Computational Aspects of Wireless Sensor Networks.

Md. Kamrul Islam
School of Computing, Queen's University
Kingston, Ontario, Canada K7L 3N6

April 10, 2007

Abstract

Wireless sensor networks, based on the infrastructure-less and ad hoc platform, consist of a number of small battery-powered sensors (also called sensor nodes) equipped with limited on-board processing, bandwidth, and sensing devices. In general, these tiny devices are deployed in hostile places forming an ad hoc network with a view to sensing and producing useful information from the environment and sending them to reliable base stations for further manipulation. As the lifetime of these sensors is dependent on their own limited battery power, special care must be taken to enhance their lifetime and therefore that of the network. The lifetime of sensor networks depends on application requirements. For every sensor network two main objectives need to be ensured: exploiting the sensors to obtain optimal information from the environment and increasing the lifetime of sensors. For applications which depend on the functioning of every sensor node, the lifetime of a network is defined as the span of time until the first node runs out of its power. Generally the definition of network lifetime can be expanded to the span of time until a subset of nodes runs out of their power. One of the possible and most promising ways to increase the lifetime span of a sensor network is to design distributed and localized algorithms for sensors that cleverly compute desired information expending less power and guaranteeing the delivery of desired information to the base stations. In this paper, we particularly focus on the computational and algorithmic aspects of localized algorithms that are expected to maintain a tradeoff between the enhancement of the lifetime of the sensor network and providing desired information.

1 Introduction and overview

The main task of a sensor network is to allow a collection of unattended sensors deployed at arbitrary locations in a remote environment to form a virtual network among them and provide cooperatively and collectively sensed data about some events of interest to the base stations (also called *sinks*). Because of their flexibility, efficacy, low-cost, accuracy, and ease of deployment, sensor networks are gaining attention and expanding their domain of applications. A wide spectrum of applications of sensor networks such as environmental monitoring (fires, floods, earthquakes, etc.), industrial sensing, infrastructure protection, intelligent homes, military battlefield surveillance, and so forth [36], have substantially motivated many researchers to continue their work with the purpose of meeting the challenges posed by these applications. The unique characteristics of sensor networks, that is,

the independent sensing capability of environment and data aggregation capacity of sensors and the infrastructure-less platform of the network, present new challenges to researchers in the design of efficient and sophisticated algorithms. These algorithms must be different from traditional algorithms used for wired and electrically-powered computer networks because of the ad hoc topology and underlying limited on-board power of sensors.

The study of sensor networks is an interdisciplinary research area. The research problems addressed by sensor networks invoke the problems of other disciplines such as computational geometry, signal processing, networking and protocols, micro data bases [35] and information management, distributed algorithms, embedded systems, random graphs, stochastic geometry, theory of combinatorics, and approximation algorithms [32], [38]. For example, many computational geometric algorithms for searching, convex hull, Voronoi diagram, and triangulations [37] can equally apply to wireless sensor networks as sensor nodes can be treated as points in a two-dimensional plane.

Due to the limited energy and bandwidth constraints of individual sensors, a direct application of these methods and techniques from different disciplines may not always be feasible; however, proper modifications to the techniques in light of the unique characteristics of sensors can provide solutions to numerous problems in sensor networks. In this paper, we study some techniques and methods from various fields and explore how modifications of these algorithms help solve different problems such as data aggregation, object tracking, area coverage, broadcasting, and topology control in sensor networks.

1.1 Characteristics of sensor networks

Sensor nodes are generally intended to deploy in harsh environments where human intervention is difficult or impossible. Hence, as implied earlier, the elements (sensor nodes) of sensor networks must have the capability of communicating in a wireless environment and, potentially, the ability to disseminate and process signals from a hostile environment. These requirements force the architecture of sensors to be compatible with such an environment.

A Wireless Sensor Network (WSN) is a network consisting of a collection of unattended battery-powered sensors which are not physically connected to each other to sense, gather, process, and transmit environmental data to the base station. The following characteristics are unique to a WSN [40].

1.1.1 Communication and sensing

Each sensor node has a certain communication capacity and the communication ranges of nodes can vary. The connectivity or the topology of a WSN depends on the available power of its nodes and it can change arbitrarily since two sensors are neighbors to each other or directly connected if they are within each other's communication range (radius). Communication can be symmetric (or bidirectional) if a node u can communicate with a node v, and v can communicate with u or it can be asymmetric (or unidirectional) if u can communicate with v but v cannot communicate with u. A node can monitor a circular area through some sensing modules embedded with it, where the radius of the observed circular area from the node is the sensing radius of the node. Typically, the communication radius (CR) is greater than the sensing radius (SR) [41].

1.1.2 Power consumption

As the sensors gather information from the environment, do some processing on them and transmit or receive signals, they consume energy. The most power is consumed when sensors transmit information and this power dissipation is a function of distance, that is, the general form of power dissipation is given by $P = d^{\alpha} + c$ where d is the distance between two sensors sending and receiving information, c is a technology-dependent constant, and α ($2 \leq \alpha \leq 6$) is the power-loss exponent [41]. Thus, it is intuitive that the limited power of sensors can only be used to sense the environment and transmit when the desired physical event occurs, and sensors can be put into sleep or switched off at other times in order to conserve energy.

1.2 Organization

The paper is organized as follows. Section 2 begins with the definition of localized processing and then highlights the characteristics of localized processing with a generic example and its importance in sensor networks. In section 3, the general system model and overview of the sensor networks are provided. Section 4 is devoted to computational and algorithmic aspects of sensor networks focusing on the collaboration activities of sensors towards achieving a global goal for some phenomenon of the physical world. This section contains descriptions of different algorithms concerning computation and cooperation of sensors for solving certain computational problems in sensor networks.

2 Localized processing

2.1 Definition and example

In this section, we give the definition of localized processing which is in fact one of the main techniques for optimizing the valuable resources of sensors.

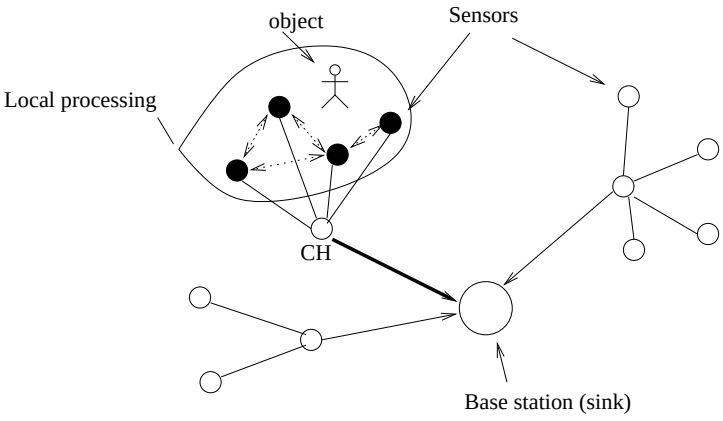


Fig. 1: An example of how nodes locally communicate and send data to the sink.

By *localized* processing we mean that a node is allowed to make a small number of communications with its neighbors which are 1-hop or 2-hops away from it to do some computations and/or make any

decision without the global view of the network. Based on this local computation on the data gathered by the neighbors and the node itself and through the coordination among them, it is hoped that the network eventually achieve some global goal. For example, measuring the temperature of a certain area for a certain period of time and sending back the data to the sink can be one of the typical goals that can be achieved through localized processing. In fact, a localized algorithm is a special case of *distributed* algorithms where each node executes its own algorithm and repeatedly exchanges information with its neighbors to learn more about the network [44]. An example of a local processing is shown in Figure 1. In this example, a subset of sensors (the filled circles) detect an object in their territory and inform their cluster head (CH) of the presence and location of that object. Upon receiving the signals from these sensors, the CH is expected to report the information back to the sink. Only the sensors surrounding the object along with the cluster head are involved in detecting the object, transmitting and receiving the signals to produce an estimate of the location of the object, and finally sending the aggregated signal directly or via other CHs to the sink. This is a form of localized processing where power saving is ensured by not involving distant sensors in the network which are not concerned in detecting and routing the information of the event.

Localized processing, due to its efficiency in terms of power consumption, has become an important research issue and there are many sensing applications or tasks that require the sensors to process data cooperatively and combine information from multiple sources [36]. Here, the sources are the sensors that detect some event of interest from the environment. On the contrary, in traditional centralized sensing and signal processing systems, environmental phenomena sensed by sensors are relayed as signals to the remote base stations for further processing [32]. If every sensor node has some data to send to another node in the network, then there could be shortage of expensive bandwidth because of the redundancy of data routed by every node. This is a well-known result in terms of wireless capacity as established in [33]. From the energy point of view, transmitting raw data (without processing or compressing) to distant nodes is wasteful of limited resources. In a networking context, localized and decentralized systems instead of centralized systems appear as promising solutions to improve and overcome the situations arising from centralized systems.

2.2 Why locality?

The deployment of hundreds of sensors over some geographic area increases the possibility of detecting the same event or covering the same areas by closely placed sensors. That is, the data these densely deployed sensors generate is expected to be highly correlated. Intuitively, if each individual node starts sending its own sensed information towards the sink, then the overall lifetime of the network will be reduced, since nodes will run out their batteries by sending unnecessary data, whereas they could be switched off to conserve energy. Moreover, the network will experience bottlenecks with so many communications, and the sink will receive redundant data. A scenario of this kind is depicted in Figure 2.

Data from multiple sensors with overlapping sensing regions is almost always correlated. By utilizing the knowledge of correlation we can eliminate the overlapped data by processing it locally. Then we can send only the optimum amount of data to the base station through multi-hop communication which definitely saves some of the expensive and limited bandwidth of sensors. So, the purpose of localized and collaborative processing is to combine the effect of localization among the sensors in order to achieve a global result from the network [32]. The importance of such local processing in

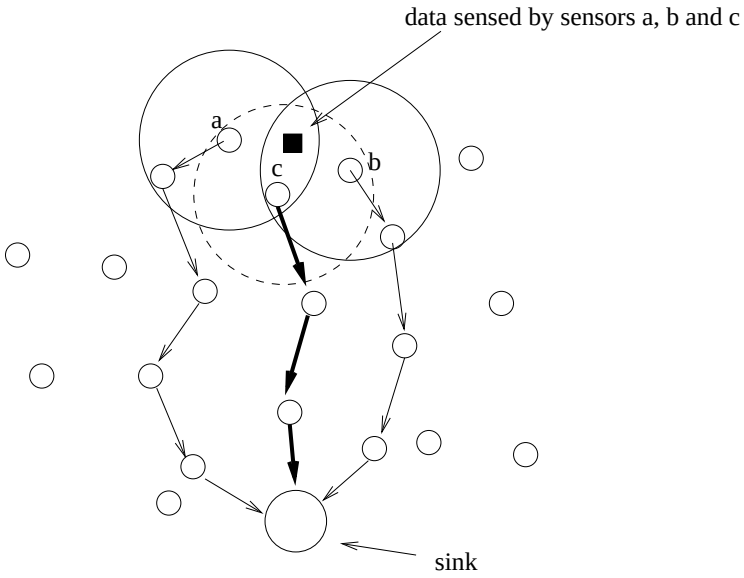


Fig. 2: Same data (denoted by the filled square) sensed by sensors a, b and c are routed through three paths to the sink. But only sensor c could have sent the data via its path to the sink keeping all the other sensors (a and b and all the nodes in their paths) idle.

sensor networks has been proven fundamental and advantageous and current research activities are focused on local algorithm design techniques.

2.3 System model

As the deployment of sensors and their network (topology) in the 2-D plane (flat region assumed) can be compared with *graphs* where the nodes and the edges of the graph correspond to the sensors and their links, we can use the algorithmic techniques of *graph theory* to address the problems of sensor networks.

Sensor nodes with omnidirectional antennas transmit signals to communicate with other nodes for the purpose of forming adjacencies, which in turn, form the connected network. As mentioned earlier, for a node u, other nodes become its neighbors if they are within its transmission radius, r. This is the general connectivity assumption among nodes. This assumption gives rise the most commonly used graph model for sensors, called the *Unit Disk Graph* (UDG) [45]. Given a set of sensor nodes $V \subset R^2$ distributed in a plane, the connectivity graph $G = (V, E)$ is called a unit disk graph if for any two nodes $u, v \in V$ they are neighbors if their distance is at most 1 ($\{u, v\} \in E \leftrightarrow |uv| \leq 1$). Here, $|uv|$ is the Euclidean distance between u and v. The transmission radius is normalized to 1. This model is the simplest model possible since radio signals are susceptible to small obstacles (which are ignored) and cannot form adjacencies if signals are interrupted by obstacles. However, most references follow the assumption of the UDG model. The underlying assumption is that G is always connected.

There are a number of other graph models such as the *Quasi Unit Disk Model* (QUDG), *Bounded Independence Graph* (BIG), the *Unit Ball Graph* (UBG), UDG with *Distance Interference* (UDI), etc. A good discussion of these models and their influence on the design and performance of algorithms for

sensor networks can be found in [44]. An example of the UDG and QUDG models is shown in Figure 3.

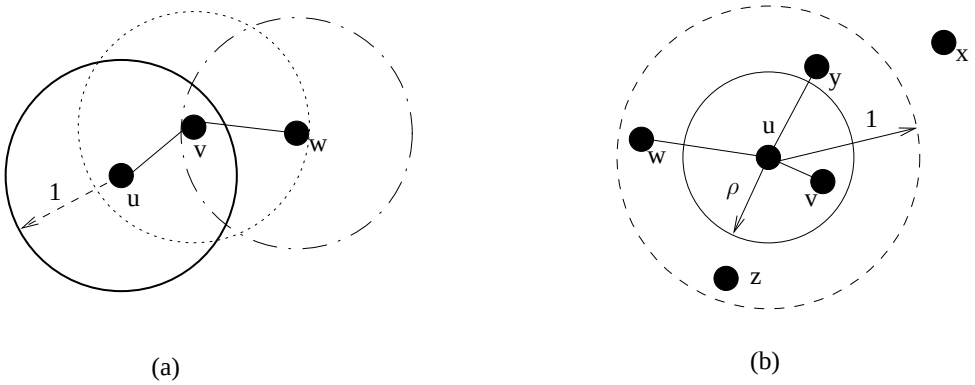


Fig. 3: (a) Unit Disk Graph: node u is adjacent to node v (distance ≤ 1), but not to node w (distance > 1). (b) Quasi Unit Disk Graph: Node u is always adjacent to v ($d(u,v) \leq \rho$)) but never to x ($dist(u,x) > 1$). All other nodes may or may not be in $u's$ transmission range. Here node u is adjacent to w and y, but not to z.

3 Computational problems in sensor networks

Sensor networks present a number of conceptual and optimization problems because of their application in various fields. The set of problems ranges from sensor placement to extraction of meaningful data from the environment. In this section, we describe some of the common computational problems considered in sensor networks which deal with local computation among a subset of closely placed sensors. When deployed in an environment, the sensors are expected to sense data, process the data and send it to the sink according to the way they (the sensors) are programmed.

Most often the requests for certain data or information to be obtained from the environment are sent to the sensors externally from a user in the form of queries. According to the demand and nature of the queries, the requested data are detected, collected, measured and processed by the sensors from the environment and sent back in the desired form to the user. In the following section, we give a general idea about the structure of queries which are sent to individual sensors to perform sensing about the event of interest.

3.1 Queries in Sensor Networks

The way sensors sense data can be classified into different categories. There are applications that require nodes to sense data periodically at all times, e.g., the nodes can be programmed to sense the temperature of a certain area at every 30 seconds or so for a certain number of days or weeks or more or they can be used to compute functions like *max, min, median, count* related to certain natural phenomena. These types of queries last longer in the network and are referred as *continuous* or *long-running* queries. Another type of queries termed as *short-term snapshot* queries, occurs when the

user wishes to obtain data at a given point of time, for example, "Retrieve the current light intensity from a chemical plant". Generally, queries are generated in the form of a standard SQL (Structured Query Language) with some unique clauses suitable for sensor networks. An example [47] of this type is given below:

```
SELECT AVG(R.Concentration)
FROM ChemicalSensor R
WHERE R.Loc in region
HAVING AVG(R.Concentration)>T
DURATION (now, now+3600)
EVERY 10
```

where the template of the query has standard database semantics except two new clauses, the *DURATION* and the *EVERY* which are unique in sensor networks as they define the lifetime of a query and the rate of query answers, respectively. Other types of problems require the sensors to be activated or triggered and report to the sink when some "interesting" event (the presence of a foreign object, a fire, an earthquake, etc.) occurs in the environment. Only that subset of sensors will be active if their detected or measured value is greater than some predetermined threshold regarding the occurrence of the incident. For example, in an object tracking application a subset of the sensors can be triggered by the presence of an object when it is found in their territory. All other times they can be put into sleep mode.

In the section that follows we provide, as the most common application of sensor networks, the description of some algorithms of how sensors coordinate and cooperate among themselves to track a moving object in some 2-dimensional field.

3.2 Object tracking

Localizing and tracking of moving objects (also called targets) is an important capability for a sensor network and can be considered as a generic problem. This is because the tracking scenario raises a number of fundamental information processing issues in distributed information discovery, representation, communication, sensing, storage and querying [32]. Object tracking has many practical applications especially in military battlefields, highway traffic monitoring, and facility security [48], where it is essential for sensors to acquire local, partial and relatively crude information estimation of targets. Most often the task of tracking objects is the result of a collaborative effort of a group of nearby sensors who cooperatively estimate the track of a target in their territory. The purpose is to obtain a good estimate of the trajectory of target objects $x^{(t)}$ from the individual measurements $z^{(t)}$ of a group of sensors at time t.

The two issues that need to be taken care of for a successful sensor network operation are efficient methods of exchanging information between sensors and collaborative signal processing between them to gather useful information. Moreover, due to the energy limitation, at any instant of time, we can employ as few sensors as possible to maintain and continue successful tracking.

3.2.1 Distributed Bayesian estimation

A query to track any moving object in the sensor field is passed to the individual sensors from the sink. Tracking is aimed to obtain a close and good estimate of the target location from the measurement

history of $\overline{z^{(t)}} = \{z^{(0)}, z^{(1)}, z^{(2)}, \cdots, z^{(t)}\}$ [48] where each $z^{(t)}$ is measured by a sensor representing the state of the moving object such as its position and/or velocity at time t. The authors in [48] use Bayesian estimation for this problem, where an estimate $\hat{x}(\overline{z^{(t)}})$ is wished to be as close as the true value of the target state $x^{(t)}$ (the actual position and velocity of the object). For example, the estimate $\hat{x}(\overline{z^{(t)}})$ can be obtained from the measurement history which may represent the average of the measurements. The aim is to minimize the average cost $E[d(\hat{x}(\overline{z^{(t)}}), x^{(t)})]$, where $d(.,.)$ is a loss function to measure the estimator performance. For instance, $d(\hat{x}, x) = |\hat{x} - x|^2$ measures the square of norm-2 distance between the estimate and its true value. For this loss function, the estimate is

$$\hat{x}^{(t)}_{MMSE} = E[x^{(t)}|\overline{z^{(t)}}] = \int x^{(t)} p(x^{(t)}|\overline{z^{(t)}}) dx^{(t)}$$

The estimator $\hat{x}^{(t)}_{MMSE}$ is known as the minimum mean-squared error (MMSE) estimator. The *a posteriori* distribution $p(x^{(t)}|\overline{z^{(t)}})$ is known as the *belief*, that is, the posteriori distribution of $x^{(t)}$ given the measurement history $\overline{z^{(t)}}$). The issue is to compute the belief efficiently. The belief is passed from a previous leader (a leader is a representative sensor node of a cluster of nodes) to a new leader to incorporate the new measurement $z^{(t+1)}$ of the new leader. This measurement $z^{(t+1)}$ or the new position of the target $x^{(t+1)}$may be independent of the past history $\overline{z^{(t)}}$. Under these assumptions, the new leader computes the new belief $p(x^{(t+1)}|\overline{z^{(t+1)}})$ using the Bayes rule:

$$\begin{aligned} p(x^{(t+1)}|\overline{z^{(t+1)}}) &\propto p(z^{(t+1)}|x^{(t+1)}) p(x^{(t+1)}|\overline{z^{(t)}}) \\ &= p(x^{(t+1)}|z^{(t+1)}) \int p(x^{(t+1)}|x^{(t)}) p(x^{(t)}|\overline{z^{(t)}}) dx^{(t)} \end{aligned}$$

In the above equation, $p(x^{(t)}|\overline{z^{(t)}})$ is the belief inherited from the previous step; $p(x^{(t+1)}|z^{(t+1)})$ is the likelihood of observation given target location and $p(x^{(t+1)}|x^{(t)})$ is related to vehicle dynamics. Once the updated belief $p(x^{(t+1)}|z^{(t+1)})$ is computed it is transferred to the new leader which is a neighbor of the previous leader. To which sensor to handoff the belief is crucial because otherwise the less informative sensor will become the leader without giving much information about the target. One method is to compute the mutual information between the current leader and its neighbors and select the one which has the maximum mutual information indicating there is little overlapping of information between them. The process of shipping the tracked data continues until it is sent back to the querying node or the user. As implied above, the algorithm is distributed and sensors cooperatively track the moving objects without having a global knowledge about the network. Each node is aware only of its neighbors, communication is only between neighbors and computation of measurement and update of belief is local.

3.2.2 Location-centric algorithms

In [49], the authors study the object tracking problem through a straightforward and simple way using an energy detection algorithm. The algorithm is based on geographic-centric clusters of sensors where a subset of sensors form a cluster, based on their geographic locations, called a *cell*. Each cluster has a cluster head (CH) which coordinates signal processing and communication of sensors of that cell to other cells. As some target enters in a cell the sensors of that cell detect the presence of that target and report it to the CH at N successive time instants. Depending on the output signals from the

sensors, the CH then estimates the possible location of the target at each time instant. One of the algorithms used to estimate the location of the target, is by using the positions (the coordinates of the sensors) of the sensors which supply their outputs to the CH. As the CH estimates the location of the target at each time instant, the CH can compute the possible trajectory of the target by using the estimates of the locations of the target computed in N time intervals. This prediction determines which of the cells the target is likely to enter, and the current CH awakes the CH of the new cell. Upon entering in a new cell, the sensors of the previous cell go to sleep and the sensors of the new cell wake up and repeat the procedure. In order to localize the target the authors take advantage of the equation

$$y_i(t) = s(t)/|r(t) - r_i|^{\alpha},$$

where $y_i(t)$ is the energy reading at sensor i, $s(t)$ is the target signal energy, and $r(t)$ is the location of the target, all at time t, r_i is the location of sensor i and α is the power loss exponent. At each time instant, the algorithm computes $y_i(t)/y_j(t)$ between all pairs of sensors in the cluster they belong to, where each ratio defines a circle in which $r(t)$ may reside. Among n sensor readings only $n-1$ of the total $n(n-1)/2$ ratios are independent where all the circles intersect at a single point. See Figure 4 for an example.

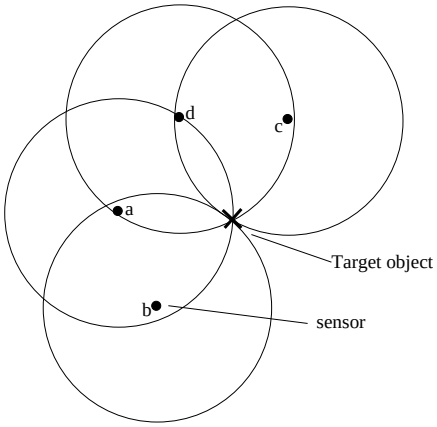


Fig. 4: (a) Four sensors a, b, c, and d identify a target object 'X' in their territory.

For instance, if we have four sensors detecting an object, we find,
$y_1/y_2 = (r(t) - r_2)^2/(r(t) - r_1)^2$; $y_1/y_3 = (r(t) - r_3)^2/(r(t) - r_1)^2$;
$y_1/y_4 = (r(t) - r_4)^2/(r(t) - r_1)^2$; $y_2/y_3 = (r(t) - r_3)^2/(r(t) - r_2)^2$;
$y_2/y_4 = (r(t) - r_4)^2/(r(t) - r_2)^2$; $y_3/y_4 = (r(t) - r_4)^2/(r(t) - r_3)^2$.

Obviously, the first three ratios (y_1/y_2, y_1/y_3, and y_1/y_4) are independent and other three ratios can be derived from them. The unknown position $r(t)$ is estimated by solving a nonlinear least squares problem of the form

$$J(X, Y) = \sum_{i=1}^{n} |(X - O_{i,x})^2 + (Y - O_{i,y})^2 - \rho_i^2|^2$$

where (X, Y) , $(O_{i,x}, O_{i,y})$, and ρ are the target coordinates, center coordinates and the radius of the circle associated with the i^{th} ratio, respectively. The assumptions made in the paper may be violated

in practice making the problem of target tracking more difficult. For example, the power loss exponent may vary between sensors and the signal strength may also be a function of direction depending on the locations of sensors.

3.2.3 Binary sensors

In [31], the authors offer a new idea of using binary sensors for tracking moving objects where the sensors are supposed to provide only one-bit of information (hence called binary sensors) regarding an object's presence or absence in their vicinity. Depending on the output of the active sensors (those producing 1 as output) their method determines the positions in the path of the target in the near past and finds the line which best fits the path points. This line is used to estimate the target's current position. Basically this method estimates a target's position at a given time as the weighted average of the sensor positions. A higher weight is given to a sensor close to the target. Since the active sensors keep track of how long they have been watching the target in their vicinity, their method can make a rough estimate of the velocity of the target with the detection duration of the sensors. The idea depends on the assumption of a straight line trajectory of the moving object and constant velocity of the target. The assumption of a straight line trajectory and constant velocity is rather weak since in a real world situation the path followed by an object may not be a straight line and the velocity is not always constant.

3.2.4 Other algorithms

In [30], a location-centric approach is advocated to perform collaborative sensing and target tracking. The idea is to develop programming abstractions that provide addressing and communication between localized geographic regions (cells) within the network rather than individual sensors. Cells are made dynamically along the trajectory of the moving object. No hints are given about the size of a cell and its effect on the overall object tracking scenario but it is assumed that the cell size is a function of observed target velocity. Although the authors demand that the algorithms are applicable to track multiple objects, the primary assumption is that the objects must remain far apart from each other in order to avoid interference of signals. The authors in [29] present a self-organized distributed target tracking algorithm with predictions based on Pheromones, Baysian and Kalman Filter techniques. They consider the object tracking problem a as two-tier process: (i) clusters of nodes locally estimate parameters used in object tracking (i.e., time, class, position, and heading), and (ii) local parameter association forms inter-cluster tracks. However, the algorithms assume that the trajectory of the moving object is linear.

3.2.5 Open problems

Extending the problem of single-object tracking to multiple-object tracking and classifying is more challenging. Although a few approaches have been proposed towards solving the multiple-object tracking problem, those are computationally expensive, require extensive communication in the network when new observations are made, and require a significant amount of storage for the representation of distribution over all data associations. When multiple targets are far apart from each other, the network can partition itself into a number of sub-networks, which are solved simultaneously, each as a single object tracking problem. Difficulty arises when targets come closer to each other which requires sophisticated techniques to solve the source separation and data association problems. The key

problem that needs to be addressed is the interference between signals from different targets as they come in close proximity. It would be interesting to come up with solutions which can correctly classify multiple objects at the cost of relatively few transmissions among the sensors, effectively avoiding the combinatorial explosion caused by data associations among multiple targets.

Another important issue in single-object or multiple-object tracking, is that many techniques depend on prior statistical information about the signals. However, the statistical information is heavily influenced, and can be easily corrupted, by the harsh nature of the environment. Moreover, the way in which the sensors are deployed, one can expect a significant fraction of the devices to be either non-operational or malfunctioning. That is why it is desirable to design algorithms that are robust in the face of a large number of device failures and adaptive to variations in environmental conditions (for example, the presence of a strong wind can affect the acoustic measurements of sensors and introduce errors) that influence the signals measured or sensed by sensors. Moreover, the effect of Doppler shifts can play significant roles in acoustic and seismic measurements due to the relatively slow speed of wave propagations in such modalities [49].

3.3 Coverage and Connectivity

The coverage and connectivity problem in sensor networks is one of the significant issues that has received much attention in recent years. The main focus of this problem is to determine how well a given area is observed by a network of sensors, that is, the concern is whether all the points of an area of 'interest' are monitored by the sensors. This is much like the Art Gallery Problem which deals with finding the minimum number of guards (observers) to guard the entire gallery so that every point in the gallery is covered by at least one guard. In general, the coverage problem not only attempts to determine whether the given area is fully covered but also provides an answer to the level of coverage of points, i.e., whether the points are covered by one or more sensors. The basic idea is to devise algorithms that would determine whether the whole area of interest is covered by a small or possibly minimum connected subset of sensors. The issue of energy saving can also be considered by these algorithms by turning off the nodes which are not included in the minimum subset. They can later be turned on when the active sensors monitoring the area run out of their energy. In this section, we describe different algorithms for solving coverage and connectivity problems and mention the relative advantages and disadvantages of these algorithms. We also discuss interesting and open problems regarding coverage and connectivity issues in sensor networks raised from this research.

3.3.1 Perimeter-coverage and Best-coverage problem

In [50], the authors formulate the coverage problem as a decision problem, in which the goal is to determine whether every point in the given service area, A, is covered by at least k-sensors (k is a constant). The algorithm answers 'Yes' if any point in A is within the range of each of the k sensors else the answer is 'No'. The authors solve the problem of coverage by showing that the area of interest A is k-covered if and only if each sensor in the network is k-perimeter-covered. The perimeter-coverage of a sensor is defined as follows: a point p on the perimeter of sensor s_i is perimeter-covered by sensor s_j if this point is within the range of s_j. The point is called k-perimeter-covered if it is covered by at least k sensors other than s_i. Polynomial algorithms in terms of the number of sensors have been offered as solutions, where the sensors can locally compute whether all the points on their perimeter are k-covered. These algorithms have a typical running time of $O(nd \log d)$, where d is the maximum

number of sensors whose sensing ranges intersect the sensing range of a sensor and n is the total number of sensors.

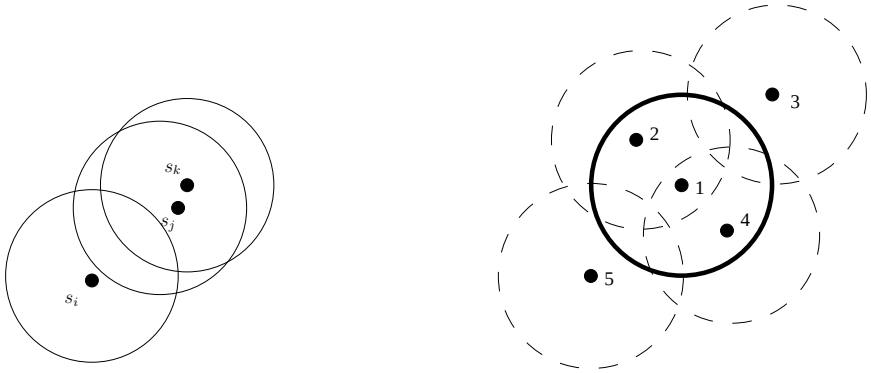
A variant of the coverage problem called the *best-coverage* problem was introduced in [28], and is defined as finding a path P in a network connecting two points s and t, which maximizes the smallest detection probability (observability) of all on points on the path. The authors also define the *worst-coverage* problem which calls for finding a path that maximizes the distance of the path to all sensor nodes. This implies how well the path P is protected by the sensors. Although they developed polynomial solutions to the best-coverage problem using Delaunay triangulations, these algorithms are centralized and they do not provide any justification why the search space should be confined to the Delaunay triangulation for finding such paths. The construction of Delaunay triangulation is centralized.

The best-coverage problem is later studied in [27] and an efficient distributed algorithm for this problem is given where the authors take into consideration that the sensing ability diminishes as the distance from a sensor increases. They actually show how to find a best-coverage path with the least energy consumption. The idea is based on the relative neighborhood graph (RNG) consisting of all edges $uv \in E$ such that there is no point $w \in V$ with edges uw and wv in E satisfying $|uw| < |uv|$ and $|vw| < |uv|$. As a preprocessing step, RNG is constructed first in a distributed manner in $O(n \log n)$ time. Following this, the distributed shortest path algorithm (the Bellman-Ford algorithm) is applied between two nodes s (source) and t (destination); it also runs in $O(n \log n)$ time to find the best-coverage path. It is shown that the best-coverage path only uses the edges of the relative neighborhood graph. Although the authors of [28] did not prove why their algorithm correctly works for the best-coverage problem, the authors in [27] did actually prove the correctness of the algorithm.

3.3.2 Area-dominating set

Assuming that the sensing radius of each sensor is the same as its transmission radius and that the sensors know their geographical positions, a localized solution has been proposed in [26] to solve the *area-dominating set* problem. The problem is to find the smallest subset of sensors that covers the monitored area. The decision is based on time rounds where the network lifetime is divided into a set of equal time rounds. The idea is that each sensor selects a time interval and at the end of the interval if the node sees that its neighbors (that have not sent any 'withdrawal message') together cover its monitoring area, it sends a 'withdrawal message' to all its neighbors and goes into the sleep mode. Otherwise, the node remains active and does not transmit any message. This process is repeated periodically to allow changes in the monitoring status of the sensors. Although the algorithm is local, it has some problems. For example, some active neighbors may die without any notice and may not activate, believing that the sensor is alive and monitoring. Moreover, the covering sensors may not be connected and thus reporting to a monitoring station may not succeed.

Several improvements have been offered in [25] to the algorithm in [26]. The basic assumption in [25] is that the transmission radius is more than twice the sensing radius, which is different from the assumption made in [26]. The authors in [25] introduce the notion of effective neighborhood of sensors in order to tackle the problems mentioned above. For a sensor s_i and its neighbors s_j and s_k, if the intersection of the sensing circles of s_i and s_k is completely covered by the intersection of the sensing circles of s_i and s_j, then s_j is called an effective neighbor and s_k is not an effective neighbor of s_i as shown in Figure 5(a). A sensor's sensing area is covered completely by its effective neighbors if and only if the segment of each effective neighbor is perimeter covered by s_i's other effective neighbors as



(a) Effective Neighbor

(b) Coverage redundancy by effective neighbors

Fig. 5: (a) An example where s_j is an effective neighbor of s_i and (b) shows how the sensing area of a sensor is covered by its effective neighbors.

shown in Figure 5(b). In this figure, the sensing area of node 1 is fully covered by its effective neighbors since each segment of sensor 1's effective neighbor lying in 1's sensing area is perimeter covered by 1's other effective neighbors. If the sensing area of a sensor is covered, it then decides to sleep. Because of the distributed nature of the algorithm, sensors apply a random backoff before making such a decision (whether to be active or go into sleep mode) since two neighbors can evaluate the algorithm at the same time. The proposed algorithm can reduce overall system energy consumption, and therefore increase network system lifetime, by turning off some redundant nodes. However, the algorithm has some restricting assumptions such as all the sensors are time-synchronized, transmission range of a sensor is at least twice the sensing range and each sensor needs to know its geographical position.

In order to reduce some computation while retaining the main idea of dividing the network lifetime into rounds, the perimeter coverage criterion was replaced by a computationally more efficient criterion relative to covering intersection points of two circles inside a given circle in [24]. The authors in [24] also relax the restriction of the ratio of sensing and communication radii SR and CR, respectively (in [25], $CR \geq 2SR$), and consider general ratios. The idea of this paper for full coverage of an area is as follows: if there are at least two covering circles and any intersection point of two covering circles inside the sensing area is covered by a third covering circle, then the sensing area is fully covered. This can be interpreted in other words. A disk d is fully covered by other disks if and only if every intersection point of two disks d_1 and d_2 inside d is covered by another disk d_3. An example of such a scenario is shown in Figure 5 (b) where any intersection point between any two circles inside circle 1 is covered by a third circle and thus circle 1 is fully covered. The advantage of this idea is that it is able to quickly decide whether the sensing area of a sensor is covered and works for any ratio of sensing and communicating radii. Furthermore, it is shown that the communication overhead is very low and no neighbor discovery phase is needed which demonstrate robustness when message collisions are considered in high density sensor deployment.

3.3.3 Open problems

The sensor coverage problem can be extended further in several ways to yield many interesting unsolved problems. For example, in a sensor network we may wish to find the areas which are insufficiently covered meaning that the points in this area are not k-covered. This is interesting because it is important to identify some less covered regions in order to provide or insert more sensors to these areas to obtain more coverage. By contrast with the insufficient coverage issue, a sensor network may be overly covered by too many sensors in certain areas. If that is the case, then we may turn off the redundant sensors to save energy and, later when the other sensors run out their energy, those sensors can be turned on. Moreover, in certain applications, we can consider that some areas are more important than others which are called *hot spots*. Identifying hot spots in some networks is crucial; for example, in battlefields some areas need more attention than other areas. In such a situation, we can deploy more sensors and have greater vigilance.

Most of the algorithms assume that the sensors know their geographical locations and are time-synchronized. It would be an interesting result if the coverage problem along with its extensions (mentioned above) can be solved without considering these idealistic assumptions. This is because equipping hundreds of sensors with GPS is quite expensive while assuming sensors are time-synchronized is rather a strong assumption. Also, it is not enough to consider coverage alone when deploying a sensor network: connectivity must also be considered. This is because some moderate loss in coverage can be tolerated by some applications, but loss in connectivity can be severe. A monitoring node A is only allowed to switch off if its sensing area is covered by a connected set of its neighbors. Suppose the neighbors of A are not all connected but they all together cover the sensing area of A. In that case, if one of the neighbors B, which is not connected with the other neighbors of A but monitors some part of the sensing area of A, fails or malfunctions then there is no way for other neighbors of A to know this. As a result, part of the area which was being monitored by B would remain uncovered. See Figure 6 where sensor a's sensing area is covered by a set of disconnected and connected neighbors.

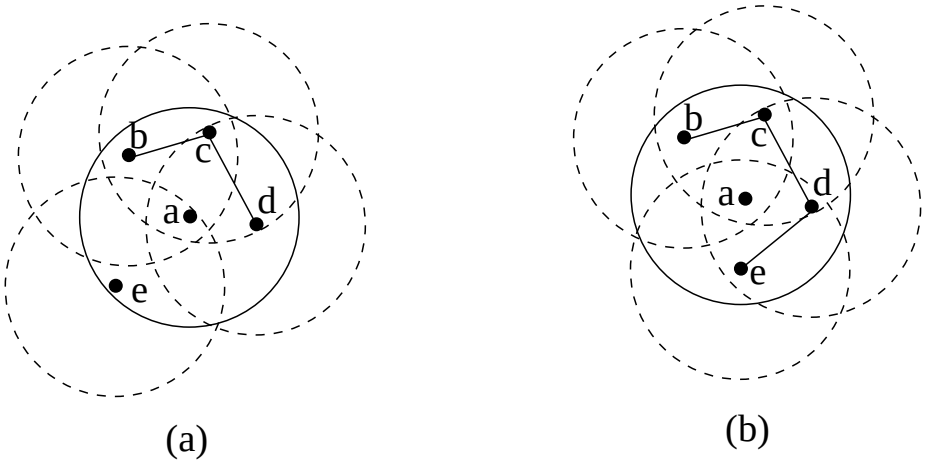


Fig. 6: (a) Sensor a's sensing area is covered by a set of disconnected neighbors and (b) covered by a set of connected neighbors.

3.4 Data Gathering

The data gathering problem in sensor networks focuses on the systematic gathering and transmission of sensed data to the base station for further processing. As raw data are gathered by sensors from the environment and sent to the base station, it is wished that the data in the base station satisfy (in terms of completeness and timeliness of data) users who initiated the queries in the network. At the same time, the overall energy expenditure of the whole network should be minimized during the collection, processing, and transmission of sensed data from the sensors to the base station. The goals are contradictory because providing an acceptable level of accurate and adequate data requires more communication among sensors with the expense of more energy. Nonetheless, in order to achieve some trade-off between the two contradictory goals, a fair amount of research has been conducted in recent times. In this section, we highlight some of the research works that have addressed the data aggregation issue with the consideration of energy saving aspect of sensor networks.

3.4.1 Maximum lifetime data gathering

In [23], the authors consider the data gathering problem in the following way: given the location of sensors and the available energy of each sensor, it is desired to find an efficient way in which the data should be collected from all the sensors and transmitted to the base station, such that the system lifetime is maximized. This is called the maximum lifetime data gathering problem. A distributed algorithm is proposed for this problem under two different scenarios: (a) sensors are allowed to perform in-network aggregation of data packets, and (b) no sensors are allowed to aggregate its data packets with that of another sensor. For the first scenario, the authors design an integer linear programming approach in which they maximize the system lifetime, T, which is the number of rounds until the first node is drained of its energy. In each round, a sensor i is allowed to send one packet p of data to its neighbor j, who in turn aggregates its own data packet with p and sends the aggregated data packet to its neighbor k. A *schedule*, S is defined which specifies, for each round, how the data packets from all the sensors are collected and transmitted to the base station. A schedule can be thought of as a collection of T directed trees, each rooted at the base station and spanning all the sensors, i.e., a schedule has one tree for each round. The schedule S induces a flow network, $G = (V, E)$, which is a directed graph having as nodes all the sensors and the base station and having edges (i, j) with capacity $f_{i,j}$ where $f_{i,j}$ represents the number of data packets that can be sent from sensor i to sensor j. Then, they consider their maximum lifetime data gathering problem as to find a flow network G with maximum flow T, that allows each sensor to push flow T to the base station and formulate it as an integer linear programming under the constraints of the flow conservation principle, capacity constraints, and energy availability of sensors.

This scenario is useful when sensors have correlated data since this allows sensors to compress data from other sensors and send the result to the base station. The second scenario applies when the data are not correlated, and consequently the sensors are not allowed to aggregate their data packets. For example, when video images are sent by sensors from distant regions of battlefields, we do not need to aggregate data to intermediate sensors. This type of situation can be handled in almost the same way as the first problem of data gathering with aggregation, without any sensors requiring to aggregate data.

The experimental results presented in that paper [23] demonstrate that the proposed algorithm significantly outperforms previous data gathering methods like PEGASIS [21] and LEACH [22] in

terms of system lifetime. The limitation of the method is that it is computationally expensive for large sensor networks since in each round it has to compute the maximum flow network. Moreover, the algorithm assumes that a sensor is permitted to aggregate its own data with that of any sensor of the network, but in reality this is not always true. It would be more reasonable to assume that a sensor can aggregate its data with the data from certain sensors only. For example, a node is only allowed to aggregate its data with the data of its children. Another problem is the delay experienced in the network. The depth of a data gathering schedule gives an estimate of the average delay that is incurred in sending data packets from any sensor to the base. It would be an interesting problem to consider the tradeoff between the delay constraints for individual sensors and the lifetime achieved by the system.

3.4.2 Distributed aggregation

A distributed randomized algorithm is proposed in [20] for data aggregation (computing aggregates like average, sum, minimum, or maximum) in sensor networks. The algorithm is based on random grouping, where in each round, each node independently becomes a group leader with a certain probability, and then invites its neighbor to join the group. Then all the members in a group update their values with the locally derived aggregate (average, minimum, etc.) of the group. Through this randomized process, the authors show that all the values progressively converge to the correct aggregate value. The main idea is to reach a global aggregate value from the values collected locally through repeated execution of their Distributed Random Grouping (DRG) algorithm. The results of this work give an upper bound on the expected number of rounds needed for all nodes running DRG to converge to the global value. Each execution of the algorithm is considered a round. Other advantages of this method are that, it works well under all topologies (where certain other techniques like flooding, can not converge to the correct global average in grid topology), and is robust against changes from link/node failures. There are a number of limitations of their technique. The algorithm is synchronized, that is, rounds are divided into synchronous time slots. It is not deterministic and finally, the expected number of rounds is input sensitive since it depends on a number of factors, namely the underlying graph topology, the grouping probability, the accuracy requirement, and the initial value distribution of sensor nodes.

3.4.3 Correlation and probabilistic aggregation

The authors of [19] suggest an approach based on correlation of data among sensors for the data aggregation problem where the network provides periodic estimates of the environment. They consider the problem of finding the 'maximum' value of some physical phenomenon, like temperature or humidity. The algorithm is distributed and it mainly depends on the spatial-temporal correlation of sensor's values which is leveraged to make the estimates of the aggregated descriptions of the environment. The smaller the desired accuracy of the estimates, the more power or energy can be saved by exchanging fewer messages. The main idea is that each node makes an estimate based on its local measurement of the physical process, and then decides whether to send this aggregated estimate to its neighbor. At each period, a node generates a value which is the mean of the estimated 'max' of the physical process and its variance and combines them with the global estimate. When a node receives an estimate from its neighbor, it fuses its own estimate with that of its neighbor and decides to transmit the combined new estimate only if it brings significant changes to its neighbor's estimate.

A method is developed in [19] for finding a threshold that determines when a node should transmit its new value to its neighbor. The higher the value of threshold, the lower the accuracy the smaller the transmission and the larger energy saving. An advantage of this algorithm is that there is no need for time synchronization between nodes and there is no requirement on the number of messages a node should receive. However, this method is computationally expensive, as each sensor needs to compute a covariance matrix in every time period and sends the matrix to its neighbor when the global aggregated description is a vector quantity.

In order to put more emphasis on energy saving while maintaining the quality of aggregated data, a technique based on dynamically changing a subset of nodes is proposed in [18]. This technique is useful for applications requiring a continuous supply of data from some physical process. In that paper, the authors present a probabilistic and localized approach. Their idea is to collect data from a dynamically changing subset of nodes called *sampler* nodes and predict data for other *non-sampler* nodes. First, they form clusters on the basis of nodes' close readings or measurements about some physical process. Then they further group nodes within each cluster into subclusters with the sensors having highly correlated readings. In the next step, a sensor is selected from each subcluster. The subset of these sensors thus forms the sampler subset. A probabilistic model is then chosen for each subcluster so that the prediction of the readings of the non-sampler sensors can be made by the sampler node without allowing the non-sampler sensors to transmit their readings. The parameters of the model are determined from the recent sample readings of the non-sampler nodes. The algorithm effectively reduces the number of nodes used to transmit data and thus decreases the energy consumption rate of the network and increases the overall network lifetime.

The main limitation of this method is that for large networks, running the cluster formation algorithm frequently takes a large number of messages to be exchanged and thus causes interference and retransmissions. Moreover, forming subclusters within clusters also suffers from the same effect of interference. The authors do not mention how often the algorithm should run and when to stop or what happens when a sampler node in a subcluster runs out of energy.

3.4.4 Open problems

A significant amount of energy is exploited in sensing, processing, sharing, and transmission of data among the nodes in the network. In data aggregation, one should put emphasis on the quality of gathered data as well as the efficiency of the energy saving strategies. For large networks, sending a query and getting the reply back depends on many factors, for example, the size of clusters, depth of the aggregation tree, node or link failure, corrupted data, and signal interference, etc. For certain applications [23], after sending a query if the reply of the query takes a lot of time to reach to the base (due to the hop-distance of the tree), the reply might have no use at that time it reached to the base or if certain nodes fail to propagate the query (or reply) then the user may never get the answer in time. It would be a nice direction of research to find a way of providing as much reliable and timely data as possible in the face of such difficulties. For many data aggregation applications [20], algorithms are designed to run in synchronized time slots which ensure less signal interference and fewer message collision and quick delivery of queries. However, in large networks, this assumption seriously affects the implementation of these algorithms as maintaining synchronization is quite difficult and practically infeasible. Thus, it will be interesting to seek robust and distributed algorithms with reasonable assumptions for data gathering problems that guarantee the availability and completeness of requested data with enhanced system lifetime.

3.5 Geometric problems in sensor networks

Sensors, when deployed in the environment, organize themselves in some connected structure in order to operate efficiently and effectively. The basic form of organization implies that the nodes build a connected structure following, most probably, the construction of a Unit Disk Graph (UDG). Such a network consisting of a set of sensors distributed in a two-dimensional plane is always assumed to be connected with high probability [17]. By utilizing the properties of the UDG of the network, we can suitably apply many well-known geometric algorithms such as finding shortest paths between two nodes, constructing low-weighted structure of the network (e.g, RNG, modified RNG, GG, etc.), determining connected dominating set, finding energy optimal routes between two nodes, preserving connectivity with reduction of degrees in nodes, and so on. In this section, we aim to describe some algorithms for solving such problems as indicated above and address some open problems related to the geometry of organization of sensor networks.

3.5.1 Geometric structures

Before we go into details in discussing some algorithms, we review some well-known geometric structures and provide a number of definitions and terminologies as they are used in the literature for evaluating the performance of these algorithms in finding energy efficient structures in sensor networks for broadcasting and multicasting operations. The definitions include spanners, power stretch factor, low-weighted structure, relative neighborhood graph, Gabriel graph, independent set, and connected dominating set.

Spanners: Let $G = (V, E)$ be an n vertex connected weighted graph. The distance in G between two vertices $u, v \in V$ is the total weight length of the shortest path between u and v is denoted by $d_G(u, v)$. A subgraph $H = (V, E' \subset E)$ is a t-spanner of G if, for every $u, v \in V$, $d_H(u, v) \leq t.d_G(u, v)$. The value of t is called the *stretch factor* [1].

Power stretch factor: Consider a path from u to v, $p = v_0, v_1, \cdots, v_k$ where $u = v_0$ and $v = v_k$ in G. The total power P consumed by this path, is defined as $P = \sum_{i=1}^{k} |v_{i-1}v_i|^\beta$. Let $P_G(u, v)$ be the least energy consumed by all paths connecting nodes u and v in G. A subgraph H of G has a power stretch factor $\rho_H(G)$ if $\rho_H(G) = \max_{u,v \in V} P_H(u, v)/P_G(u, v)$ [1].

Low-weighted Structure: Given a graph G over a set of points, let $\omega(G)$ be the total length of the links in G and $\omega_\beta(G)$ be the total power needed to support all links in G, i.e., $\omega_\beta(G) = \sum_{uv \in G} |uv|^\beta$ where $|uv|^\beta$ is the amount of power needed to support link uv. A structure G is called low-weighted if $\omega_1(G)$ is within a constant factor of $\omega_1(MST)$ [1]. For simplicity we may omit the subscript 1 when $\beta = 1$.

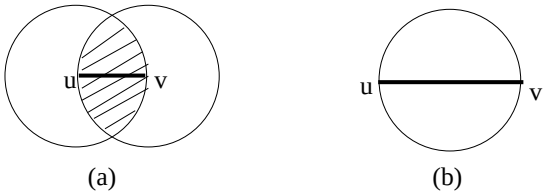


Fig. 7: (a) The *lune* (dashed area) using edge uv is empty of other nodes for RNG. (b) The edge uv is in GG since the circle with diameter uv does not contain any other nodes.

Relative Neighborhood Graph and Gabriel Graph:

The Relative Neighborhood Graph of G, denoted by $RNG(G)$, is a subgraph consisting of all edges uv for any points u and v such that there is no point $w \in V$ with edges uw and wv satisfying $|uw| < |uv|$ and $|vw| < |uv|$.

Let $disk(u, v)$ be the disk with diameter uv. The Gabriel graph $GG(G)$ contains an edge uv from G if and only if $disk(u, v)$ contains no other vertex $w \in V$. It is well known that both RNG and GG have $O(n)$ edges and contain the Euclidean minimum spanning trees as subgraphs. See Figure 7.

Connected Dominating Set and Independent Set:

A subset S of V is a *dominating set* if each node $u \in V$ is in S or is adjacent to some node $v \in S$. A subset C of V is a *connected dominating set* (CDS) if C is a dominating set and induces a connected subgraph. A dominating set with minimum cardinality is called a *minimum dominating set*, denoted by MDS. Similarly, a connected dominating set with minimum cardinality is called a *minimum connected dominating set* (MCDS). The nodes in a dominating set are called *dominators* and those not in the dominating set are called *dominatees*. A subset of V is an *independent set* if, for any pair of vertices, no edge is between them. An independent set is a *maximal independent set* (MIS) if no more vertices can be added to the set to generate a larger independent set. An MIS is a dominating set.

3.5.2 Topology control and broadcasting

In a typical wired network, the topology is fixed throughout the network lifetime unless there are physical damages in nodes and/or wires of the network. However, in a sensor network the scenario is quite different where the topology of a network depends on the transmission power of individual sensors and the topology in such a network can be changed upon the adjustment of transmission power of sensors. The underlying goal of topology control in wireless sensor networks is to maintain network connectivity, optimize network lifetime, and make it possible to design power-efficient routing [12, 13, 8, 10, 11, 14]. We expect the spanner, the derived topology from the original graph, to contain only a linear number of links because restricting the size of the network is very important in reducing the overall power consumption of the network and the amount of transmission of routing information.

Broadcasting is a communication paradigm that allows to send data packets from a source to multiple receivers or in some cases to all the nodes in the network. A significant amount of the expensive bandwidth and the limited power of nodes will be wasted unless we control unnecessary transmissions or flooding of packets. Thus, the primary objective of broadcasting is to allow only a small subset of nodes (more specifically, minimum connected dominating set) to relay packets and ensure that the packets be received by all the nodes in an efficient way. This guarantees a limited number of transmissions of messages and consumes minimum total transmission power in the network. Towards this goal, a number of research works have [3, 5, 6, 7, 14, 15, 13] pursued in recent years to devise distributed methods and techniques to approximate minimum connected dominating sets (MCDS) of nodes for broadcasting through a virtual backbone in sensor networks. In the following subsections we discuss some methods for topology control and broadcasting in sensor networks.

3.5.3 Low-weighted structure

A locally constructed structure known as the low-weighted modified relative neighborhood graph has been introduced in [1]. The idea is based on a simple modification of the original RNG. However,

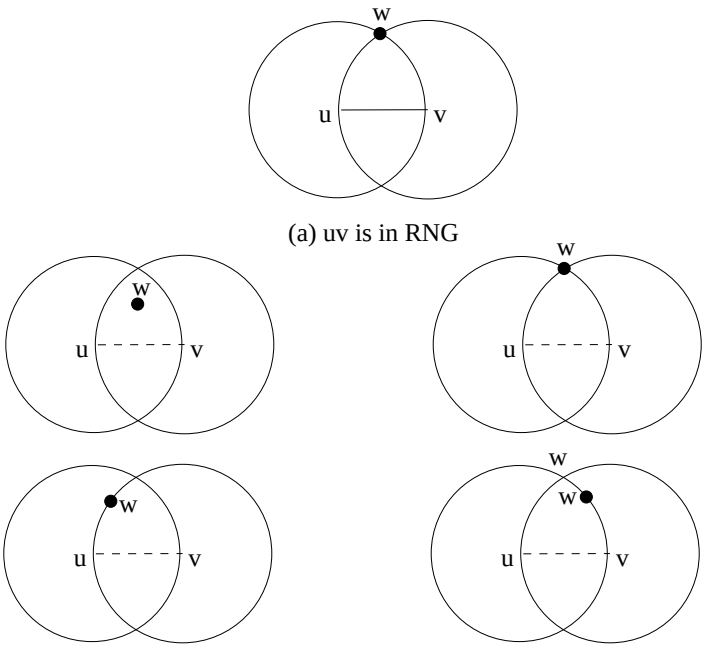
RNG may have unbounded node degree, e.g., considering $n-1$ points equally distributed on the circle centered at the nth point v, the degree of v is $n-1$. For the sake of lowering the weight of a structure, the structure should contain as few edges as possible without breaking the connectivity. The construction of modified RNG which is sparser than RNG is as follows: given RNG, first they construct a graph called RNG'. The RNG' consists of all edges such that (1) the interior of $lune(u,v)$ (the intersection of two circles centered on u and v with radius uv) contains no points $w \in V$ and, (2) there is no point $w \in V$ with $ID(w) < ID(v)$ on the boundary of $lune(u,v)$ and $|wv| < |uv|$, and (3) there is no point $w \in V$ with $ID(w) < ID(u)$ on the boundary of $lune(u,v)$ and $|wu| < |uv|$ and, (4) there is no point $w \in V$ on the boundary of $lune(u,v)$ with $ID(w) < ID(v)$, $ID(w) < ID(u)$ and $|wv| = |uv|$. Here $ID(u)$ means the ID of a sensor, which is generally an integer.

See Figure 8 for RNG' where an edge uv is not included in the modified relative neighborhood graph. After the RNG' is constructed, each node u locally broadcasts its incident edges in RNG' to its one-hop neighbors and also listens to them. Assume u receives a message from its neighbor x about the existence of edge xy. For each edge uv in RNG', if uv is the longest among uv, xy, ux, and vy, then node u removes edge uv. The final structure, denoted by H, is formed by all remaining edges of RNG'.

The authors show that H is a bounded degree, planar, and connected structure and the total edge length is within a constant factor of that of the minimum spanning tree, i.e., $\omega(H) = O(\omega(MST))$. Their method is local, uses only $O(n)$ messages to build such structure and every node uses only its two-hop information. Although the total edge length of this structure is within a constant factor of that of the MST, the energy consumption using this structure is within $O(n^{\beta-1})$ of the optimum, i.e., $\omega_\beta(H) = O(n^{\beta-1}).\ \omega_\beta(MST)$. The interesting thing of their structure H is that it is better than the previously best-known sparse structure RNG in energy consumption where RNG uses $O(n)$ times the total energy used by H. That is, $\omega_\beta(RNG) = O(n^\beta).\ \omega_\beta(MST)$. The main application of this structure lies in efficient broadcasting in wireless networks. However, for broadcasting, as mentioned above the method cannot achieve a constant factor energy consumption of the optimum structure of MST. To find such a structure that has a constant factor energy consumption is still an open problem .

3.5.4 Structure based on $LMST_k$

The computational cost for the construction of H is high since for each link $uv \in RNG'$, node u has to test whether there is an edge $xy \in RNG'$ and $x \in N_1(u)$ such that uv is the longest among uv, xy, ux, and vy. In order to improve the computational complexity, a method based on *local minimum spanning tree* (LMST) is proposed in [10] that finds a sparse energy-efficient structure for message broadcasting. The construction of this structure is as follows: each node u first collects its one-hop neighbors $N_1(u)$. Node u then computes the minimum spanning tree $MST(N_1(u))$ of the induced unit disk graph on its one-hop neighbors $N_1(u)$. Node u keeps a directed edge uv if and only if uv is an edge in $MST(N_1(u))$. The union of all directed edges of all the nodes is known as the *local minimum spanning tree*, denoted by $LMST_1$. Actually, according to the construction this is a graph not a tree. If only symmetric edges are kept, then the graph is called $LMST_1^-$, i.e., it has an edge if and only if both directed edge uv and vu exist. Ignoring the directions of the edges in $LMST_1$, they call the graph $LMST_1^+$, i.e., it has an edge uv if either edge uv or vu exists. It is shown in [10] that the graph $LMST_1^-$ and $LMST_1^+$ are planar, connected, and have bounded degree (6). Later the definition of one-hop LMST is generalized to k-hop neighbors and $LMST_k$ is shown to be the



(a) uv is in RNG

(b)The four cases where uv is not included in modified RNG, RNG'.

Fig. 8: (a) RNG and (b) modified RNG called RNG'.

union of all edges of all minimum spanning trees $MST(N_k(u))$. An important result of their paper is that $MST \subseteq LMST_k \subseteq RNG'$. For $k = 2$, although $LMST_k$ has some properties such as being planar, bounded degree, and low-weighted, the communication cost of building $LMST_2$ is very large. This is because a large number of message transmissions are required for collecting two-hop neighbors' information $N_2(u)$ for each node.

In order to increase energy savings in broadcasting, it is expected that only a small subset of nodes will be engaged in broadcasting messages to get the messages to all the nodes in the network. And the nodes outside the subset will only receive but not transmit messages. This leads to the problem of finding connected dominating set (CDS) where the nodes in the set are responsible for transmitting all the messages in the network. In the following section, we review some distributed methods that approximate minimum connected dominating sets (MCDS) for unit disk graph.

3.5.5 Distributed dominating set

For general graphs some randomized distributed algorithms based on a synchronous model of computation are presented in [7] for the minimum dominating set problem that run in polylogarithmic time and are independent of the diameter of the network. These algorithms return a dominating set of size within in a logarithmic factor of the optimum with high probability. The algorithm runs in $O(\log n \log(\Delta + 1))$ (Δ is the largest degree over all nodes in the graph) rounds with high probability. Each round involves a constant number of message exchanges among any two neighbors. The computed dominating set is within $O(\log \Delta)$-approximation ratio in expectation and within $O(\log n)$-time with high probability. An open problem that arises from their research is to see whether a

distributed deterministic algorithm exists for $O(\log n)$-time and $O(\log \Delta)$-approximation ratio for the MDS problem. It would be interesting to determine the best approximation-time tradeoff achievable by a deterministic distributed algorithm.

The algorithm proposed in [6] is based on some pruning technique for finding a connected dominating set in UDG. First, they find a connecting dominating set and then prune out certain redundant nodes from the CDS. The initial CDS, $\mathcal{C}$ contains all nodes that have at least two non-adjacent neighbors. A node u is said to be *locally redundant* if it has either a neighbor in $\mathcal{C}$ with larger ID which dominates all other neighbors of u or two adjacent neighbors with larger IDs which together dominate all other neighbors of u. Their algorithm then keeps removing all locally redundant nodes from $\mathcal{C}$. It is shown that the algorithm works well in practice when the nodes are distributed uniformly. The main limitation of their algorithm is its high approximation ratio, which could be as high as $n/2$, where n is the number of nodes.

In [5], a distributed construction of a connected dominating set (CDS) is presented following a particular clustering scheme where the CDS consists of two types of nodes: the clusterheads (dominators) and the border-nodes. Initially all nodes are considered white. The status of a node, after the clustering method finishes, could be a dominator with color black or dominatee with color gray. Each node has a unique parameter called *rank* which is an ordered pair of its degree and its location. The ranks of all nodes give rise to a total ordering of all nodes.

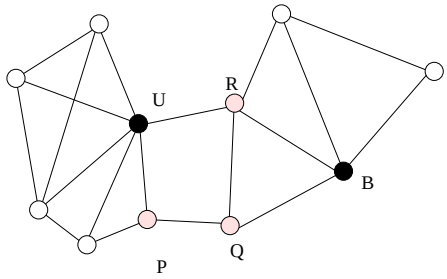


Fig. 9: Node U and B are clusterheads and P, Q, and R are border-nodes.

The algorithm works in the following way: At each step, all white nodes which have the lowest rank among all white neighbors are colored black, and the white neighbors are colored gray. The ranks of the white nodes are updated if necessary. A white node claims itself to be a dominator (clusterhead) if it has the smallest ID among all of its white neighbors, if there is any, and broadcasts **IamDominator** to its 1-hop neighbor. A white node receiving **IamDominator** message marks itself as dominatee and broadcasts **IamDominatee** to its 1-hop neighbors. The set of dominators generated by the above method is actually a maximal independent set (MIS). After the clusterhead nodes are selected, the border-nodes are selected to connect them. A node is border-node if it is not a clusterhead (dominator) and there are at least two clusterheads within its two-hop neighborhood. The border-nodes are used to connect the dominators and they together (the border-nodes and the dominators) form the connected dominating set. Here the assumption is that each node knows the IDs of all its 1-hop neighbors, which can be achieved if each node broadcasts its ID to its neighbors initially. An illustration of the clusterheads and the border-nodes is shown in Figure 9.

In the worst case, the algorithm exhibits an approximation ratio of $n/2$. If only IDs are used as ranks, which remain unchanged throughout the process, both the time complexity and the message

complexity of this algorithm are $\Theta(n)$. But the inclusion of node degrees in the ranks involves a significant number of messages to be exchanged (message complexity in the worst case is $O(n^2)$) because the degrees of nodes can change frequently in large and dynamic networks causing a lot of huge rank updates.

In order to overcome these limitations a new distributed algorithm for finding CDS is proposed in [3]. The main features of this method include a good approximation factor of at most 8, $O(n)$ time complexity, and $O(n \log n)$ message complexity. One of the important results established in the paper is that $\Omega(n \log n)$ is shown to be the lower bound on the message complexity of any distributed algorithm for nontrivial CDS. They achieve the message optimality following the leader-election algorithm in [16]. The method for CDS consists of two phases which construct a maximal independent set (MIS) first and then a dominating tree. They define the rank of a node which is the combination of its ID and its level (number of hops to the root of the spanning tree). Initially all nodes are colored white. The labelling process (coloring the nodes either black or gray) begins from the root and finishes at the leaves. The node with the lowest rank marks itself as black and broadcasts a *DOMINATOR* message to its neighbor. The marking process then continues with the following rules:

1. If the first message that a node receives is a *DOMINATOR* message, it marks itself gray and declares itself as a dominatee by sending a *DOMINATEE* message.
2. Whenever a node receives a *DOMINATEE* message from all its lower rank neighbors, it marks itself black and sends a *DOMINATOR* message.

The marking process finishes when it reaches the leaf nodes. The set of black nodes form an MIS.

In the second phase, a CDS is constructed which represents spanning all the black nodes by using *INVITE* and *JOIN* messages. Actually, the CDS is constructed in such a way that the nodes in the CDS form a tree called dominating tree. Initially, the root joins the CDS and broadcasts an *INVITE* message. The *INVITE* message is relayed to all two-hop neighbors out of the current CDS. When a black node receives the *INVITE* message for the first time, it joins the dominating tree together with the gray node, which relayed the message. It then sends a *JOIN* message. The process terminates until all the black nodes join the CDS.

The following example shows how their algorithm works. See Figure 10. In this figure, node 0 is the root of the spanning tree that is chosen by using the leader-election algorithm [16]. The solid lines represent the edges of the rooted spanning tree, and the dashed lines represent other edges in the UDG. Node 0 is marked black first and it broadcasts a *DOMINATOR* message (solid arrows in Figure 10(a)). After receiving this message, nodes 2, 4, and 12 are marked gray and they broadcast the *DOMINATEE* message. For simplicity only the *DOMINATEE* messages from node 4 are shown as the dashed arrows in Figure 10(a). Then node 5 is selected to be a *DOMINATOR* as it has received *DOMINATEE* messages from all its lower rank neighbors (node 4 only). Following the algorithm, Figure 10(b) shows the colors of the nodes when the labelling process finishes. The final process builds the dominating tree from the root. The *INVITE* message (solid arrows in Figure 10(b)) is sent from node 0 and it is relayed to its two-hop black neighbors 3, 5, and 7. These black nodes join the dominating tree, as well as their relaying gray nodes 2 and 4. The thick links in Figure 10(b) illustrate the edges in the final dominating tree. All nodes in the tree form a connected dominating set. The improved approach in [2] merges the MIS construction with the dominating tree building processes. As shown in Figure 10(c), node 2 is colored black when it first receives a DOMINATOR message from its child 3, and node 4 is marked black for the same reason. Finally, all the black nodes

form a connected dominating set.

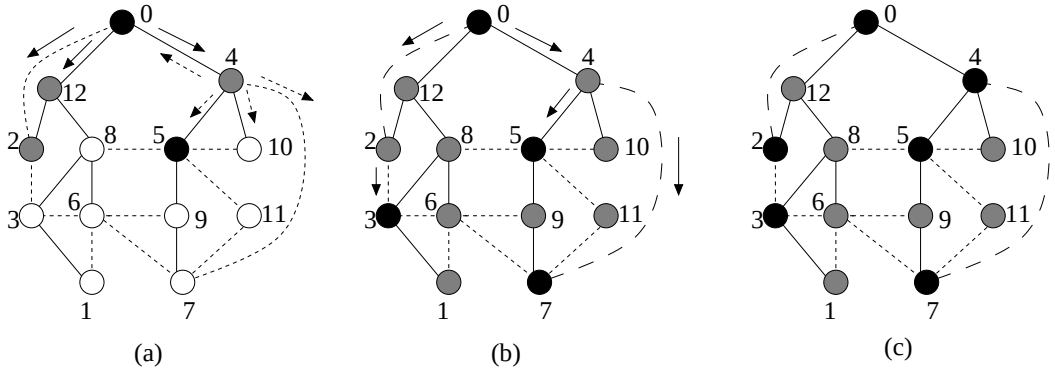


Fig. 10: An example to show the CDS construction.

The algorithm suffers from the expensive construction of spanning trees. Building spanning trees and maintaining CDS in large and volatile network with common node and link failures incur significant number of retransmissions and communication overhead.

An improved algorithm is reported in [4] that does not rely on the construction of spanning trees. The algorithm forms a CDS in UDG with size at most $192.|OPT|+48$ and message complexity is $O(n)$. Initially all the nodes are candidates for the dominators. Whenever the ID of a node becomes the smallest among all of its one-hop neighbors, it will change its status to dominator. Consequently, its candidate neighbors become dominatees. After all nodes change status, each dominator node identifies a path of at most three hops to another dominator with larger ID. The candidate nodes on this path become connectors. All dominators and connectors compose a connected dominating set. In Figure 11, the CDS formation is shown following the algorithm in [4]. Nodes 1, 2, 3, and 4 declare themselves to be dominators at the beginning. Then node 5 declares itself as a connector on the paths from 1/2 to 3/4, so do nodes 6 and 7. Finally, all the connectors and dominators form a connected dominating set.

3.5.6 Open problems

One of the main issues is still to find a connected dominating set locally which will have small constant approximation ratio for power-efficient broadcasting in sensor networks. For this problem and many other, most of the theoretical results assume that the power emission is the major component of power consumption. In some devices, the power emission is at the same level of the power needed for being idle or to receive packets [9]. So, it is necessary to design power structures (e.g., virtual backbones of CDS in sensor networks which are power-efficient for broadcasting or topology control) with theoretically proven worst-case performance under this new energy model where the receiving power is not negligible. Another important aspect of designing energy efficient protocols for broadcast and multicast is the physical constraints of the wireless communications. Most often, it is assumed that the signals sent by a node will be received by all its neighbors at once. However, in practice, this is not the case always, and considering it will make designing energy efficient broadcast and multicast protocols much harder. The difficulty is that the sender needs to coordinate with its receivers so

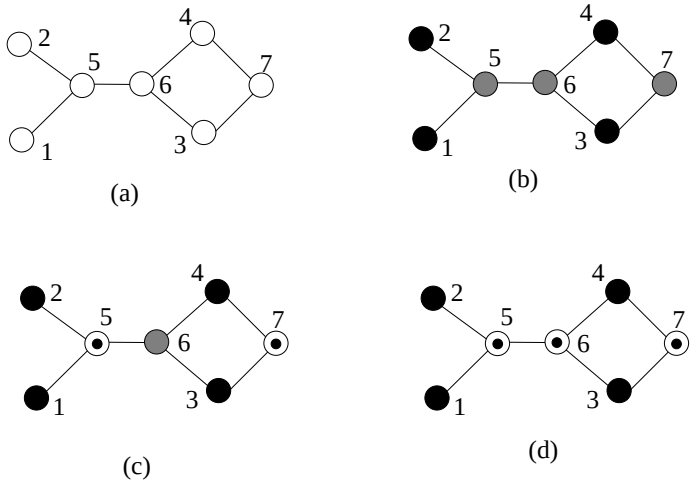


Fig. 11: An example of message-optimal CDS construction [4].

that all are ready to receive. As indicated in [9], a natural question is then how to set the threshold t such that, if the number of receivers that are ready is more than t, only then node u sends the packets for them; otherwise node u will not transmit. The setting of the threshold will affect the total actual energy consumption of the broadcast protocols and may affect the system performance and the stability of the networks.

There is also a contradictory issue that needs to be exposed. In large networks, a reduced size of a CDS implies low connectivity in the network which means that failure of any node in the CDS may disconnect the CDS and ruin the backbone formed by the CDS. In such a situation, proper broadcasting with this dissconnected dominating set will not be possible. On the other hand, this small sized CDS (if no node of this set fails) guarantees less interference, fewer transmissions and low power consumption in the network for broadcasting or multicasting. However, if the size of a CDS is relatively large then we have a strong connectivity among the nodes in the CDS in the network and failure of some nodes may not interrupt the broadcasting or multicasting operation. But there is a potential chance that the network will suffer from interference which will eventually cause more retransmissions and more power consumption in the network. So, finding a tradeoff between these two extremes is an interesting research problem.

4 Conclusion

As discussed in the paper, sensors with their unique characteristics can be used to track objects, monitor certain area, estimate physical phenomena such as temperature, pressure, humidity, and so on, broadcast messages containing interesting 'event's and so forth. Thus it is essential for us to utilize the capability of sensors as much as we can for the collection of necessary and timely data by increasing their lifetime and saving bandwidth which are directly dependent on their low and limited battery power. In this paper, we tried to highlight the computational aspects of sensors which mainly deal with the individual sensors' ability to compute information and communicate with each other locally with a view to achieve some global goal.

Although we have mentioned several open problems raised from our study in each of the categories, namely, object tracking, coverage problems, data aggregation, and topology control and broadcasting, we highlight them again and summarize them together to conclude the paper.

1. Although a few attempts have been made for tracking multiple-object, these methods are computationally expensive, dependent on huge communication in the network, and require a substantial amount of memory to represent the distribution of data associations of multiple targets. Even in the single-object tracking scenario, noise and other environmental phenomena play an important role on signals transmitted by targets. The effect of these external factors can be taken into account to formulate robust methods to make tracking more efficient, correct, and timely. So the need for devising new distributed algorithms considering the problems of current methods and environmental factors is highly demanding.
2. Monitoring area by sensors is investigated which raises some interesting problems. Most of the algorithms assume that the sensors know their geographical locations and are time-synchronized. It would be an interesting result if the coverage problem along with its extensions (identifying hot spots, finding less covered area, etc.) can be solved without considering these idealistic assumptions. This is because equipping hundreds of sensors with GPS is quite expensive while assuming sensors are time-synchronized is rather a strong assumption.
3. In data aggregation, one should put emphasis on the quality of gathered data along with the energy saving strategy to obtain an acceptable tradeoff between them. For large networks, sending a query and getting the reply back depends on many factors, for example, the size of clusters, depth of data aggregation trees, node failures, corrupted data, and signal interference, etc. Among all these factors, delay constraint or latency shows a strong challenge to the data gathering problem. It is a good research direction to search for faster heuristics, considering depth constraints (depth in data gathering trees) for individual sensors to attain desired tradeoffs between the delay experienced for queries to be answered and the overall lifetime increased by the system.
4. Connected dominating sets have proven to provide an important backbone in underlying sensor networks and are essential for many applications, for example, energy efficiency routing, media access coordination, broadcasting, multicasting and so on. There is a tradeoff between fast operation, bandwidth efficiency and connected dominating set (CDS) size. If the size of a CDS is larger, then we have relatively slow operation for broadcasting (or multicasting) and the bandwidth will not be efficiently utilized as chances of message collisions increase. On the other hand, small CDS size can ensure faster service and efficient bandwidth utilization. However, finding a CDS whose size is within a small constant factor (currently 8) of the minimum connected dominating set is still a challenging problem. Another critical issue about CDS is its stability in the face of frequent topology change. In order to maintain the stability of a CDS in terms of its member nodes, the distributed algorithms should run at the same time. Otherwise by the time a CDS is constructed the network topology may have changed such that the result of the algorithm is not a connected dominating set at all. Thus, the consideration of node failure, insertion of new nodes, link's instability and node mobility are important issues for building robust CDS algorithms for sensor networks.

References

[1] Xiang-Yang Li. Approximate MST for UDG Locally. In *COCOON*, 2003.

[2] K. M. Alzoubi, P.-J. Wan, O. Frieder. Distributed Heuristics for Connected Dominating Set in Wireless Ad Hoc Networks. *IEEE ComSoc/KICS Journal on Communication Networks* 4(1): pp. 22-29, March 2002.

[3] K. M. Alzoubi, P.-J. Wan, O. Frieder. New Distributed Algorithm for Connected Dominating Set in Wireless Ad Hoc Networks. *IEEE HICSS35*, Hawaii, Janauary 2002.

[4] K. M. Alzoubi, P.-J. Wan, and O. Frieder. Message-Optimal Connected-Dominating-Set Construction for Routing in Mobile Ad Hoc Networks. *The Third ACM International Symposium on Mobile Ad Hoc Networking and Computing*, June 2002.

[5] I. Stojmenovic, M. Seddigh, J. Zunic. Dominating sets and neighbor elimination based broadcasting algorithms in wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 13(1), January 2002, 14-25.

[6] J. Wu and H. Li. Domination and Its Applications in Ad Hoc Wireless Networks with Unidirectional Links. *Proc. of International Conference on Parallel Processing (ICPP)*, 189-200, August 2000.

[7] Lujun Jia, Rajmohan Rajaraman, and Torsten Suel. An efficient distributed algorithm for constructing small dominating sets. *Distributed Computing*, 15(4): 193-205 (2002).

[8] X.Y. Li and I. Stojmenovic. Broadcasting and topology control in wireless ad hoc networks, in: *Handbook of Algorithms for Mobile and Wireless Networking and Computing, (A. Boukerche and I. Chlamtac, eds.), CRC Press* to appear.

[9] X.Y. Li and I. Stojmenovic. Broadcasting and topology control in wireless ad hoc networks. *Handbook of Algorithms for Mobile and Wireless Networking and Computing, CRC Press*, to appear.

[10] Wen-Zhan Song, Yu Wang, Xiang-Yang Li, and Ophir Frieder. Localized algorithms for energy efficient topology in wireless ad Hoc networks. *ACM Mobile Network and Applications (MONET)*, 10(6), 911-923, 2005.

[11] Xiang-Yang Li. Localized construction of low weighted structure and its applications in wireless ad Hoc networks. *ACM Wireless Network (WINET)*, 11(6), November 2005

[12] B. Das and G. Bharghaban. Routing in adhoc networks using minimum connected dominating set. In *Proc. IEEE International Conference for Communication (ICC 97)*, 1997.

[13] I. Stojmenovic, M. Seddigh, and J. Zunic. Dominatig sets and neighbor elimination based routing algorithms in wireless sensor networks. In *IEEE Trans. Parallel Distributed Systems* 13(1), pp. 14-25, 2002.

[14] P. Wan, K. Alzoubi, and O. Frieder. Distributed consturction of connected dominating set in wireless adhoc networks. In *INFOCOM*, 2002.

[15] Y. Wang and X.Y Li. Geometric spanners for wireless adhoc networks. In *ICDCS*, 2002.

[16] I. Cidon and O. Mokryn. Propagation and Leader Election in a Multihop Broadcast Environment. In *International Symposium on Distributed Computing*, 104-118, 1998.

[17] R. Ramanathan and R. Rosales-Hain, Topology control of multihop wireless networks using transmit power adjustment. In *Proc. INFOCOM*, 2000.

[18] B. Gedik and L. Liu. Energy-aware data colllection in sensor networks: A localized selective sampling approach. *Technical reports*, Georgia Institue of Technology, git-cercs-05-18, 2005.

[19] A. Boulis, S. Ganeriwal, and M.B. Srinavista. Aggregation in sensor networks: an energy-accuracy trade-off. *SPNA*, May 11, 2003.

[20] J. Chen, G. Pandurangan, and D. Xu. Robust computation of aggregates in wireless sensor setworks: distributed randomized algorithms and analysis. *IEEE Transaction on Parallel and Distributed Systems*, pp. 987-1000, Vol. 17, No. 9, September 2006.

[21] S. Lindsey, C.S Raghavendra, and K. Sivalingam. Data gathering in sensor networks using Energy*Delay Metric. *IPDPS Workshops on Issues in Wireless Networks and Mobile Computing*, 2001.

[22] W. Heinzelman, A.P. Chandrakasan, and H. Balakrisnan. Energy efficient communication protocols for wireless microsensors networks. *proc. of Hawaiian Intl. Conference on Systems Science*, 2000.

[23] Konstantinos Kalpakis, Koustuv Dasgupta, and Parag Namjoshi. Maximum lifetime data gathering and aggregation in wireless sensor networks. *In the Proceedings of the 2002 IEEE International Conference on Networking (ICN'02)*, Atlanta, Georgia, August 26-29, 2002. pp. 685-696.

[24] Antoine Gallais, Jean Carle, David Simplot-Ryl, and Ivan Stojmenovic. Localized sensor area coverage with low communication overhead. *Fourth Annual IEEE International Conference on Pervasive Computer and Communications PerCom*, Pisa, Italy, March 13-17, 2006.

[25] Jie Jiang and Wenhua Dou. A coverage-preserving density control algorithm for wireless sensor networks. *Ad-Hoc, Mobile, and Wireless Networks*, Vol 3158, pages 42-55.

[26] D. Tian and N.D. Georganas. A covergae-preserving node scheduling scheme for large wireless sensor networks. *First ACM intl. Workshop on Wireless Sensor Networks and Applications*, 32-41, September 2002.

[27] X. Li, P. Yuan and O. Frieder. Coverage in wireless adhoc sensor networks. *IEEE Trnasactions on Computer*, Vol 52, No. 6 June 2003.

[28] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. Srivastava. Coverage problems in wireless ad-hoc sensor networks. *INFOCOM*: 1380-1387, 2001.

[29] R. Brooks, P. Ramanathan, and A. Sayeed. Distributed target classification and tracking in sensor networks. *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1163-1171, 2003.

[30] R. Brook, C. Griffin, and D. Friedlander. Self-organized distributed sensor network entity tracking. *Intl. J. of High performace Computing* vol. 16, no. 3, pp 207-220, 2002.

[31] W. Kim, K. Mechitov, J. Choi, and S. Ham. On Tracking Objects with Binary Proximity Sensors. *Fourth International Conference on Information Processing in Sensor Networks (IPSN '05)*, April 2005.

[32] F. Zhao and L. Guibas. Wireless Sensor Networks: An Information Processing Approach. *Morgan Kaufmann (ISBN 1-55860-914-8)*, 2004.

[33] P. Gupta and P.R. Kumar. The capacity of wireless networks. *IEEE Trans. on Information Theory*, 46(2): 388-404, 2000.

[34] A. Scaglione and S.D. Servetto. On the interdependence of routing and data compression in multi-hop sensor networks. *In Proc. 8th Annual International Conference on Mobile Computing and Networks (MobiCom 2002),* pages 140-147, Atlanta, GA, ACM Press, September 2002.

[35] DARPA Sensor Information Technology program. http://www.darpa.mil/ito/research /sensit/index.html.

[36] F. Zhao, J. Liu, J.J. Liu, L. Guibas, and J. Reich. Collaborative signal and information processing: an information directed approach. *Proc. IEEE*, 91(8): 1199-1209, 2003.

[37] P. L. George and H. Borouchaki. Delaunay Triangulations and Meshing. *HERMES, Paris*, 1998.

[38] http://www.cs.virginia.edu/ ipsn06.

[39] S.S. Pradhan, J. Kusuma, and K. Ramachandran. Distributed compression in a dense microsensor network. *IEEE Signal Processing Magazine*, 19(2)s: 51-60, March 2002.

[40] http://en.wikipedia.org/wiki/Wireless_sensor_networks.

[41] S. Olariu and I. Stojmenovic. Design guidelines for maximizing lifetime and avoiding energy holes in sensor networks with uniform distribution and uniform reporting. *IEEE INFOCOM*, Barcelona, Spain, April 24-25, 2006.

[42] T. Rappaport. Wireless Communications: Principles and Practices. *Prentice Hall*, Upper Saddle River, NJ, 1996.

[43] G.J. Pottie and W.J. Kaiser. Wireless Integrated Network Sensors. *Comm. of the ACM*, vol. 43 No. 5, pp. 51-58, May 2000.

[44] S. Schmid and R. Wattenhofer. Algorithmic Models for Sensor Networks. *Proc. of the 14th International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS)*, Island of Rhodes, Greece, April 2006.

[45] B.N. Clark, C.J. Colborn, and D.S Johnson. Unit Disk Graphs. *Discrete Mathematics*, 86: 165-177, 1990.

[46] I. Stojmenovic, A. Nayak, and J. Kuruvila. Design guidelines for Routing Protocils ain Ad Hoc and Sensor Networks with a realistic Physical Layer. *IEEE Communications Magazine*, March 2005.

[47] Y. Yao and J. E. Gehrke. Query Processing in Sensor Networks. *Proceedings of the First Biennial Conference on Innovative Data Systems Research* (CIDR 2003), Asilomar, California, January 2003.

[48] J. Liu, J. Reich, and F. Zhao. Collaborative in-network processing for target tracking. *EURASIPJ.Appl.Sig.Proc.*, (4): 378-391, March 2003.

[49] D. Li, K.D. Wong, Y.H. Hu, and A.M. Sayeed. Detection, classification, and tracking of targets. *IEEE Signal Processing Magazine*, March 2002.

[50] C. Huang and Y. Tseng. The Coverage Problem in a Wireless Sensor Network. *ACM SWSNA*, San Diego, USA, September 2003.