that has already been done needs to be undone. In order to avoid *reverse effect* the edge is termed invalid. The invalid edge criteria is shown in Figure. 3.6. Edge between u and v is invalid, because a boundary node B_N or AE_N lie inside the circle formed by the edge. Since these nodes cannot be put to sleep, they are called invalid sleep nodes IS_N.

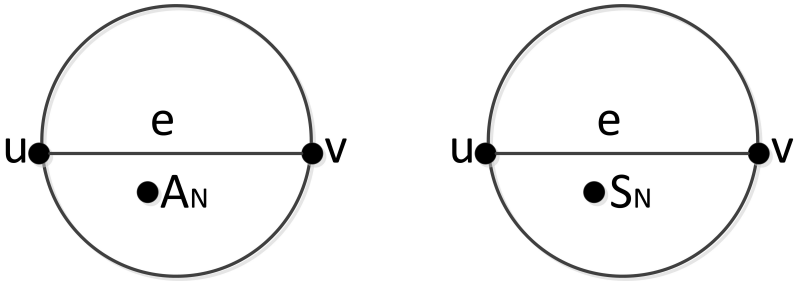


FIGURE 3.5: Valid Edge Criteria

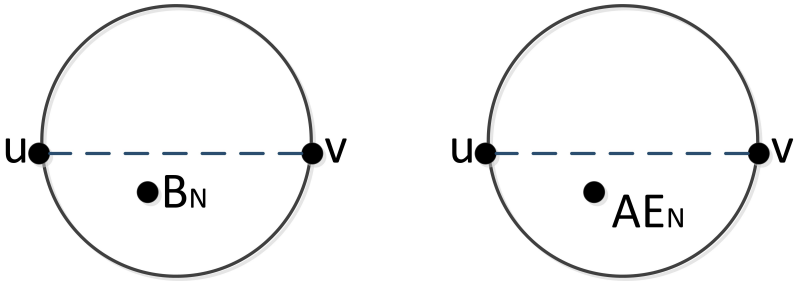


FIGURE 3.6: Invalid Edge Criteria

Algorithm. 2 shows the edge making algorithm.

Algorithm 2 Edge Making Algorithm

Input: Nodes in sensing range
Output: Edge nodes and sleep nodes

```
1: Sort_Sensing_Neighbors()
2: for all sorted nodes do
3:    if active node then
4:       if no I_SN node inside circle formed by edge as diameter then
5:          edge is valid
6:       else
7:          edge is invalid
8:       end if
9:    end if
10: end for
```

3.4.2 Coverage Property of Edges

Here we are considering all the nodes in sensing range instead of communication range, reason will be explained later. And all the neighbors are sorted from farthest to nearest. The node tries to make edges with the nodes from the sorted list, one by one, starting from farthest node. If there is a node inside the circle, formed by the edge as diameter, it will be put to sleep. The node cannot make an edge with sleep node. The node will traverse the sorted list one by one, until it is completely traversed. At the end of this process, the node will have made edges with some nodes and will have put some nodes to sleep.

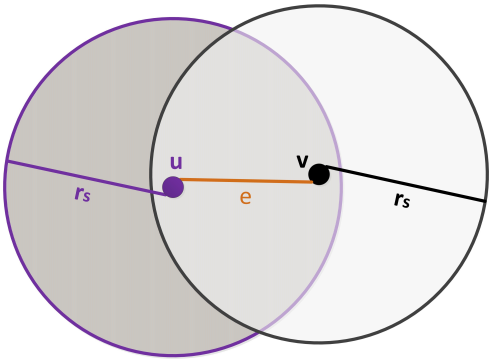
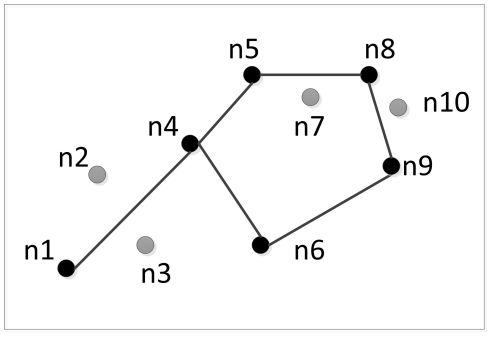


FIGURE 3.7: Coverage of edge in proposed face structure

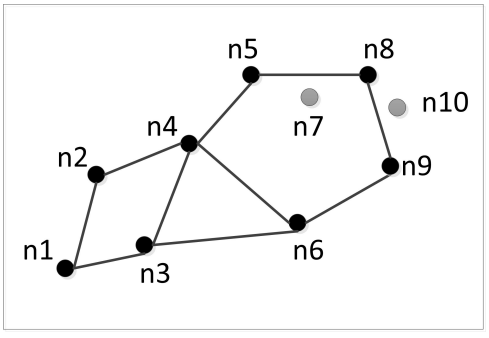
3.4.3 Condition on Minimum Number of Edges

For a node to make a face, it needs to make at least two edges. If a node makes less than two edges it will result in incomplete face structure as shown in Figure. 3.8. In Figure. 3.8a $n1$ has three nodes in its sensing range $n2, n3, \& n4$, it made edge with $n4$ and put $n2$ and $n3$ to sleep. Now there are no other nodes to make edges with, hence $n1$ is only able to make one edge, so it cannot make even one face. In this scenario, proposed face structure is incomplete.

If such a case occurs, in which a node is unable to make at least two edges, it will try to make Gabriel edges. In Figure. 3.8b, node $n1$ made Gabriel edges with nodes $n2$ & $n3$. Hence, it made a complete face.



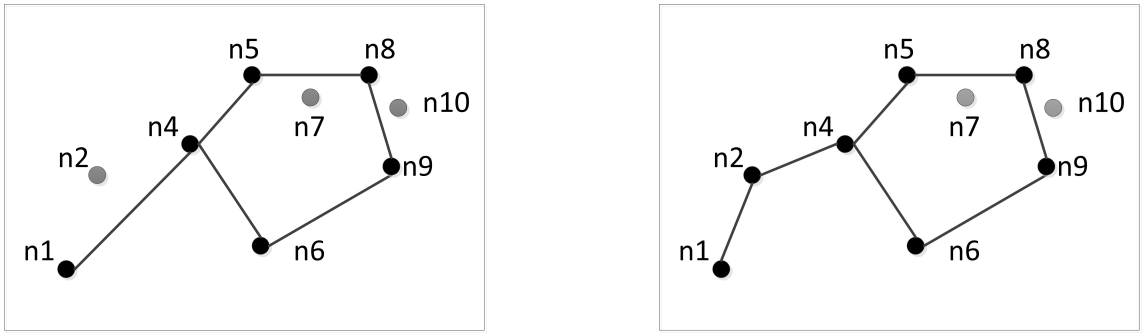
(A) Proposed graph



(B) Gabriel graph

FIGURE 3.8: Condition on minimum number of edges: scenario 1

However, it is possible that even using Gabriel graph approach, a node cannot make at least two edges because it also depends on the distribution of the nodes in the network, which is completely random. This scenario is shown in Figure. 3.9, in which $n1$ has two nodes in its sensing range $n2\&n4$, in proposed graph, it makes an edge with $n4$ and puts $n2$ to sleep. $n1$ can make only one edge using proposed graph approach. It tries to make edges using Gabriel graph approach, and still it cannot make more than one edge. In this scenario, $n1$ is able to make edge with $n2$ only.



(A) Proposed Graph (B) Gabriel Graph

FIGURE 3.9: Condition on minimum number of edges: scenario 2

3.4.4 Proof that Proposed Face Structure is Planar Graph

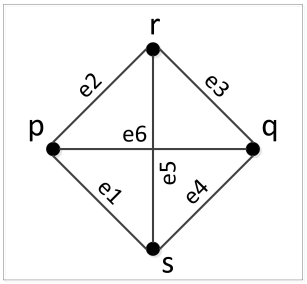
The proposed face structure is a planar graph, means there are no crossing edges in the graph like Gabriel graph. In Gabriel graph, crossing edges in a fully connected graph do not satisfy the criteria of Gabriel edges, hence they are removed. The condition that is used to check the validity of edges is that if there is a node inside the circle formed by the edge as diameter, the edge is invalid, represented in equation form as:

$$\forall r \neq p, q : d^2(p, q) < [d^2(p, r) + d^2(q, r)] \tag{3.1}$$

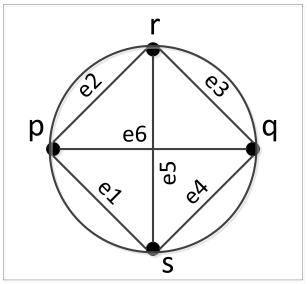
According to equation 3.1 of Gabriel graph, if there is a node r inside or on the circumference of circle formed by edge pq as diameter, the edge pq will be invalid.

In Figure. 3.10a, a fully connected graph between nodes $p, q, r\&s$ is shown. The $e6$ and $e5$ are crossing each other, hence the fully connected graph between these nodes is not a planar graph. If p checks the validity of edge $e6$ with q as shown in Figure. 3.10b, nodes r & s lies on the circle formed by the edge $e6$ as diameter, hence $e6$ is not a valid Gabriel edge. Similarly, edge $e5$ between nodes s and r is invalid because nodes p & q lie on the circle formed by the edge $e5$ as diameter. Gabriel graph after removing crossing edges is shown in Figure. 3.10c.

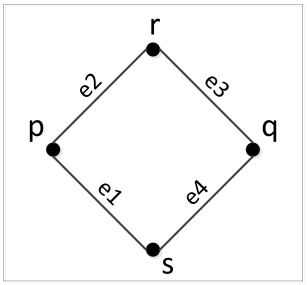
The only difference between criteria of Gabriel edge and proposed edge is that, if there are nodes lying inside the circle, they are put to sleep and an edge is made as opposed to Gabriel graph where the edge will be termed invalid. In the proposed graph, if node p makes edges first, it will pick the farthest node q and check the validity of edge $e6$. Nodes r & s are lying on the circle formed by the edge $e6$ as diameter, so they are put to sleep. Hence, the proposed graph does not have any crossing edges either as shown in Figure. 3.10d.



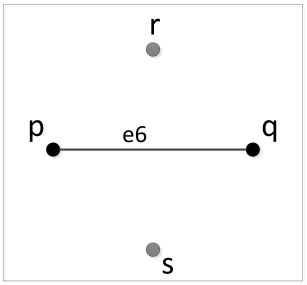
(A) Fully connected graph



(B) circle around e5 & e6 as diameter



(C) Gabriel graph



(D) Proposed graph

FIGURE 3.10: Proof that proposed face structure is planar graph

An example of proposed face based WSN is shown in . 3.11. n1, n4, n10, n11, n20, n27, n35, n37, n38, n39, n40, n31, n26, n18, n5, & n3 are boundary bodes of the network. Some nodes have been put to sleep and did not make any edges. Even though, coverage of the network is reduced due to putting nodes to sleep, but the edges between the nodes are still covered. This property will be crucial in designing the target tracking protocol.

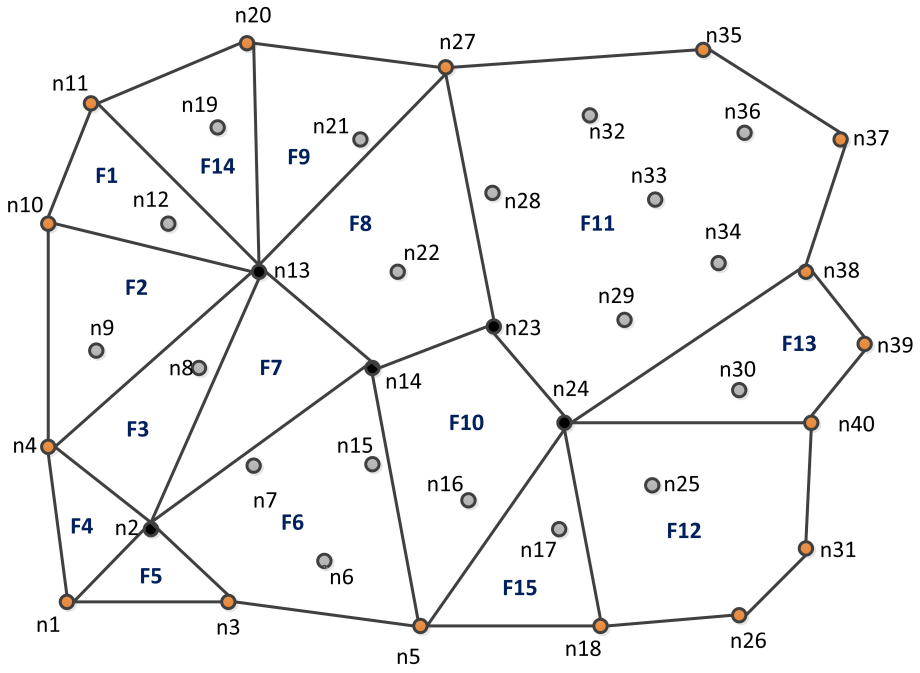


FIGURE 3.11: Proposed Face based WSN

3.5 Face Discovery

After the face structure is built, every node knows the information of its immediate neighbors N_{im} but is not aware of its spatial neighbors N_{sp}. Immediate neighbors are the neighbors to which a node has made edges. These are also called direct neighbors. The number of faces, the nodes is adjacent to is equal to the number of edges of nodes. All nodes in adjacent faces of a node are called its spatial neighbor nodes. To get the information of these nodes a node needs to run a face discovery protocol [37] in which

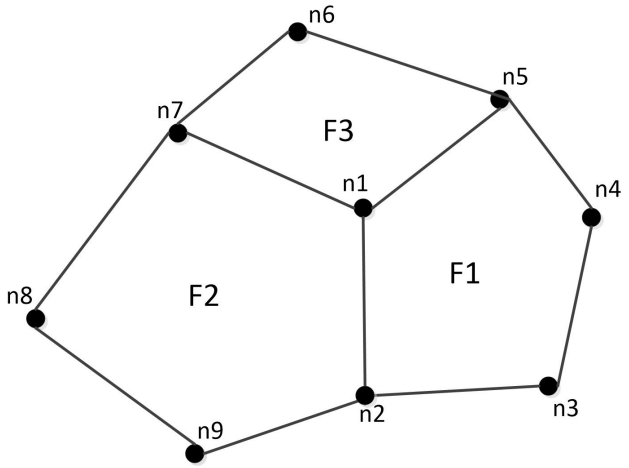


FIGURE 3.12: Gabriel graph based Face Structure

the information of all the nodes in adjacent faces is gathered. For further detail of face discovery protocol refer to Appendix B.

As shown in Figure. 3.12, node $n1$ has three adjacent faces $F1, F2, F3$ and has three immediate neighbors $n2, n5, n7$. All the nodes in adjacent faces are its spatial neighbors that are $n2, n9, n8, n7, n6, n5, n4, n3$.

3.6 Proposed Target Tracking Protocol

Before the target tracking is triggered, all the nodes in network that are part of face topology are in a state of duty cycle to conserve energy as shown in Figure. 3.13. A duty cycle is a sleep-awake periodic state where nodes wake up periodically to check for any incoming messages and if there is no message, they go back to sleep to save energy [50]. Otherwise, if the nodes remain active all the time, it will cause unnecessary energy consumption. First step in target tracking is to detect the presence of the target when it has entered the network. When to start detecting the target depends on the nature of application. For applications, that require the detection of target as soon as it enters the network, boundary nodes of the network can be kept active all the time so, as soon as, the target crosses the boundary of the network, it is detected. However, this will incur

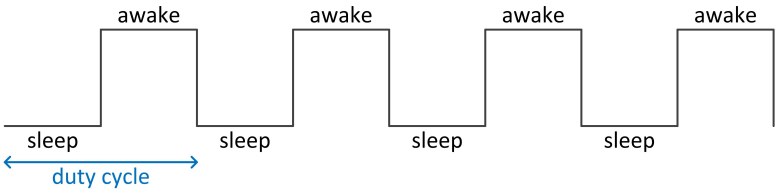


FIGURE 3.13: sleep awake periodic state of nodes

more energy consumption as the boundary nodes will remain active, and their energy will be drained quickly. Otherwise, all the nodes in the network are in sleep-awake periodic state where they become active upon receiving a message. A mobile sink/ tracker asks the network to find the target, and all the nodes become active to detect the target. The nodes that detect the target, start the tracking process and remaining nodes go back to sleep-awake periodic state.

The proposed target tracking protocol has following characteristics.

- Due to the fact that each edge is covered by two nodes, a target cannot cross an edge without being detected by the two nodes on the crossing edge.
- No prediction mechanism is required.
- Only the nodes of face in which target is located cooperate with each other to track the target.

The proposed target tracking protocol does not require prediction to track the target as the face envelops the target in which the target is located. The target can move in any direction by crossing any edge of the face. Since all the nodes in face in which target is located will be active, and all edges are covered, the target can't escape the face undetected. As shown in Figure. 3.14, target can leave the face from any of its edges $e1, e2, e3, e4, e5, e6$. For example, if the target leaves the face by crossing the edge $e1$, both $n1$ & $n2$ will detect the target crossing the edge. Hence, in this scenario, no prediction mechanism is required.

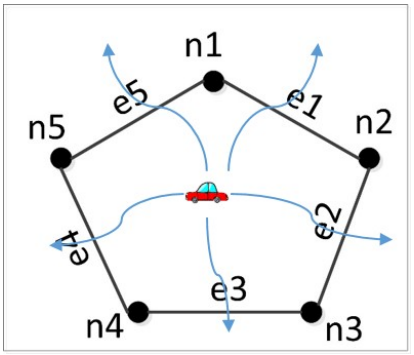


FIGURE 3.14: Target leaving a face

Mathematical notations used in target tracking protocol are given in Table. 3.2.

Notation	**Meaning**
F	Face
B	beacon node
C	coop node
ctp	coop track packet
bup	beacon update packet

TABLE 3.2: Mathematical Notations for Target Tracking

3.6.1 Target Detection

When target detection is triggered, a node becomes active and starts sensing the environment. If the target moves into the sensing range of a node it will be detected by the node. The location of target can be calculated using a distributed target localization technique [51, 52]. The nodes that do not sense any target in their vicinity, go back to default duty cycle. The nodes that are sensing the target check if they are the nearest node to the target. Since a node has the location information of its neighbor nodes, it can calculate if it is the nearest node to the target. Among all the nodes that are sensing the target, the node that is the nearest node to the target becomes the representative/beacon node (B) and the nodes that are detecting the target but are not the nearest node go back to the default duty cycle state. A beacon node is responsible for asking the nodes to cooperatively track the target and communicating with mobile sink node.

The beacon node will detect the face in which the target is present using a face detection algorithm. The face detection algorithm is designed using the "point in polygon" problem [53] from computational geometry. In this problem, it is found if a point lies inside a polygon or not. Since a node has the information of its adjacent faces, it will take its faces as polygons and target as point and find out in which face the target is present by solving the point-in-polygon problem for each face. When the face in which the target is present is found, the target detection process is complete. Next, the beacon node will start the target tracking process by asking the nodes in face, in which target is present, to become active and start tracking the target.

Algorithm. 3 shows target detection process where each node senses the environment and the node nearest to the target becomes beacon node and remaining nodes go back to default duty cycle state.

Algorithm 3 Target Detection

Input: ∅
Output: selection of beacon node

```
1: if sensing the target is true then
2:    if nearest node is true then
3:       if beacon node is true then
4:          face_detection()
5:          target_tracking()
6:       else
7:          default duty cycle state
8:       end if
9:    else
10:      default duty cycle state
11:   end if
12: else
13:   default duty cycle state
14: end if
```

3.6.2 Target Tracking

Once the beacon node has detected the face in which target is present, it forwards a coop_track_packet (*ctp*) in the face to ask the nodes in that face to cooperatively track the target. As shown in Figure. 3.15a, the target is detected in face F1 by beacon node $B1$ and it sends a *ctp* in face F1. The *ctp* travels in the face using right hand rule as explained in Appendix B. The nodes that receive this packet become active, update their status as coop nodes (C) and start tracking the target. A coop node is a node which tracks the target on the request of beacon node. This way the whole face has completely trapped the target and it cannot leave the face without being detected.

The target is tracked by beacon and coop nodes at periodic intervals. At each interval the location of the target is taken and is stored in a list by each tracking node. The current and previous location of target forms a small segment of target trajectory and each node checks the intersection of path segment with its two adjacent edges to the face containing the target. If at any point the target is leaving the face through an edge e, the intersection of target trajectory segment with edge e will be detected by the nodes to which that edge belongs. As shown in Figure. 3.15b, the target leaves the face by crossing the edge $e4$ and the intersection of target path with edge $e4$ is detected by the nodes $C3$ & $C4$.

Once the target leaving the face by crossing an edge is detected, one of the nodes of the edge that is nearest to the target will become new beacon node. The new beacon node will send a beacon_update_packet (*bup*) in the old face which the target left. The coop nodes upon receiving the *bup* packet will give-up their coop status and stop tracking the target and go back to sleep-awake periodic state. The old beacon node upon receiving the *bup* packet will give up its beacon status and stop tracking the target. Since the *bup* packet is sent by the new beacon node, it will store it as next beacon node. It will remain active for the purpose of communicating with the mobile sink node when it inquires about

the target. When the sink will reach this beacon node, it will guide the sink to the next beacon node.

On the crossing edge $e4$ in Figure. 3.15b, among the nodes $C3$ & $C4$, $C4$ is the closest to the target, hence it becomes the new beacon node $B2$. $B2$ sends a *bup* packet in face F1 as shown in Figure. 3.15c. $B1$ will store $B2$ as next beacon node.

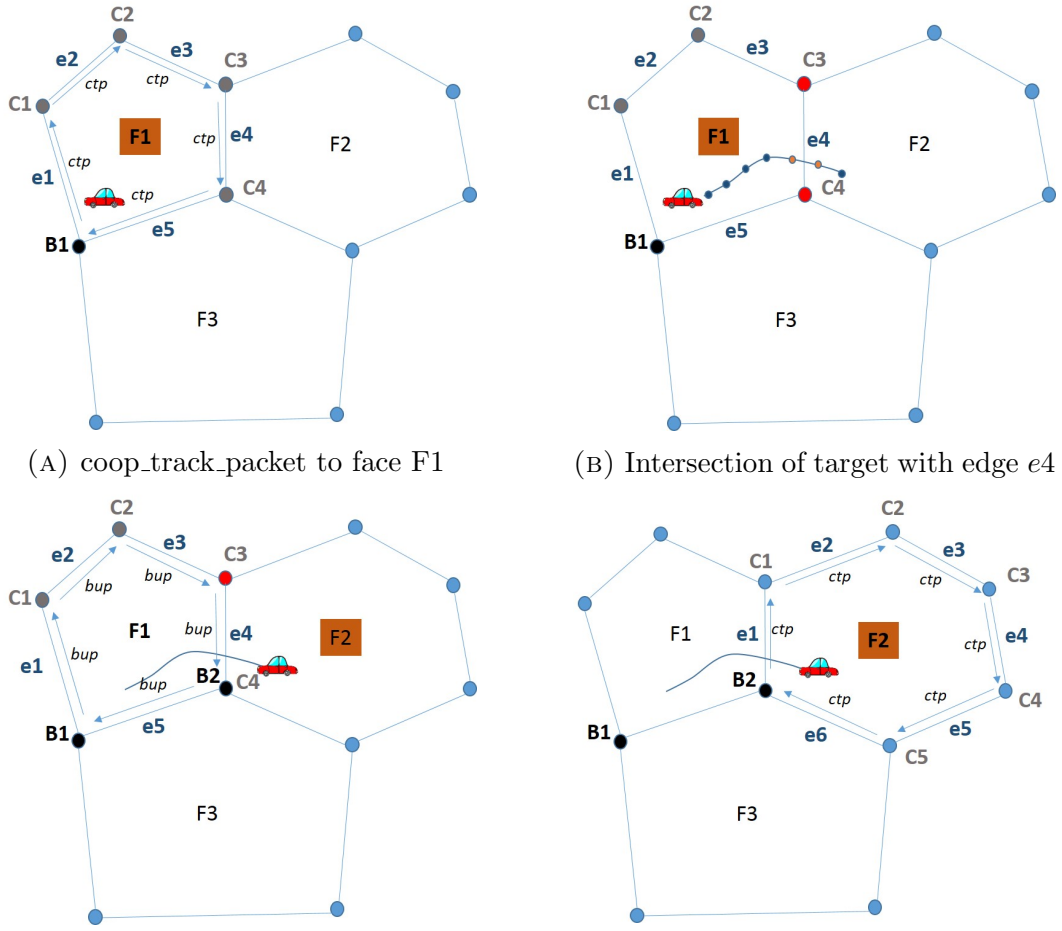
The new beacon node will send a coop_track_packet (*ctp*) to the new face in which the target is present now. The nodes in new face upon receiving *ctp* packet will become active, update their status as coop nodes and start tracking the target. As shown in Figure. 3.15d, new beacon node $B2$ sends *ctp* packet in face F2 and all the nodes in face F2 become coop nodes.

Algorithm. 4 shows the process of target tracking. Target tracking is a process of sensing the target until it is present in the sensing range and reading its location at set intervals. While the node is sensing the target, it checks for the intersection of target trajectory with its edges adjacent to the current face. If the target crosses and edge the intersection will be detected. If the node that detected the intersection is nearest to the target, it becomes the new beacon node and sends a beacon_update_packet in the previous face to inform the nodes to stop tracking the target and a coop_track_packet in current face to ask the nodes to start tracking the target.

Algorithm 4 Target Tracking

```
1: while sensing the target do
2:    store the location of target in memory
3:    check for intersection with edges adjacent to current face
4:    if intersection is true then
5:       if nearest node to target then
6:          beacon node is true
7:          send BEACON_UPDATE_PACKET to previous face
8:          send COOP_TRACK_PACKET to current face
9:       end if
10:   end if
11: end while
```

When a mobile sink will enter the network, it will go to first beacon node and will ask it about the location of the target, if the beacon node has target in its vicinity, it will inform the sink node about target's location else it will provide the information of next beacon node. Mobile sink will follow the trail of beacon nodes until it reaches the last beacon node which still have the target in its sensing range, hence the sink node will eventually capture the target.



(A) coop_track_packet to face F1

(B) Intersection of target with edge $e4$

(C) beacon_update_packet to face F1

(D) coop_track_packet to face F2

FIGURE 3.15: Target tracking scenario

An overview of target tracking protocol in proposed face structure is shown in Figure. 3.16. The target moves in the network in random path and is tracked by faces F5, F6, F10 and F11 along its journey. In each face, the nearest node to the target was selected as beacon node. $n1$ in F5, $n2$ in F6, $n14$ in F10, and $n23$ in F11 acted as beacon nodes.

Finally, when the target was leaving the network, $n23$ became the beacon node, but since the target had left the network, it could not be detected in any face and hence, $n23$ concluded that the target has left the network.

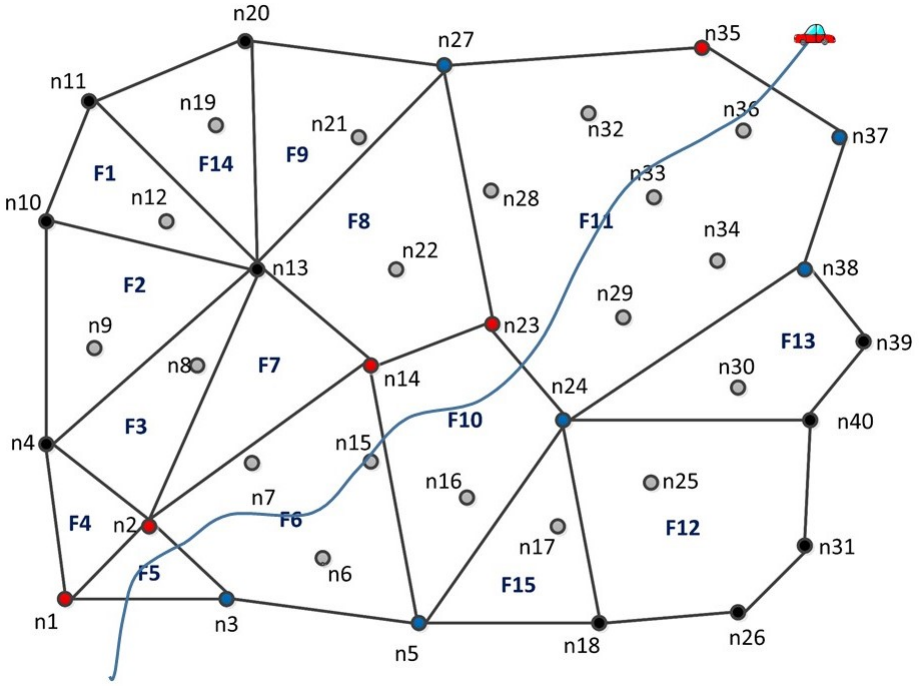


FIGURE 3.16: Overview of target tracking in proposed face-based WSN

3.7 Proposed Fault Tolerance Protocol

Due to the limited battery of sensor nodes, it will deplete after some time. When the battery of a node is depleted completely, it will die and its edges will break too causing partitioning in the network. The time of failure of a node due to battery depletion can be predicted ahead of time by monitoring battery voltage. Hence, the fault tolerant mechanism can be triggered before the complete failure of node. Since we have sleep nodes in network it is a good idea to use them to replace the faulty node. The fault tolerant mechanism makes use of the sleep nodes in network as replacement nodes to keep the network functioning for as long as possible.

Two battery threshold levels are defined, first threshold level ($TL1$) and second threshold level ($TL2$), where ($TL2 < TL1$), for complete execution of fault tolerance protocol.

When the battery level of node reaches below first threshold level ($TL1$), the nearest sleeping node from one of its faces is selected to be the replacement node and a "replacement node packet" is sent asking the sleeping node to become the replacement node. The dying node will also send the information of its edges in the "replacement node packet".

The sleeping node will wake up upon receiving the "replacement node packet" and will store the information of edges of dying node sent in the packet. It will remain awake and will wait to receive another message from the dying node.

Once the battery level reaches below second threshold level ($TL2$), it break its own edges and informs its neighbor nodes to break edges with it, sends another message to the selected replacement node and then goes to sleep. The replacement node on receiving the second packet will make edges using the edge making algorithm. It will only consider the nodes with which the dead node had edges with and the information of these nodes was sent in replacement node packet.

Two battery threshold levels are used so that, a node has enough time to select a replacement node and send it the information of its edges. After the second threshold level, a node has enough time to inform its immediate neighbors to break edges.

The notations used in proposed fault tolerance protocol are given in Table. 3.3.

3.7.1 Replacement Node Selection

A node N_d is monitoring its battery at small time intervals Δt. When the battery level of the node reaches below $TL1$, it searches for the nearest sleeping node in its sensing range. Since each node stores the information of its neighbors in the form of a neighbor

Notation	Meaning
Δt	Time interval of battery monitoring
$TL1$	First battery threshold level
$TL2$	Second battery threshold level
N_s	Nearest sleeping node
R_p	Replacement Node
N_d	Node whose battery is depleting
E_d	List of edge nodes of N_d
$E_d[i]$	A node in E_d

TABLE 3.3: Mathematical Notations for Fault Tolerance

table, it sorts all the sleeping nodes from the neighbor table and finds the nearest sleeping node N_s. It is possible that a node does not have any sleeping node in its range, in that case, selection of a replacement node is not possible. Once the nearest sleeping node N_s is selected, N_d has to check if N_s lies in one of its adjacent faces. Since N_d knows the location of N_s, this is a *point in a polygon* problem, where N_s is a point and adjacent faces of N_d are polygons. In *point in a polygon* problem, it is checked if a point lies inside a polygon [53]. If N_s lies in one of the adjacent faces of N_d, it is selected as a replacement node R_p. Otherwise, N_s cannot be selected as R_p because it will not be able to make edges with edge nodes E_d of N_d.

Upon successful selection of R_p, N_d will send a "replacement_node_packet" to R_p containing the information of its edge nodes E_d. R_p will wake up on receiving this packet and stores the received data in its memory and waits for further messages from N_d.

Figure. 3.17 demonstrates the selection of R_p in case of battery depletion of $n7$. $n7$ selects $n5$ as the nearest sleeping node. $n5$ belongs to face $F3$ which is the adjacent face of $n7$, hence it is selected as replacement node. $n7$ sends "replacement_node_packet" to $n5$ containing the information of its edge nodes $[n6, n3, n9]$.

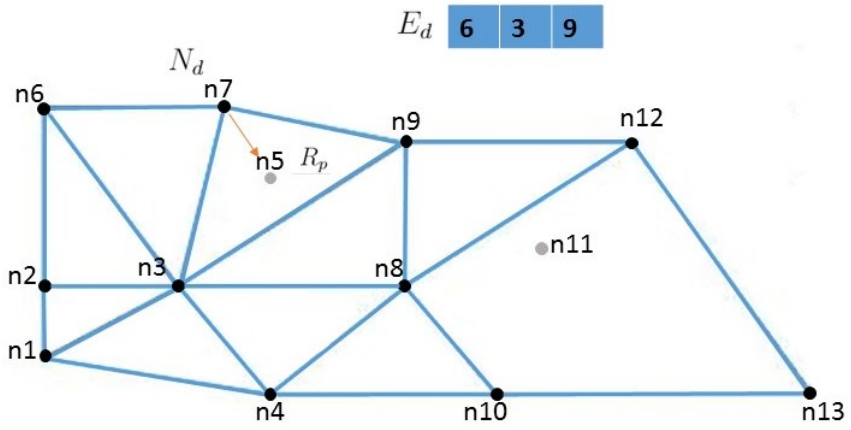


FIGURE 3.17: Example of Replacement Node Selection

3.7.2 Edge Making Algorithm

An edge making algorithm is used by replacement node R_p to make edges. R_p has a list of nodes E_d to make edges with provided by N_d. R_p will try to make edges with each node in the list E_d one by one.

The replacement node cannot decide the validity of edges on its own as once it was put to sleep it had no information regarding the surrounding area or neighbor nodes. So it sends the "is valid_edge_packet" to each node in E_d one by one. The recipient nodes will check the validity of edge with replacement node using the edge validity algorithm and will reply with an "edge_validity_reply_packet" indicating the validity of edge with R_p.

Before replying to replacement node, the recipient node runs a two-step process; 1) edge validation step and 2) edge reformation step. In the first step, it is checked if the edge is valid or not. The second step will be performed in case the edge is valid. In the second step, it is checked if some reformation of edges is required or not. We may need to reform some edges i.e. break old edges to make new edges.

A flowchart of edge making algorithm is shown in Figure. 3.18.

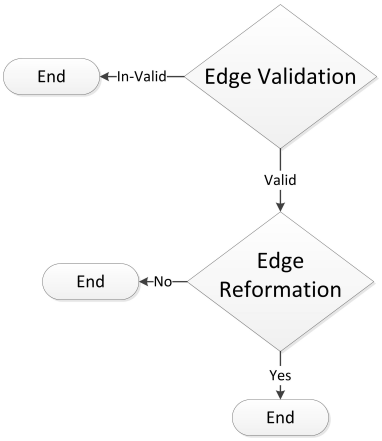


FIGURE 3.18: Edge Making Algorithm

3.7.2.1 Edge Validation Step

The edge validity criteria is very similar to the edge validity criteria of edges in proposed face structure i.e. an edge is valid if there is no active node that has already made edges inside the circle formed by edge as diameter.

The R_p node will ask all nodes in E_d, one by one, if the edge of R_p is valid with these nodes. The edge of replacement node R_p with $E_d[i]$ is valid if there is no node with edges inside the circle formed by the edge as diameter, otherwise, the edge with replacement node is invalid. $E_d[i]$ will check the possibility of edge with R_p because it has the information of all the nodes in neighborhood.

As shown in Figure. 3.19, $E_d[i]$ checked for any of its edges with immediate neighbors I_N comes inside the circle formed by the edge as diameter. In Figure. 3.19a, the edge e is valid because $I_N[i]$ doesn't lie inside the circle formed by edge e as diameter. On contrary, the edge in Figure. 3.19b is invalid.

An example of edge validation is shown in Figure. 3.20. Node $n5$ asks $n6$ if the edge between $n5$ and $n6$ is valid. Node $n6$ checks if any of its edge nodes $E_d[n2, n3]$ come inside the circle formed by the possible edge as diameter. Since no nodes come inside the circle, the possible edge is valid. $n6$ replies $n5$ with the validity of the edge.

(A) Edge valid with R_p (B) Edge invalid with R_p

FIGURE 3.19: Edge validation step

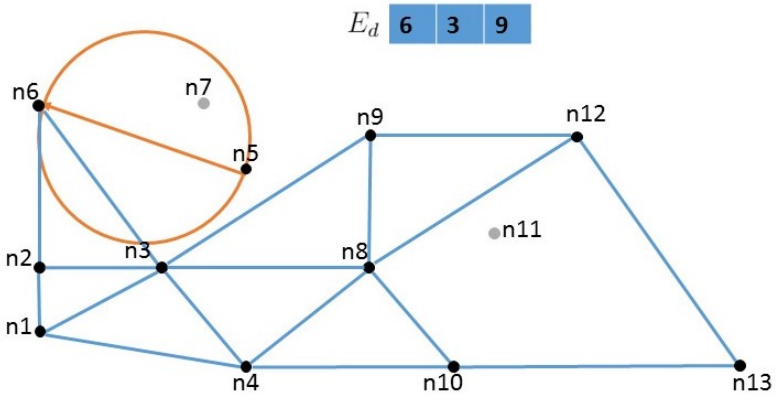


FIGURE 3.20: Edge Validation Example

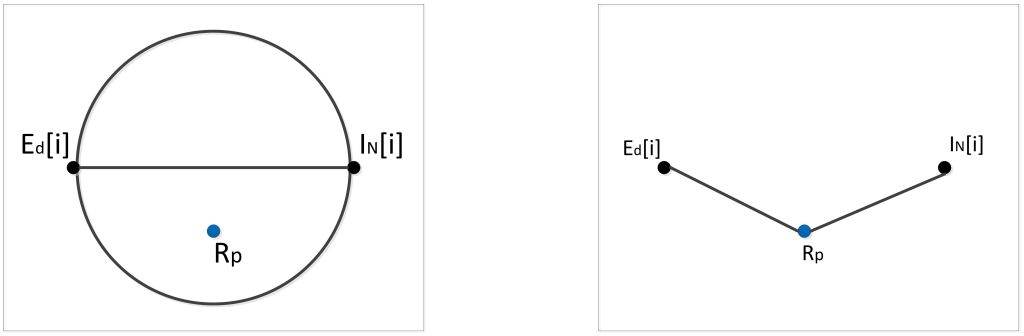
3.7.2.2 Edge Reformation Step

In case the edge between $E_d[i]$ and R_p is valid, the edge reformation step will be performed in which it will be checked if any reformation of edges is required. $E_d[i]$ checks if R_p is coming inside the circle formed by any of the edges of $E_d[i]$ with its immediate neighbors. In that case, already existing edges of $E_d[i]$ are becoming invalid, hence the reformation of edges will be required.

Before, the replacement node R_p was a sleep node and had no effect on any of the edges, but now it is active and has made edges, so the validity of other edges around it need to be checked again. If an edge of $E_d[i]$ with its immediate neighbor has become

invalid, it will be broken and $E_d[i]$ and its immediate neighbor node will be connected R_p.

The criteria of edge reformation is shown in Figure. 3.21. In Figure. 3.21a, node $E_d[i]$ is checking for reformation of its edge with its immediate neighbor node $I_n[i]$. R_p lies inside the circle formed by the edge as diameter. Hence this edge is invalid. This edge will be broken and a new path will be made through R_p as shown in Figure. 3.21b.



(A) Edge reformation required (B) Edge reformation

FIGURE 3.21: Edge Reformation Step

An example of edge reformation is shown in Figure. 3.22. Node $n6$ after deciding that edge with replacement node $n5$ is valid, checks if any reformation of its edges is required. $n6$ has edges with nodes $n2\&n3$. It checks if $n5$ lies inside the circle formed by any of its edges as diameter. Since, $n5$ does not lie inside any of the circles, no reformation is required.

Final result of fault tolerance protocol example is shown in Figure. 3.23. When node $n7$ was about to die, it selected node $n5$ as replacement node. Node $n7$ had edges with nodes $n9, n3$ and $n6$. Node $n5$ successfully made edges with these nodes. $n3$ had an edge with $n8$ which broke during edge reformation and $n8$ connected to replacement node $n5$.

The flowchart of fault tolerance protocol is shown in Figure. 3.24. A node keeps monitoring its battery levels until it reaches below first threshold level $TL1$. At this point, the node tries to select a replacement node. If a replacement node is successfully selected,

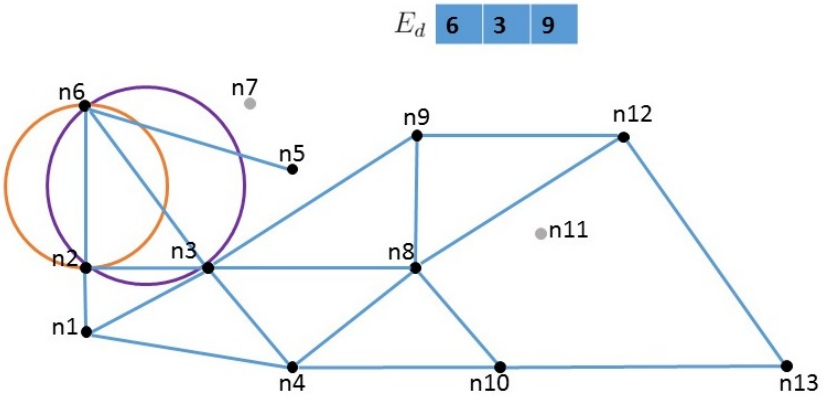


FIGURE 3.22: Edge Reformation Example

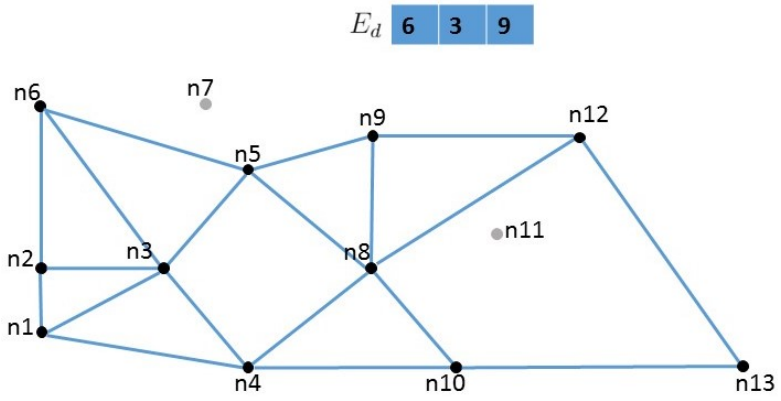


FIGURE 3.23: Final Result

the node sends a "replacement_node_packet" to R_p. The node again starts monitoring its battery, until it reaches below a second threshold level $TL2$. At this point the node breaks its edges, sends "break_links_packet" to its edge nodes I_N and "make_links_packet" to R_p and turns its radio off. If no R_P was selected, the node will follow all the steps except sending "make_links_packet" to R_p node.

After edge validation and edge reformation steps are complete, the final step is the face discovery of newly formed faces. The replacement node $n5$ will run the face discovery procedure and discover its faces, which are [$n5$, $n3$, $n6$] and [$n5$, $n3$, $n9$, $n8$]. Then $n5$ will send a face information packets in its faces to inform the nodes in faces about the discovery of new faces.

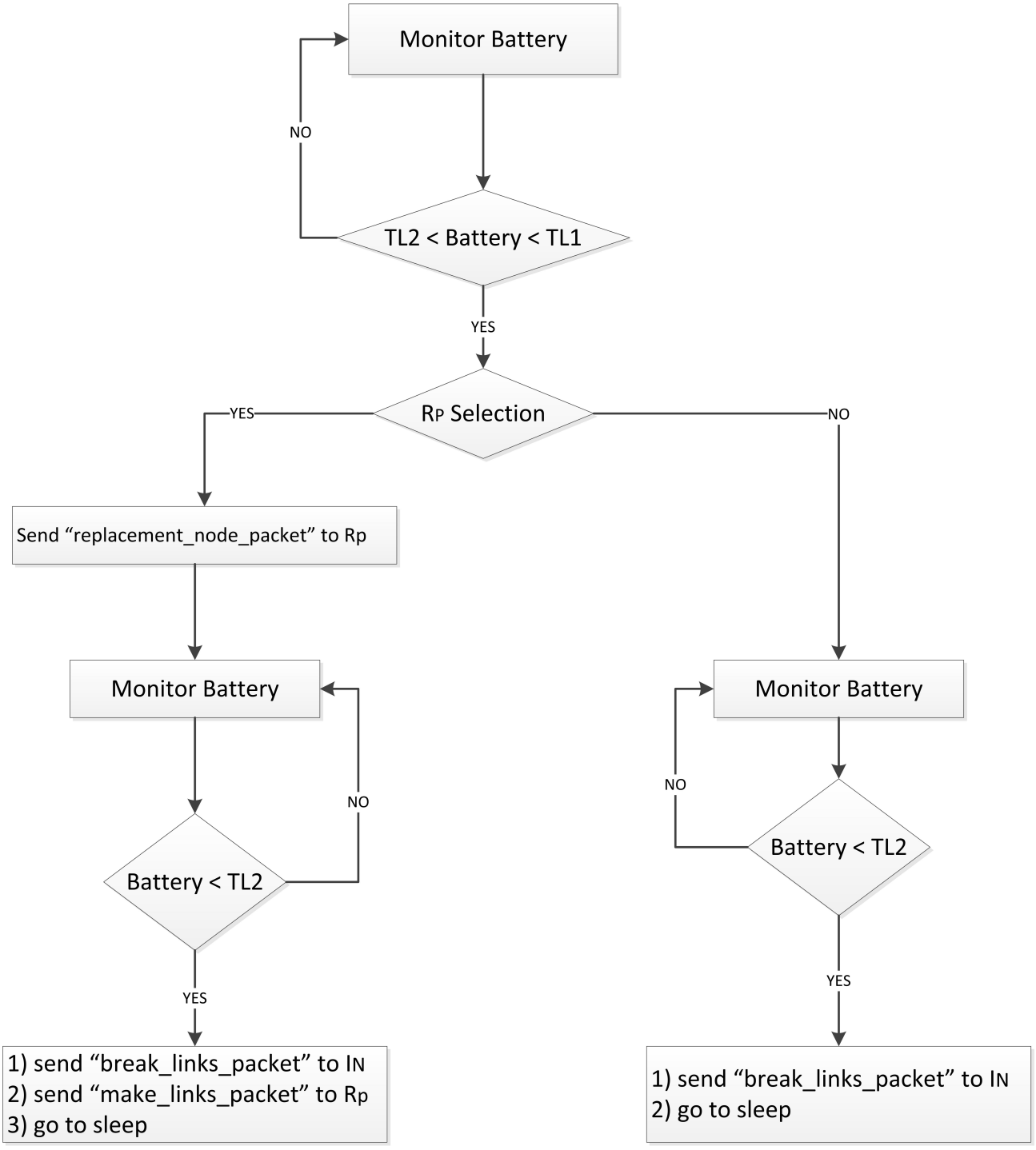


FIGURE 3.24: Flow chart for fault tolerance

Chapter 4

Results

In this chapter the performance of proposed face-based tracking (PFT) protocol is compared with dynamic object tracking (DOT) protocol [35] and Face Track protocol [36]. The proposed work and related work have been evaluated through extensive simulations in Castalia-3.2 framework [54] on OMNeT++ v3.4 network simulator. The proposed face-based tracking (PFT) protocol and fault tolerance protocol is simulated on proposed face structure (PFS) and DOT [35] and Face Track [36] are simulated on Gabriel-based face structure (GBFS). The performance of these approaches is compared in terms of energy consumption, lifetime and accuracy.

4.1 Performance Metrics

Performance is evaluated by using following performance metrics:

1. Energy Consumption: Average energy consumption in building the face topology and running target tracking application.

2. Lifetime of Network: The average amount of time until the network dies. [55]

3. Accuracy of Target Tracking: Tracking accuracy is determined as number of successful tracking events out of total number of tracking events.

4.2 Simulation Settings

The nodes are deployed in a network of 40m x 40m in 2D square plane in a random distribution manner. Face structure is built on 50, 100, 150 and 200 sensor nodes to see the number of sleep nodes for various densities of the network. The wireless communication links between sensor nodes are perfectly bidirectional i.e. two nodes can directly communicate with each other. A target in the sensing range of node can be perfectly localized using a target localization protocol. All the nodes in network have same communication and sensing range. The communication range (r_c) of a node is set at greater than or equal to twice the sensing range (r_s) of node i.e. ($r_c \geq 2r_s$). In the simulations the sensing range of nodes is $5m$ and communication range of nodes is $10m$.

All nodes in network synchronize with the sink in first $1-10$ ms. The default speed of target is 1 m/s and default background noise is set to zero, i.e. no background noise.

The simulation begins by sensor nodes gathering information of neighbor nodes and then the face topology is built and face discovery protocol is run for face recognition. Initially 100s are set for topology construction phase and 10s are set for running face discovery protocol. At this point, the network is ready for target tracking. The target is introduced in the network at a random time and random location. The target moves in the network in a random path. Target tracking is triggered when the target is detected by the network. The experimental results are averaged over 100 simulation runs.

The initial energy of nodes is set to 50 J. For wireless communication between nodes CC2420 radio parameters are used in simulations. The CC2420 is IEEE 802.15.4 compliant RF transceiver that is low cost, low power and highly integrated solution

for robust wireless applications [56]. The radio works in three modes transmit (TX), listen (RX) and sleep. In transmission mode the power consumption is 42.24 mW at transmission level of -5 dBm. In listening mode, the power consumed is 62 mW, and minimum power consumption is in sleep mode i.e., 1.4 mW.

CSMA-CA [57] MAC protocol is used to implement the duty cycle. The other parameters of this protocol are kept at default values. The active nodes in network are at duty cycle of 1.0 (no duty cycle) i.e. they are listening all the time, the sleep nodes are at duty cycle of 0.1 which means they are awake only for 10% of the time during a cycle to listen to any incoming messages and sleep for remaining 90% of the time.

4.3 Simulation Results

Once the face structure is built, some number of nodes have become asleep depending on the density of the network. Density of the network is defined as number of nodes per unit area of the network [58]. More the density of the network, more will be the number of sleep nodes as shown in table 4.1.

Number of Nodes (n)	Sleep Nodes	Density of Network (n/m^2)
50	9	0.03125
100	15	0.0625
150	22	0.09375
200	29	0.125

TABLE 4.1: Number of Sleep Nodes for Different Densities

4.3.1 Evaluation of Energy Consumption and Network Lifetime

The average energy consumption and average lifetime of the network for the proposed face-based tracking (PFT) protocol is compared to the DOT [35] and Face track [36] protocol as shown in Figure. 4.1 and Figure. 4.2.

The energy consumption of the proposed protocol (PFT) is much less than Face track and DOT protocol. This is mainly due to the presence of sleep nodes in the network which consume the least amount of energy. Less energy consumption means nodes will be active for longer periods of time i.e. the life time of network will be improved. The energy consumption of DOT protocol is higher than Face Track protocol due to the involvement of all spatial neighborhood nodes in target tracking as compared to only one face tracking the target in Face Track protocol. Both these protocols do not have any sleep nodes in network hence the gap between our protocol and these protocols is larger as compared to gap between DOT and Face Track protocol.

The energy consumption varies with speed and the average energy consumption of face based target tracking is less than the DOT and Face Track protocol because of the presence of sleep nodes in network as shown in Figure. 4.3.

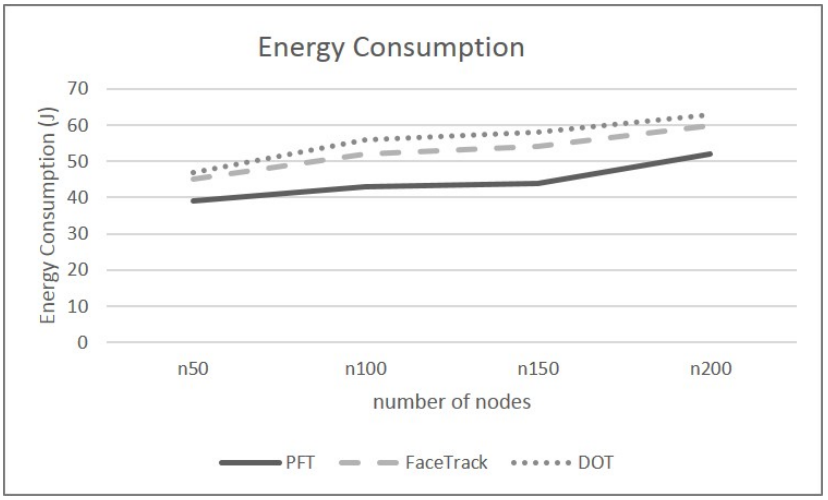


FIGURE 4.1: Average Energy Consumption of Network

4.3.2 Evaluation of Accuracy

We are defining the accuracy as the number of successful target tracking events over a total number of events. The total number of tracking events in each simulation is

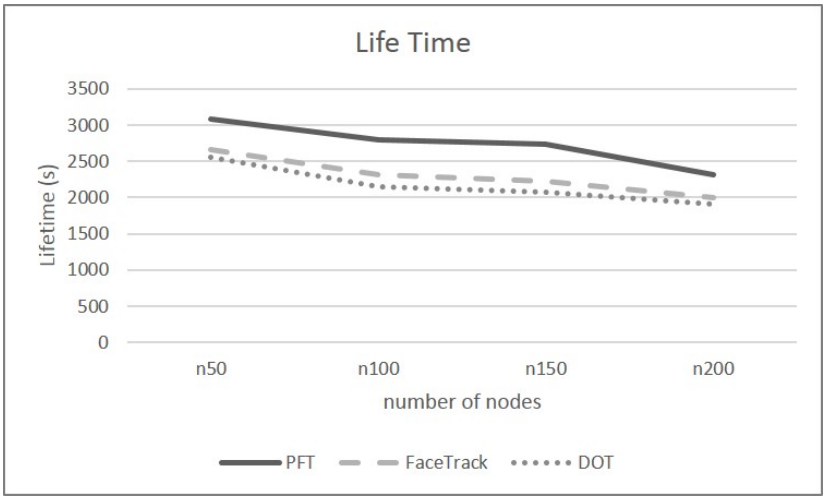


FIGURE 4.2: Average Lifetime of Network

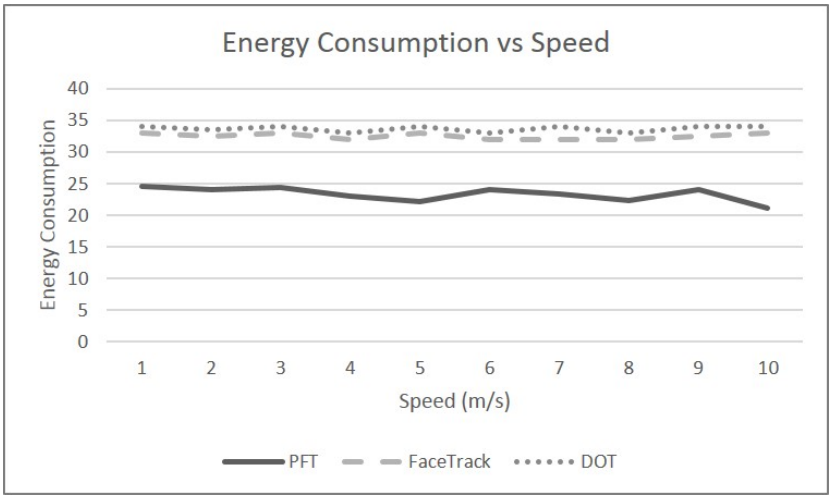


FIGURE 4.3: Energy Consumption vs. Target Speed

100. Many factors can affect the accuracy of the tracking such as, the speed of target, background noise in environment and duty cycles of nodes.

4.3.2.1 Effect of Speed on Accuracy

The effects of speed of target on accuracy is studied on low speeds (1-10 m/s) and high speeds (20-50 m/s). On low speeds, the effect on accuracy is almost the same for Face Track and proposed face-based tracking (PFT) protocol because in both of these protocols only the face in which the target resides is tracking the target as compared to DOT in

which all the spatial neighborhood nodes of the beacon node are cooperatively tracking the target as shown in Figure. 4.4.

At higher speeds (20-50 m/s), a target may escape a face very quickly before being tracked by the face resulting in loss of tracking. At such speeds, the accuracy of proposed face-based tracking (PFT) protocol becomes better than Face Track protocol, because the faces in proposed face structure are generally bigger in size than Gabriel face structure. Hence, the target will need more time escaping a bigger face, as compared to small faces. The effect of high speeds of target on accuracy is shown in Figure. 4.5.

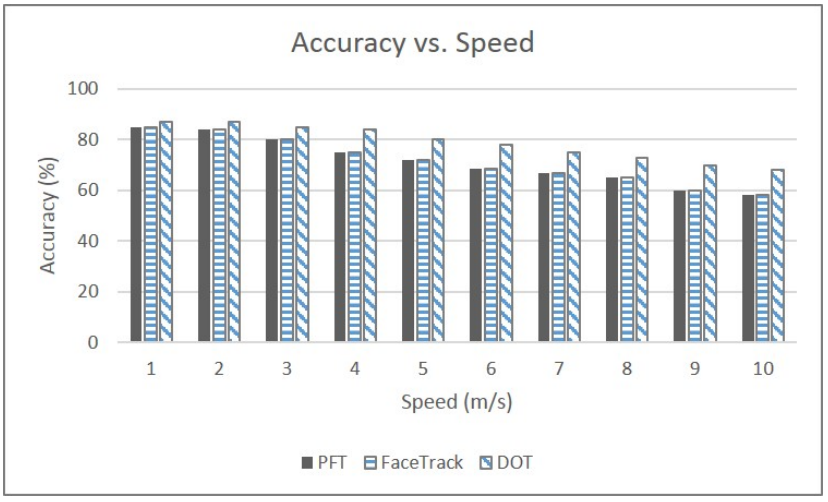


FIGURE 4.4: Accuracy of Tracking vs. Speed

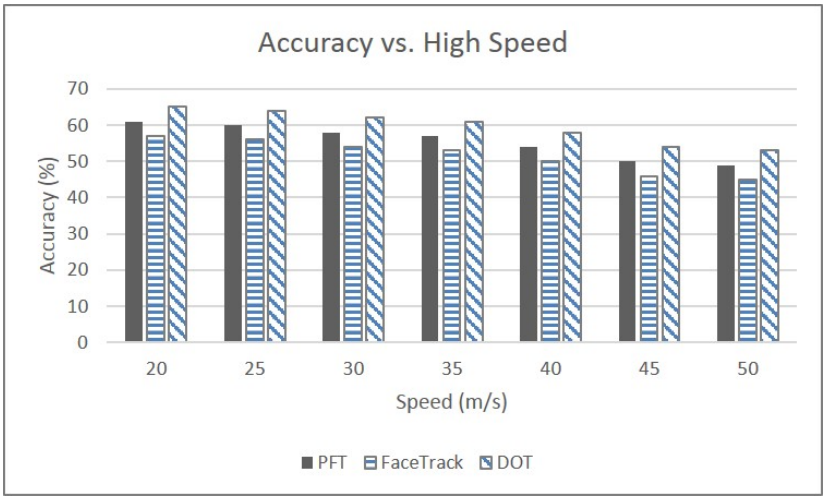


FIGURE 4.5: Accuracy of Tracking vs. High Speed

4.3.2.2 Effect of Duty Cycle on Accuracy

The duty cycles of nodes play a great part in determining the accuracy of the network. When the face topology is being built, all the nodes are fully active i.e., duty cycle of every node is 1.0. The nodes that are put to sleep are at duty cycle of 0.1. After the face topology is built, default duty cycle of the nodes that are part of topology can be varied from 0 to 1, to see its effect on accuracy. Once the target tracking is triggered, the nodes that are involved in target tracking are fully active and the remaining nodes are at default duty cycle to save the energy. The default duty cycle will impact the accuracy of target tracking as the nodes in future face will be sent a coop track packet to activate them and start tracking the target.

The trade off between accuracy and energy consumption can be achieved by setting an optimal value of duty cycle. For low duty cycles, the energy consumption will be low but the accuracy will also be less and for high duty cycles the accuracy will be high but the energy consumption will also be high.

As shown in Figure. 4.6, the accuracy for duty cycle of 0.1 to 0.5 is from 50% to 70%. For the duty cycle of 0.6 to 1.0 the accuracy of the tracking increases to a range of 70% to 90%. The accuracy of proposed PFT protocol and Face Track protocol is extremely similar due to the fact that both these protocols use only one face at a time to track the target. The DOT protocol has higher accuracy at all duty cycles. This is mainly due to the fact that, in DOT protocol the beacon node asks all its spatial neighbors to cooperatively track the target.

4.3.2.3 Effect of Background Noise on Accuracy

The background noise in WSNs in practical scenarios is never zero and it affects the performance of WSNs. Messages can be lost and communication links quality can vary

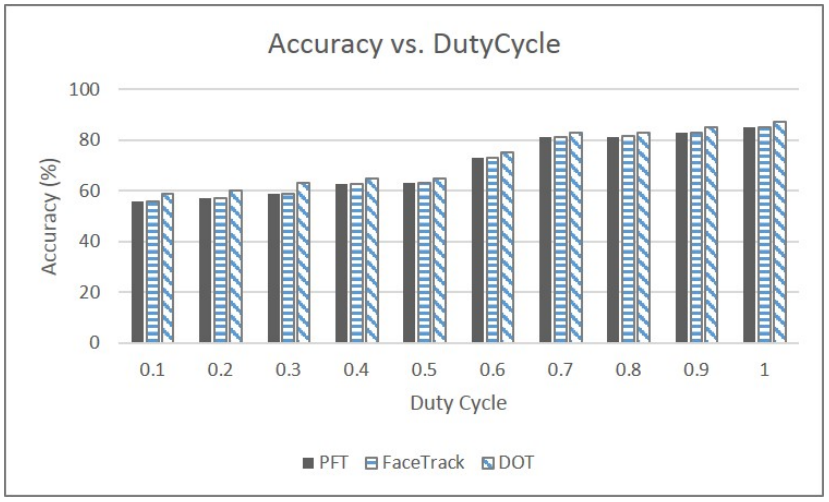


FIGURE 4.6: Accuracy of Network vs. Duty Cycle

greatly in presence of back ground noise.

All the experiments before are performed under ideal channel conditions i.e. the background noise is zero and the communication links between nodes are perfectly bidirectional. In this experiment, the protocol is simulated under more practical conditions for various levels of noise ranging from 1 to 6. The accuracy of tracking decreases with increasing levels of noise due to the occurrence of tracking errors as shown in Figure. 4.7. The accuracy of DOT protocol is still higher than our proposed protocol and Face Track protocol because of more number of faces tracking the target at the same time.

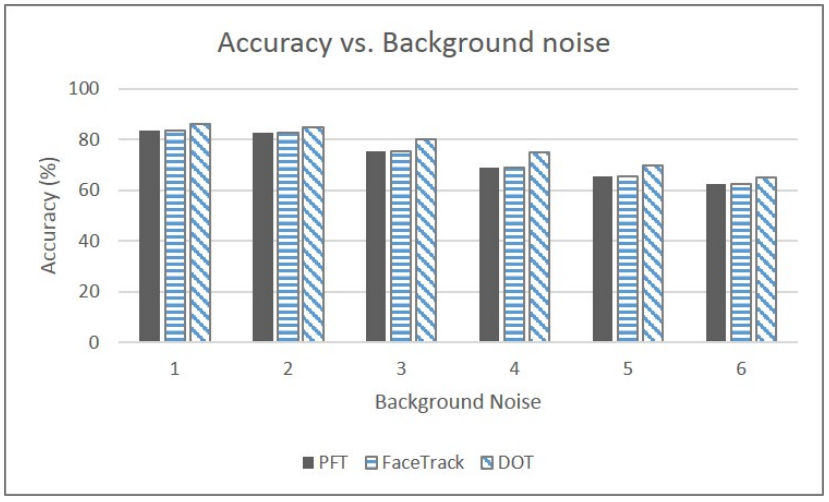


FIGURE 4.7: Accuracy of Network vs. Background Noise

4.3.3 Evaluation of Fault Tolerance

The fault tolerance protocol is implemented on proposed face structure. Its performance is analyzed in terms of improvement in lifetime as compared to when no fault tolerant protocol was implemented. When the fault tolerance mechanism is active in the network, the lifetime of the network improves as the node whose battery reaches a critical threshold level is replaced by selecting a nearby sleeping node as replacement node.

The first threshold level $TL1$ of node is set at 10% and the second threshold level $TL2$ is set at 5% of remaining energy. The network lifetime is increased in the range of 16% to 25% depending on the number of sleep nodes in the network. For 50 nodes the increase in lifetime is 16% and for 200 nodes the increase in lifetime is 25% as shown in Figure. 4.8.

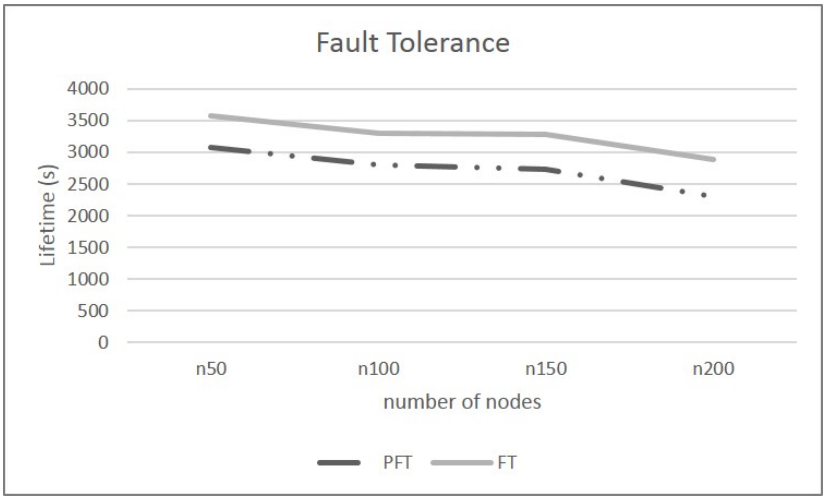


FIGURE 4.8: Increase in Lifetime with Fault Tolerance

Chapter 5

Conclusion and Future Work

A new distributed face structure is presented which is better than Gabriel based face structure (GBFS) in terms of energy efficiency and lifetime of the network. The proposed face structure does not consist of all the nodes in network, instead some nodes are put to sleep while ensuring the coverage of edges in the network. Due to the presence of sleeping nodes in the network the proposed face structure consumes lesser energy and prolongs the lifetime of network.

A distributed target tracking algorithm is designed for proposed face structure, utilizing the property of the face structure that the edges are covered. The edge coverage property ensures that a target leaving a face is always detected. In the proposed target tracking protocol, the node that is nearest to the target becomes beacon node and starts tracking the target. When the target is leaving a face by crossing an edge, the intersection of target path with the edge is detected by the two nodes on that edge and the node that is the closest among the two nodes becomes the new beacon node. The mobile sink node/ tracker communicates with the trail of beacon nodes to reach the last known location of target and capture it.

An energy efficient and fault tolerant distributed object tracking system is designed on proposed face structure (PFS) that tracks a moving object accurately with less energy consumption as compared to the existing GBFS based tracking protocols and maintains its accuracy even at high speeds while the performance of GBFS based tracking protocols decreases at high speeds due to the fact that the faces in proposed face structure are bigger in size and the target leaves a smaller face quickly as compared to a bigger face.

DOT protocol [35] has best accuracy but worst energy consumption. The accuracy of proposed face track (PFT) protocol and face track [36] protocol is comparable but the energy consumption of PFT protocol is lesser than face track protocol. Furthermore, the accuracy of PFT protocol becomes better than face track protocol at high speeds. Hence, in this study, a trade off between energy consumption and accuracy has been achieved.

Moreover, a distributed fault tolerant protocol is designed to use sleep nodes in network as replacement nodes in case of node failure to recover from faults. The proposed fault tolerance protocol, replaces a node whose battery reaches a critical threshold level with a nearby sleeping node. This approach maintains the network performance for as long as possible by avoiding the partitioning in the network.

The proposed protocol tracks only a single target in the network. In future, we will consider tracking multiple objects in the network and the presence of obstacles in the network.

The sensor nodes in the network are homogeneous. In future, the heterogeneous nature of sensors will be considered.

Appendix A

Face Structure

In wireless sensor networks (WSNs), face topology is built using Gabriel graph (GG) [59] or Relative Neighborhood graph (RNG) [27]. Both Gabriel graph (GG) and Relative Neighborhood graph (RNG) are planar graphs.

In Gabriel graph (GG), an edge *e(p, q)* exists between two nodes p and q if there is no other node r inside the disk formed by the edge as diameter as shown in figure A.1. In equation form it can be represented as follows:

$$\forall r \neq p, q : d^2(p, q) < [d^2(p, r) + d^2(q, r)] \tag{A.1}$$

In Relative Neighborhood graph (RNG), an edge *e(p, q)* exists between two nodes p and q if there is no node r inside the intersection of communication regions of both nodes as shown in figure A.2. In equation form it can be represented as follows:

$$\forall r \neq p, q : d(p, q) < max[d(p, r), d(q, r)] \tag{A.2}$$

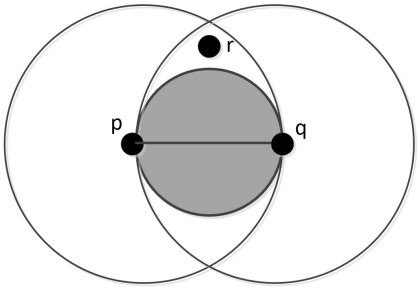


FIGURE A.1: Gabriel Graph

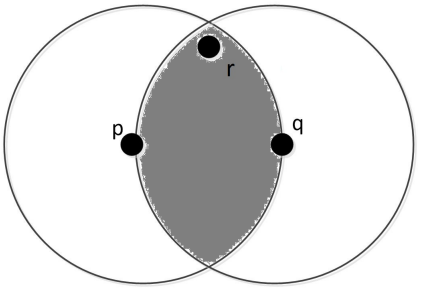


FIGURE A.2: Relative Neighborhood Graph

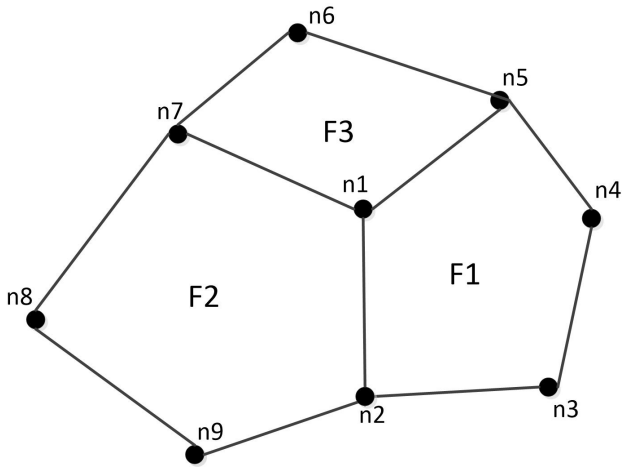


FIGURE A.3: Gabriel graph based Face Structure

In figure A.3, an example of face-based WSN is shown. In this figure, there are three faces $F1$, $F2$ and $F3$. $F1$ consists of nodes $n1$, $n2$, $n3$, $n4$, $n5$, similarly, $F2$ consists of nodes $n1$, $n2$, $n7$, $n8$, $n9$ and $F3$ consists of nodes $n1$,$n5$, $n6$ and $n7$.

The direct neighbors of a node are the nodes with whom it can communicate directly through the edges. The adjacent faces of a node are the faces with whom a node is directly attached through edges. The number of adjacent faces is equal to number of edges of a node. For example in figure A.3, node $n1$ has three direct neighbors $n2$, $n5$, $n7$ and three adjacent faces $F1$, $F2$ and $F3$.

To build Gabriel graph (GG) a node only needs the information of local neighbors. A node p checks the validity of Gabriel edges with all its neighbor nodes N. An edge is

valid if there are no nodes inside the circle formed by edge as diameter. The algorithm for building the Gabriel graph based face structure (GBFS) is as follows:

Algorithm 5 Gabriel graph based Face Structure (GBFS)

Input: Neighbor nodes N
Output: Validity of edges with N

```
1: for all q ∈ N do
2:    for all r ∈ N do
3:       take midpoint o of possible edge pq
4:       if d(o, r) ≤ d(o, p) then
5:          Edge is invalid
6:       else
7:          Edge is valid
8:       end if
9:    end for
10: end for
```

In algorithm 5, a node checks the possibility of edges with all its neighbor nodes N one by one. A node p takes a node q from its neighbor list N and checks the possibility of an edge with this node. The node p take the midpoint o of possible edge pq and checks if there is any node r inside the circle of radius op. If there is any node inside the circle the edge is invalid, otherwise, edge is valid.

The two target tracking protocols that are implemented for comparison purpose are implemented on Gabriel graph.

Appendix B

Face Discovery Protocol

In a face architecture, a node not only needs the information of immediate neighbors (N_{im}) but it also needs the information of all the neighbors in adjacent faces. The nodes in adjacent faces are called spatial neighbors (N_{sp}). To discover the information of spatial neighbors, a spatial neighborhood discovery protocol proposed in [37] is used.

Spatial neighbors are discovered after the face architecture is built. A node already knows the information of immediate neighbors (N_{im}) in a face structure. To discover the spatial neighbors (N_{sp}), every node sorts its immediate neighbors in anti-clockwise direction and stores them in a list called *ring buffer*. Then every node sends a neighborhood discovery message to its direct neighbors. The discovery message traverses in each face following *right hand rule*. According to the right hand rule, when a node receives a discovery message from a neighbor node, it appends its own id in the message, and forwards the message to the next node in ring buffer. The discovery message terminates when it reaches the node that initiated this message. At this point the complete face is discovered. Then another message traverses the whole face to inform all the nodes in face about face discovery. Each node in the face stores the information of discovered face in a face table.

The face discovery protocol is explained with an example. A face structure is shown in figure. B.1

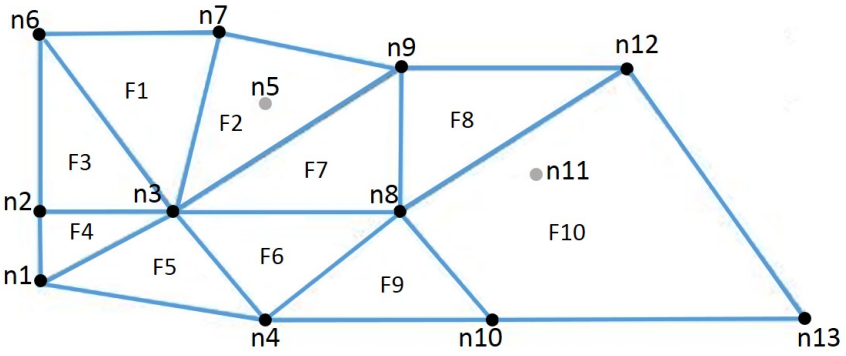


FIGURE B.1: Proposed Face Structure

First step of face discovery protocol is to build a ring buffer. Hence, every node will sort its neighbors first and store them in a ring buffer. For example, the ring buffer of node $n8$ will be $[n9, n3, n4, n10, n12]$. Node $n8$ send a discovery message towards each of its immediate neighbors as shown in figure. B.2. The messages traverse the face according to the right hand rule and are terminated when they reach back to node $n8$. At this point, node $n8$ has discovered all of its faces, which are [F6, F7, F8, F9, F10]. Nodes $n8$ stores the information of discovered faces in a face table as shown in figure B.3. Similarly, all the nodes in network will discover their spatial neighborhood nodes.

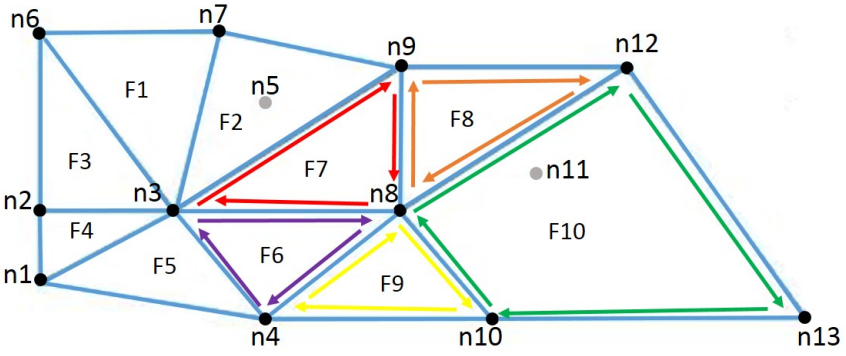


FIGURE B.2: Face discovery messages from $n8$

Face	Nodes in face
F6	[n4, n3]
F7	[n3 n9]
F8	[n9 n12]
F9	[n4 n10]
F10	[n10 n12 n13]

FIGURE B.3: Face table of $n8$

Bibliography

[1] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

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

[3] J. Albath, M. Thakur, and S. Madria, "Energy constraint clustering algorithms for wireless sensor networks," *Ad Hoc Networks*, vol. 11, no. 8, pp. 2512 – 2525, 2013.

[4] S. Bhatti, J. Xu, and M. Memon, "Energy-aware fault-tolerant clustering scheme for target tracking wireless sensor networks," in *7th International Symposium on Wireless Communication Systems*, pp. 531–535, 2010.

[5] O. D. Incel, A. Ghosh, B. Krishnamachari, and K. Chintalapudi, "Fast data collection in tree-based wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 11, pp. 86–99, Jan 2012.

[6] H. Lu, J. Li, and M. Guizani, "Secure and efficient data transmission for cluster-based wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, pp. 750–761, March 2014.

[7] Q. Huang, S. Bhattacharya, C. Lu, and G.-C. Roman, "Far: Face-aware routing for mobicast in large-scale sensor networks," *ACM Trans. Sen. Netw.*, vol. 1, pp. 240–271, Nov. 2005.

[8] F. Wang, X. Bai, B. Guo, and C. Liu, "Dynamic clustering in wireless sensor network for target tracking based on the fisher information of modified kalman filter," in *3rd International Conference on Systems and Informatics (ICSAI)*, pp. 696–700, Nov 2016.

[9] C.-Y. Lin, W.-C. Peng, and Y.-C. Tseng, "Efficient in-network moving object tracking in wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 8, pp. 1044–1056, 2006.

[10] X. Ji, Y. Y. Zhang, S. Hussain, D. X. Jin, E. M. Lee, and M. S. Park, "FOTP: Face-Based Object Tracking Protocol in Wireless Sensor Network," in *Fourth International Conference on Computer Sciences and Convergence Information Technology*, pp. 128–133, 2009.

[11] E. F. Nakamura, A. A. F. Loureiro, and A. C. Frery, "Information fusion for wireless sensor networks: Methods, models, and classifications," *ACM Comput. Surv.*, vol. 39, Sept. 2007.

[12] B. Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking - MobiCom '00*, pp. 243–254, 2000.

[13] A. Akl, T. Gayraud, and P. Berthou, "A metric for evaluating density level of wireless sensor networks," in *IFIP Wireless Days (WD)*, pp. 1–3, 2011.

[14] E. Shih, S.-H. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan, "Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks," in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, MobiCom '01, pp. 272–287, ACM, 2001.

[15] G. J. Wang, M. Z. A. Bhuiyan, J. N. Cao, and J. Wu, "Detecting Movements of a Target Using Face Tracking in Wireless Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 4, pp. 939–949, 2014.

[16] Y. Zhu, A. Vikram, and H. Fu, "On topology of sensor networks deployed for tracking," in *Wireless Algorithms, Systems, and Applications* (Y. Cheng, D. Y. Eun, Z. Qin, M. Song, and K. Xing, eds.), (Berlin, Heidelberg), pp. 60–71, Springer Berlin Heidelberg, 2011.

[17] O. Demigha, W. Hidouci, and T. Ahmed, "On Energy Efficiency in Collaborative Target Tracking in Wireless Sensor Network: A Review," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1210–1222, 2013.

[18] I. Banerjee, P. Chanak, H. Rahaman, and T. Samanta, "Effective fault detection and routing scheme for wireless sensor networks," *Computers & Electrical Engineering*, vol. 40, no. 2, pp. 291 – 306, 2014.

[19] M. K. Watfa and R. A. Assi, "Daim: A distributed algorithm for isolating malfunctioning nodes in wireless sensor networks," in *Advances in Education and Management* (M. Zhou, ed.), (Berlin, Heidelberg), pp. 523–530, Springer Berlin Heidelberg, 2011.

[20] M. B. L. de Souza, H. Vogt, "A survey on fault tolerance in wireless sensor networks," 2007.

[21] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong, "A macroscope in the redwoods," in *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, SenSys '05, (New York, NY, USA), pp. 51–63, ACM, 2005.

[22] S. Chouikhi, I. El Korbi, Y. Ghamri-Doudane, and L. Azouz Saidane, "A survey on fault tolerance in small and large scale wireless sensor networks," *Computer Communications*, vol. 69, pp. 22–37, sep 2015.

[23] M. Yu, H. Mokhtar, and M. Merabti, "Fault management in wireless sensor networks," *IEEE Wireless Communications*, vol. 14, pp. 13–19, December 2007.

[24] R. Beaubrun, "Methods for node localization in wireless sensor networks," in *IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems*, pp. 521–522, Oct 2014.

[25] S. Arora and S. Singh, "Node localization in wireless sensor networks using butterfly optimization algorithm," *Arabian Journal for Science and Engineering*, vol. 42, pp. 3325–3335, Aug 2017.

[26] L. Fang, W. Du, and P. Ning, "A beacon-less location discovery scheme for wireless sensor networks," in *Secure Localization and Time Synchronization for Wireless Sensor and Ad Hoc Networks* (R. Poovendran, S. Roy, and C. Wang, eds.), pp. 33–55, Springer US, 2007.

[27] G. T. Toussaint, "The relative neighbourhood graph of a finite planar set," *Pattern Recognition*, vol. 12, no. 4, pp. 261–268, 1980.

[28] M. Z. A. Bhuiyan, G.-j. Wang, L. Zhang, and Y. Peng, "Prediction-based energy-efficient target tracking protocol in wireless sensor networks," *Journal of Central South University of Technology*, vol. 17, no. 2, pp. 340–348, 2010.

[29] Y. Shen, K. T. Kim, J. C. Park, and H. Y. Youn, "Object Tracking Based on the Prediction of Trajectory in Wireless Sensor Networks," in *IEEE 10th International Conference on High Performance Computing and Communications & IEEE International Conference on Embedded and Ubiquitous Computing*, pp. 2317–2324, 2013.

[30] J.-M. Hsu, C.-C. Chen, and C.-C. Li, "POOT: An efficient object tracking strategy based on short-term optimistic predictions for face-structured sensor networks," *Computers & Mathematics with Applications*, vol. 63, no. 2, pp. 391–406, 2012.

[31] M. Z. A. Bhuiyan, G. Wang, and J. Wu, "Target Tracking with Monitor and Backup Sensors in Wireless Sensor Networks," in *Proceedings of 18th International Conference on Computer Communications and Networks*, pp. 1–6, 2009.

[32] M. Z. A. Bhuiyan, G. Wang, and A. V. Vasilakos, "Local Area Prediction-Based Mobile Target Tracking in Wireless Sensor Networks," *IEEE Transactions on Computers*, vol. 64, no. 7, pp. 1968–1982, 2015.

[33] T. S. Chen, C. H. Wu, and J. J. Chen, "Object Tracking by mining movement Trajectories in Wireless Sensor Networks," in *IEEE Network Operations and Management Symposium (NOMS)*, pp. 1–8, 2014.

[34] T.-S. Chen, J.-J. Chen, and C.-H. Wu, "Distributed object tracking using moving trajectories in wireless sensor networks," *Wireless Networks*, vol. 22, no. 7, pp. 2415–2437, 2016.

[35] H.-W. Tsai, C.-P. Chu, and T.-S. Chen, "Mobile object tracking in wireless sensor networks," *Computer Communications*, vol. 30, no. 8, pp. 1811 – 1825, 2007.

[36] G. Wang, M. Bhuiyan, J. Cao, and J. Wu, "Detecting movements of a target using face tracking in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 4, pp. 1–11, 2014.

[37] Q. Huang, C. Lu, and G. C. Roman, "Reliable mobicast via face-aware routing," in *Proceedings - IEEE INFOCOM*, 2004.

[38] S. Chouikhi, I. E. Korbi, Y. Ghamri-Doudane, and L. A. Saidane, "Fault tolerant multi-channel allocation scheme for wireless sensor networks," in *IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 2438–2443, April 2014.

[39] A. A. Abbasi, M. F. Younis, and U. A. Baroudi, "Recovering from a node failure in wireless sensor-actor networks with minimal topology changes," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 1, pp. 256–271, 2013.

[40] B. Khelifa, H. Haffaf, M. Madjid, and D. Llewellyn-Jones, "Monitoring connectivity in wireless sensor networks," in *IEEE Symposium on Computers and Communications*, pp. 507–512, July 2009.

[41] H. Liu, P. Wan, and X. Jia, "On optimal placement of relay nodes for reliable connectivity in wireless sensor networks," *Journal of Combinatorial Optimization*, vol. 11, pp. 249–260, Mar 2006.

[42] H. Liu, A. Nayak, and I. Stojmenović, *Fault-Tolerant Algorithms/Protocols in Wireless Sensor Networks*, pp. 261–291. London: Springer London, 2009.

[43] W. M. Elsayed, S. F. Sabbeh, and A. M. Riad, "A Distributed Fault Tolerance Mechanism for Self-Maintenance of Clusters in Wireless Sensor Networks," *Arabian Journal for Science and Engineering*, vol. 43, no. 12, pp. 6891–6907, 2018.

[44] M. A. Sheikh, M. Drieberg, and N. B. Z. Ali, "An improved distributed scheduling algorithm for wireless sensor networks," in *4th International Conference on Intelligent and Advanced Systems (ICIAS2012)*, vol. 1, pp. 274–279, June 2012.

[45] R. Zhao and X. Shen, "Broadcast protocols for wireless sensor networks, smart wireless sensor networks." http://www.intechopen.com/books/smart-wireless-sensor-networks/broadcast-protocols-forwireless-sensor-networks, 2010.

[46] J. S. Deogun, S. Das, H. S. Hamza, and S. Goddard, "An algorithm for boundary discovery in wireless sensor networks," in *High Performance Computing – HiPC 2005* (D. A. Bader, M. Parashar, V. Sridhar, and V. K. Prasanna, eds.), (Berlin, Heidelberg), pp. 343–352, Springer Berlin Heidelberg, 2005.

[47] R. Beghdad and A. Lamraoui, "Boundary and holes recognition in wireless sensor networks," *Journal of Innovation in Digital Ecosystems*, vol. 3, no. 1, pp. 1 – 14, 2016.

[48] A. M. Khedr and W. Osamy, "Minimum perimeter coverage of query regions in a heterogeneous wireless sensor network," *Information Sciences*, vol. 181, pp. 3130–3142, aug 2011.

[49] R.-H. Hwang, C.-C. Wang, and W.-B. Wang, "A distributed scheduling algorithm for ieee 802.15.4e wireless sensor networks," *Computer Standards & Interfaces*, vol. 52, pp. 63 – 70, 2017.

[50] R. R. Rout and S. K. Ghosh, "Enhancement of lifetime using duty cycle and network coding in wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 12, pp. 656–667, February 2013.

[51] A. Hajihoseini Gazestani, R. Shahbazian, and S. A. Ghorashi, "Decentralized Consensus Based Target Localization in Wireless Sensor Networks," *Wireless Personal Communications*, vol. 97, no. 3, pp. 3587–3599, 2017.

[52] R. Shahbazian and S. A. Ghorashi, "Distributed cooperative target detection and localization in decentralized wireless sensor networks," *The Journal of Supercomputing*, vol. 73, no. 4, pp. 1715–1732, 2017.

[53] D. Sunday, "Inclusion of a point in a polygon." http://geomalgorithms.com/a03-_inclusion.html.

[54] A. Boulis, "Castalia framework." https://github.com/boulis/Castalia, 2007.

[55] Y. Chen and Q. Zhao, "On the lifetime of wireless sensor networks," *IEEE Communications Letters*, vol. 9, pp. 976–978, Nov 2005.

[56] T. Instruments, "Cc2420." http://www.ti.com/product/CC2420.

[57] M. Khanafer, M. Guennoun, and H. T. Mouftah, "A Survey of Beacon-Enabled IEEE 802.15.4 MAC Protocols in Wireless Sensor Networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 856–876, 2014.

[58] S. Toumpis, "Mother nature knows best: A survey of recent results on wireless networks based on analogies with physics," *Comput. Netw.*, vol. 52, pp. 360–383, Feb. 2008.

[59] K. R. Gabriel and R. R. Sokal, "A New Statistical Approach to Geographic Variation Analysis," *Systematic Biology*, vol. 18, pp. 259–278, sep 1969.