Undergraduate Project Report
2012/13

Scalable Processing of Queries on Sensor-Data in Internet of Things

Name:	[Chuyu Wang]
Programme:	[E-Commerce]
Class:	[2009215114]
QM Student No.	[090467712]
BUPT Student No.	[09213134]
Supervisor:	[Yue Chen & Martin Alexander Neumann]

13/05/2013

Table of Contents

Abstract

Wireless Sensor Networks (WSNs) have been a hot research area for a decade due to its unique capability to monitor physical phenomena. Traditionally, WSN purely performed data collection and aggregation. But in recent years, new applications of WSN such as tracking, disaster alarming, increasingly require in-network information storage and querying, which means each node of a WSN serves as data originator and data router at the same time. So the malfunctioning of a few nodes can cause significant topological changes and might require rerouting of packets and reorganization of the network. Besides, in most application scenarios every sensor node is equipped with limited power source and power recharge is almost impossible or impractical, so power conservation and power management take on additional importance. Given these facts, it is challenging to propose real-time query evaluation strategies that can deal with a multi-hop ad hoc sensor network while integrating energy efficient mechanisms. My aim of this research work is to find an approach to anytime-enabled query processing on distributed sensor-data in a sensitive, energy-efficient manner. I address the aforesaid challenges by designing an adaptive strategy that allows both event queries and non-event queries. Experiments and data analysis show that my strategy achieves better scalability and much faster response speed for accident query (one kind of event queries), as well as greatly balances power consumed by the whole network thereby prolonging the network life.

摘要(Chinese translation of the Abstract)

因其在监测物理现象方面的独特性能，无线传感器网络（WSN）是近十年来的研究热点。过去，无线传感器网络只是负责数据收集和数据融合，但近年来由于无线传感器网络在新领域，如目标追踪，灾难警报的应用越来越广泛，对网络内数据存储及处理提出了新的要求。也就是说，现在的无线传感器网络的每个节点都同时扮演着数据收集器和路由器的角色。所以某些节点的功能故障会引起网络拓扑的改变从而要求数据包重发或者网络重组。另一方面，由于能量限制，即每个节点的能量供给是有限的，能量节约和能量管理也是无线传感网络领域很重要的一个方面。综合考虑节能机制的实时查询响应策略的提出对大规模的自组织传感器网络至关重要。所以我们研究目标是针对分布式传感器数据提出一种实时的查询处理策略，这种策略是灵敏的，节能的。针对上面提出的问题，我设计了一个自适应策略，它同时支持事件查询和非事件查询。实验和数据分析显示，该策略比传统方法有更好的可扩展性，对突发事件查询（事件查询的一种）有更快的响应速度，并且很好地平衡了整个网络的能量消耗，从而延长了网络的使用寿命。

Chapter 1: Introduction

In Internet of Things (IoT) networks, many devices and sensors are connected with each other via Internet to provide internet-wide services. Specifically, with a number of sensors connected to a network in a wireless manner, it constitutes a Wireless Sensor Network (WSN). Sensors are devices that can sense or monitor physical phenomena, such as temperature or humidity of a region. A sensor can be as small as a penny but a sensor node is usually composed of several units: sensing unit, processing unit, communication unit and power unit [1]. Often hundreds of or even thousands of sensors are deployed (it allows random deployment such that the position of sensor nodes need not be engineered or predetermined) in a human-interested area, especially areas that people cannot reach easily or where a long term sensing task is required. By forming an ad-hoc network (this means that sensor network protocols or algorithms must support self-organizing), sensors can collect and transfer measured data to some certain locations therefore scientists or researchers can analyse them. This kind of technology is what we called Wireless Sensor Network (WSN). The typical network architecture of a WSN can be shown as **Figure 1**, from which we can see there are 3 main elements of a WSN: sensor nodes, objects and observers. Sensor nodes' tasks are gathering, processing and transferring measured data. Objects are what observers are interested in, such as humidity of a region. Observers can get their interested information by analysing the received data.

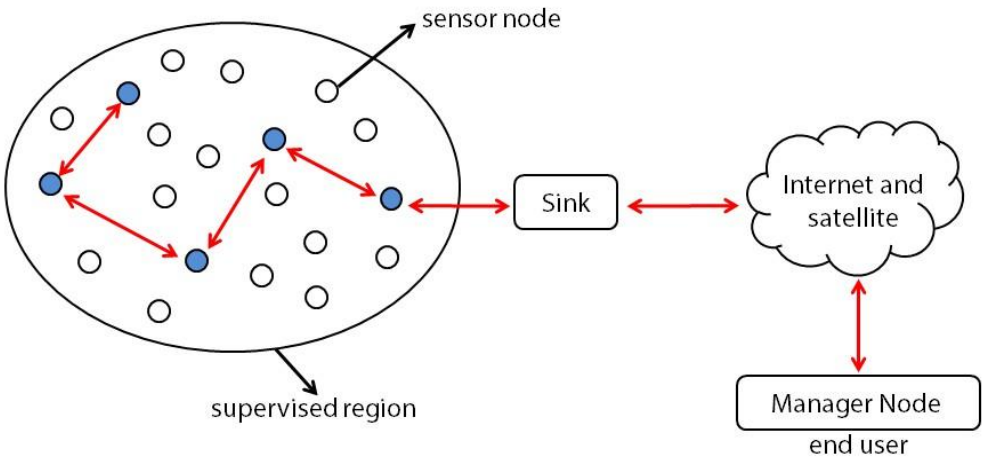


Figure 1: General Network Structure of WSN [1]

A typical application scenario of WSN can be described as: Sensor nodes are randomly deployed in

a supervised area. Once they are deployed, they build an ad-hoc network by broadcasting a message to communicate with their neighbours. Whenever there is a query issued by observers, the query can be disseminated through the network then sensor nodes take actions required by the query, namely sending the measured data to designated nodes i.e. sink nodes by multi-hop routing. Finally, relevant data would be transferred to the base station for further analysis.

1.1 Evolution and Applications of WSNs

There are many awkward issues in the area of WSN, which are associated with deployment cost, hardware constraints, protocols and system reliability etc. To understand those issues, it is helpful to briefly examine their history. Like many advanced technologies, the origin of WSNs was born by military applications. They are used in military detection such as detecting and tracking Soviet submarines in the earliest days. As wireless communication and electronics technologies develop, they are not limited to military and heavy industrial applications. Specially, recent advances in semiconductor, networking and material science technologies are driving the ubiquitous deployment of large-scale WSNs, which differ greatly from those deployed 10 years ago. Thanks to these technologies, the state-of-the-art WSNs have lower deployment and maintenance costs, last longer and are more resilient [2]. Because of these features, we can find sensors are widely used in our homes, work places and beyond, bringing new sources of information and convenience to human's lives. Examples of how WSNs can increase productivity, reduce cost and even save people's lives can be shown in **Table 1**.

Table 1: WSNs Bring a Multitude of Benefits to Our Lives [2]

Application Field	Benefits
Agriculture: e.g. Measure micro-climates on farms; Measure soil humidity	Increase crop yield per square km; Improve the utilization of water resources
Industry: e.g. Electrical/gas/water metering	Optimize utility distribution systems and reduce inefficiencies
Human Life: e.g. Detect Human presence in homes and offices; Smart house	Reduce wasted power in HVAC and lighting; Make human life more convenient
Transportation: e.g. Monitor traffic on road systems	Steer traffic away from jams, accidents and construction zones; alert emergency services
Environmental Monitoring: e.g. Forest fire detection; Landslide detection; sensing volcanoes	Prevent natural disasters; Saving people's lives and reduce economic loss

1.2 Challenges in Today's WSNs

WSN researchers are faced with many challenges regarding different aspects: hardware, software, routing protocols and security etc. The challenges that I focus on is those associated with routing protocols and algorithms. Although many protocols and algorithms have existed in the field of traditional wireless ad hoc networks, they are not well suited to the specific requirements of today's WSNs. To understand this point, we should first figure out the differences between WSNs and traditional ad hoc networks, which are: 1) sensor nodes are usually densely deployed and of a large amount of number that can be several orders of magnitude higher than the nodes in an ad hoc network [1]; 2) the topology of WSNs can change much more frequent than ad hoc networks due to each sensor node can play the role of router in a WSN; 3) Sensor nodes in a WSN are limited in power, computational capabilities and memory while nodes of traditional ad hoc networks are usually equipped with powerful CPUs and sufficient power supply; 4) Sensor nodes may not have global ID like IP address to identify themselves.

Hence, we can see that the challenges of WSN routing protocols and algorithms on one hand lie in WSNs' intrinsic characteristics: limited power resources, dynamic network topology and limited storage and computation capability of sensors, on the other hand lie in users' requirements on response-time (or latency), accuracy and correctness. To address these challenges is actually finding an optimal trade-off among responding time, accuracy and correctness considering the constraints of WSNs. To be more specific, in a large scale sensor network, hundreds or thousands of sensor nodes are involved, so it is challenging to find ways to fast gather relevant data from such a huge amount of nodes. And sensors cannot perform resource intensive computation due to resource constraints and low calculation capability. Thus, sensor network protocol or algorithm design should be adapted to these features.

1.3 Research Motivation

To address the aforementioned challenges, we should first realize that users may have different requirements on the response, especially on latency, correctness and accuracy. For instance, considering a temperature monitoring system, we can imagine a large distributed system and a large number of relevant sensor data. Users probably want a fast response of the rough temperature level, such as cold, warm or hot. On the other hand, users may want a correct answer, as they would be disappointed if the temperature is cool but the system reports it as hot. The accuracy is probably of

minor relevance here, as an answer like "the temperature is cool" is probably enough. Thus in such a use-case we could trade-off accuracy for response-time. Besides, users might want a fast response when they want to check if there is a fire accident. Therefore, my focus of this project is to propose a query evaluation strategy that can find optimal trade-offs considering different user requirements and balance energy consumption of the whole network.

By exploiting the birthday paradox and a tool that is useful for mapping the sensor nodes to physical locations -- Distributed Quad-Tree (DQT) [7], I proposed an anytime-enabled query evaluation strategy for a distributed WSN. The main idea is that I handle different cases with different resolutions. Those resolutions are all based on DQT construction, which do great help to fast data routing. Because by sectioning the deployment area using DQT construction, sensor nodes within a small cell share the same DQTID such that structure tree of sensor nodes are greatly simplified (see Section 3.1.2). For the average temperature query case, energy saving and fast relevant data collection are the main concerns, so I exploit DQT and 0-1 programming to fast select temporary head nodes with relatively high remaining power that can do query distribution and data gathering tasks. For the fire accident query case, the main concern is response-time so I mainly explored birthday paradox and use DQT to distribute data uniformly. This process is as: every time a sensor updates a data indicating accidents, it will distribute a certain number of data replicas throughout the network, and DQT would be used to control the distribution process to let it be a uniformed distribution. Similarly, once there is a corresponding query, a certain number of query replicas will be uniformly distributed in the network. Once these two kinds of replicas (accident data replicas and query replicas) meet, we can consider the query is successful. And according to birthday paradox, if the number of replicas is proper, the rendezvous probability can be nearly 100%, which will be verified in Section 3.3.1.

Contributions: our contributions for the trade-offs of WSNs are threefold: 1) achieving fast and adaptive responses to different types of queries; 2) reducing the cost of communication and energy; 3) achieving better scalability.

Organisation of the report: I describe backgrounds of WSNs in Chapter 2. Detailed description of my proposed anytime-enabled query evaluation strategy is presented in Chapter 3, which first shows the overview of the strategy then follows two models that handle different cases: event query case and non-event query case. Chapter 4 presents simulation results and analysis. The last chapter derives conclusions and points out further work of this project.

Chapter 2: Background

There are many kinds of schemes, algorithms, structures, and architectures for WSN to solve a particular problem. Different schemes have their own characteristics that can fit best in particular situations. Much of the previous research has focused on individual algorithms in each branch of WSN research such as routing protocols, topologies and energy-efficient storage schemes etc. To properly tailor my own query evaluation strategy, it is necessary to study different types of WSN architectures, different types of queries and some common schemes used in different applications. From such study, I consider how to design a query strategy that can control its response-speed adaptively while try to reduce energy usage in a scalable sensor network.

2.1 WSN Architectures

There are three architectures of WSN classified by the way they store data: Local Storage, External Storage and Data-centric Storage (shown as **Figure 2**). These three ways have their respective strengths and weakness and can be best fit to different WSN applications.

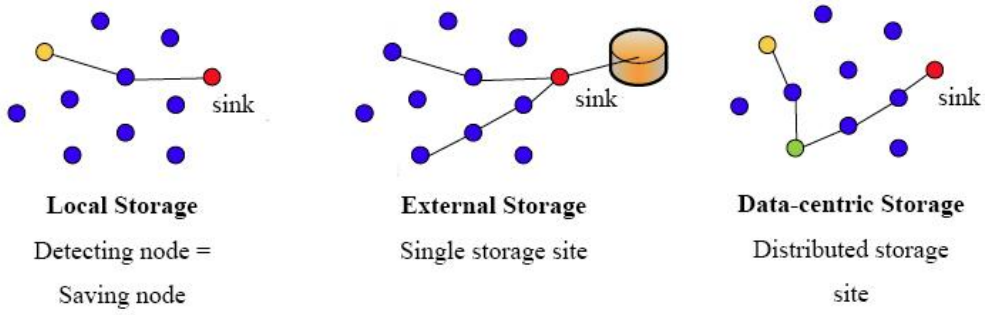


Figure 2: Three Ways of Data Storage in WSN [2]

These three manners of data storage determine three typical WSN architectures, which is discussed below.

Distributed Architecture: it is the extreme case that there would be as many data stores as sensors, which effectively means that there are no data stores: **each sensor is its own data store**, and all sensors cooperate in evaluating queries. This approach costs least communication since communication only occurs during query evaluation, but the drawback is also obvious: query evaluation is much more complex.

Centralized Architecture: all data of a WSN is continuously collected **in a single central data store**. On this data store you evaluate all queries. In this case, "scalability" means that also for large networks the '**necessary data**' (for correctness and accuracy) can be pumped '**fast enough**' (for response-time) into the central data store. The necessary data are always present in a central place, such that query evaluation is a minor problem. On the downside, this approach involves a lot of communication.

Decentralized Architecture: this architecture would have **a set of distributed data stores in the network**, which cooperate in collecting data and evaluating queries. And it is somewhere in the middle between the distributed architecture and centralized architecture. It balances the communication cost and complexity of query evaluation to some extent but may cause a hot spot problem.

Here, our research is considering the decentralized architecture. We evaluate the query on each data store and merge the results. But we could not follow the traditional method to designate specified nodes as the data store since such nodes would run out of energy much faster than the others and would cause hot spot problem in a network. We use the tool -- DQT to dynamically choose data store by taking the communication of advertising and remaining energy into account.

2.2 Types of Queries

Some previous work has categorized the queries that can be used in sensor network applications or in monitoring systems. There are different criteria for query classification, such as continuous versus oneshot queries [6]. Nowadays, there are plethoric strategies designed to fit a certain type of queries, among which two common types are event query and range query.

Event query: event query is a type of query for checking whether a predefined event happened in a region, which is a major part of querying services in WSNs. For instance, an observer may need to know the location of a fire or a soldier in a battlefield may need to know the location of the nearest enemy tank. This type of query involves sense data of only one or several nodes, if we rout the query to the base station for evaluation every time, WSNs would suffer high latency and high energy consumption.

Non-event query: one typical example of the non-event queries is range query, which is defined as a query that requests all the objects falling into a given range of interest [7]. Some examples of range querying are finding all rooms with light on or listing all locations whose temperature is

above 65F etc. In cases of range query, a naive method is to index every sensor data, which is similar with the indexing technology used in traditional search engine. But due to energy and storage constraints, traditional indexing technology cannot be deployable in practice for WSNs.

In this report, we would consider a query evaluation strategy that can address the mentioned issues thereby deriving a strategy that can fit event and non-event querying. This means that our strategy allows users themselves to create specific, customized queries. Typically this would be done via a user-friendly GUI-based system without the need to know the database schema behind [8]. To achieve this goal, we introduce the concept of certainty. If the result satisfies the user-defined certainty thresholds, it is deemed acceptable, and the query is answered using corresponding models instead of diving deeply into the network to gather the data. Further, users can define to what degree the accuracy of the answer should be. Overall view of this strategy will be presented in Chapter 3.

2.3 Routing Protocols and Algorithms

In a WSN, we need not only storage, but also a scheme for forwarding the data to the designated nodes from the sensing sources. That means, WSNs' service cannot be fulfilled without being supported by data dissemination or routing protocols. Models for data delivery required by sensor network applications are classified into continuous, event-driven, and observer-initiated [3]. Continuous data delivery means data are sending to some stores periodically by multi-hops, which is useful for some periodic monitoring systems. Event-driven delivery is that an event triggers an action, which is usually sensing or forwarding data to a sink. Observer-initiated delivery is that source nodes do gathering or forwarding data whenever there is a query issued. No matter which delivery scheme is used, the forwarding process is confirmed to the algorithms and protocols applied. Thus, great effort has been put into WSN protocols design and forwarding algorithms design. Three typical protocols in this field are discussed below.

Flooding: this scheme requires no need to maintain network topologies or related routing computation. It works as: whenever a node receives a message, it broadcasts the message to all its neighbours then neighbours broadcast to sub-neighbours and continue. For instance, if a source node wants to send a message to the target node, it first advertises data replicas to every neighbour, who sends the data to its neighbours recursively until the data reaches the target node or expires the Time to Live (TTL). This is a naive method which has high communication expense and does take the energy-efficiency into consideration.

Gossiping: gossiping is a derivation of flooding. Instead of sending the incoming packets to all its neighbours, a sensor node sends the packets to a neighbour selected randomly [1]. When the neighbour receives the packet, it sends it to another sensor node randomly chosen from its neighbours and this process continues until the packet reaches the destination. Though this approach reduces communication, the problem is that it takes a long time to go through all sensor nodes.

Sequential Assignment Routing (SAR): The SAR protocol takes energy and shortest path into consideration, so it selects the path based on energy resources, additive QoS metric of each path and the packet's priority level [1]. These features are based on multiple trees, the roots of which are neighbors of one-hop from the sink node. And most nodes actually belong to several trees such that it a sensor node can choose a tree of best performance to relay data back to the sink node. The problem is also obvious that it adds more complexity on the algorithm executed on each node.

Along with various protocols, a set of algorithms that performs network organization, topology maintenance, energy management etc. are proposed. Some typical ones are presented in the following:

Hierarchical method: this scheme divides data transmission into levels and a typical example of this is a clustering structure in which sensor nodes are divided into clusters and each clusters has its own cluster head [3]. The cluster heads deal with queries distribution and result gathering. Besides, cluster heads maintain the topology information by regularly communicating with nodes that under their charge. Thus data delivery would be more efficient than full flooding. Another method of this type is that data delivery follows a tree structure, which means that a tree topology of the network is stored in some nodes such that the routing can be predetermined. But high latency may be introduced if the search follows the path of the tree structure strictly.

Greedy Perimeter Stateless Routing (GPSR): GPSR is a novel routing algorithm that suits wireless sensor networks well. It consists of two methods for forwarding packets: *greedy forwarding* and *perimeter forwarding* [4]. The former one is used wherever possible while the later one is used in the regions where greedy forwarding cannot be applied. Greedy forwarding decisions depend on information about router's immediate neighbours which are within the node's communication radius. The process can be illustrated by **Figure 3**. As we can see, the red circle indicates the radio range of D and the blue one stands for the radio range of x. Distance from x to y is the radius of the blue circle and the distance from y to D is the radius of the red circle. Hence the

route: x› y› D is the route that costs least hops. And the perimeter forwarding method uses right-hand rule to map a perimeter such that data forwarding can follow that perimeter.

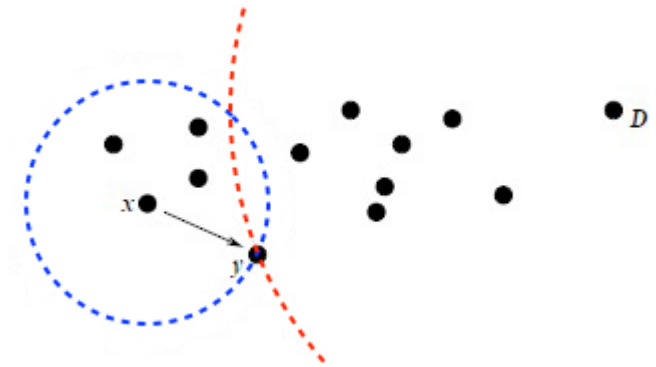


Figure 3: Greedy forwarding example, y is x's closest neighbour to D. [4]

Indexing: indexing technology is very important in WSN networking and it always cooperates with GPSR algorithm [4]. Unlike traditional indexing used in search engine, indexing in WSN uses physical sensor locations in its protocol. The widely used technique is Geographic Hash Table (GHT). It is a static mapping scheme that adds multi-level data according to data range and geometric area coverage. Together, a predetermined hash function is integrated in every sensor node that maps sensed values to the relevant storage nodes [5]. This scheme would be problematic in the presence of node failures, because storage role cannot be implicitly taken over. What's more, due to the distributed nature and limited resource of WSNs, it is impractical to maintain a centralized index to support query processing in WSNs.

The above described method cannot satisfy the query requirements on anytime capability and adaptive queries solely. Thus our interest here is to propose a query evaluation strategy that can efficiently transfer the data at lower communication cost and power consumption. So it should be adaptive to the type of query, i.e. queries can disseminated to all nodes in the network or some parts of nodes of the network. The technique used here is similar to GHT but could be more flexible since every node would no longer maintain geographic information but only a DQTID to identify its neighbours. I discuss this technique in details in Section 3.1.2.

Chapter 3: Design and Implementation

In this Chapter, the detailed design of my proposed strategy for query evaluation is presented. It consists of three main sections, where the first section presents the overview of the strategy, the second section describes a non-event query model and the last section presents an event query model. In the first section, I will start with the problems and design requirements, and then it follows the framework of my strategy.

3.1 Overview of Anytime-Enabled Query Evaluation Strategy

3.1.1 Problems and Algorithm Design Requirements

From the above chapters, we can summarize that the main problems in WSNs are: limited energy of nodes and difficulties in processing queries in a dynamic network of a large scale. To be more specified, we can describe the difficulties for designing a WSN query evaluation strategy as below:

- Due to the energy constraints, we always need to reduce the energy consumed by nodes as much as possible, but there is a dilemma. An optimal strategy always needs to reduce cooperation among nodes thereby reducing communication which cost most energy. However, this will inevitably increase direct communication between nodes and base stations, which will be very harmful for balancing the energy within the whole network. Because the nodes that communicate with base station directly will expire its energy much faster than the others thereby degrading the overall performance of the whole network.

- If the overall topology information i.e. ID and geographic location information of all nodes of a network can be maintained, it is easy to have an energy-efficient, response-fast query processing strategy. However, gathering and maintaining the overall topology information would cost huge amount of communication since the topology of a WSN is always dynamically varying. Further, it is not practical to equip every node with a GPRS device to identify locations.

Considering the aforesaid constraints, design requirements of the strategy I focus on in this report are: any-time enabled, scalable and adaptive.

Any-time enabled: it has two levels of meaning: 1) data is time sensitive which means users can decide the time scope of the result they want and 2) the required result should be given in a certain

period of time that means the responding time should be fast enough for different applications.

Scalable: scalability means that also for large networks the ‘necessary data’ can be pumped ‘fast enough’ into the central data store. It requires the strategy can work well despite the network size may increase from hundreds of nodes to thousands of nodes.

Adaptive: it requires the query evaluation strategy should be adaptive to different user requirements. For instance, some users want fast response at the sacrifice of accuracy while some others want results with certain accuracy degree at the sacrifice of response-time.

3.1.2 Framework: Sensor Deployment Area Sectioning

In this section, a useful tool for virtually mapping sensors to physical locations is presented, which is called Distributed Quad-Tree (DQT). The idea of DQT construction was introduced in [7]. In contrast to extensive usage of quad-trees in a centralized manner [10], DQT is completely distributed and stateless. It achieves this feat by employing an encoding technique that maps a quad-tree over the deployment area by exploiting the location information [7]. In the following, the construction of DQT and mapping from localization to DQT addressing are explained.

Refined DQT Structure and Construction

For constructing DQT, the network is divided into grid cells according to the area of the supervised region. At the same time, we embed a DQT over the network. Thus the number of cells is determined by the area of given region such that all sensor nodes in the adjacent four cells are within each other’s communication radius. Therefore all nodes in the four cells are within one hop distance. For instance, if the area of supervised is near 2R*2R, we divide the sensor deployment plane into 16 cells. And if the area is near 4R*4R, the number of cells would be 64.

Here, we employ an encoding trick first describe in [11] to encode each cell. As you can see from **Figure 4**, the number marked in each cell is actually the DQTID of all sensor nodes within a cell. And four numbers (0, 1, 2, 3) are used to stand for four directions (NW, NE, SW, and SE) respectively. When there are only 4 cells, one digit numbers are enough to represent all cells. As the number of cells increase 4 times, two digits should be used. For instance, as shown in 2R*2R rectangle in **Figure 4**, each NW, SE, SW and SE partition is divided into four sub-partitions again.

Then the first digit of DQTID in each partition remains the same while the second digit varies from 0 to 3 clockwise. Take the NW partition as an example, 00, 01, 02, 03 cells are belong to the NW region from the view of the whole plane, so they have the same number 0 as the first digit. Since the region is divided into four sub-regions again, the second digit is made by deciding which direction it belongs to from the view of the one-fourth plane. Therefore a regular pattern can be deduced for 4R*4R plane, 8R*8R plane etc.

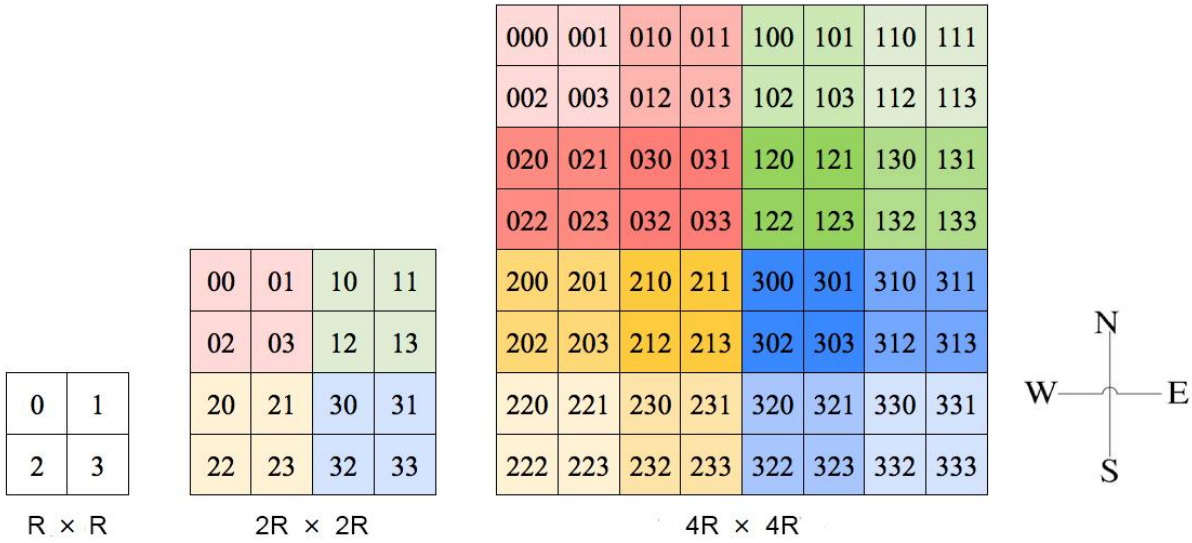


Figure 4: Construction of DQT

This structure is quite simple and since the construction of DQT does not depend on sensor values, no extra communication cost is introduced in the construction process. What's amazing is that it avoids using complex geographic information to maintain neighbour information tables. By exploiting this feat, we can derive a simple, fast and efficient method to distributed query replicas and data replicas (shown in Section 3.3.2).

Mapping from Location to DQT addressing

We assume that the WSN sensor nodes on a two-dimensional plane and their coordinates (x_i, y_i) are made available to themselves once they are deployed. In Section 4.1 we have look at how to assign DQTID to each cell. In this section, we will explore how sensor nodes can calculate DQTID for themselves by given coordinate.

Basically, we divide the plane into cells of a multiple of 4, and each cell is small rectangle with same width i.e. w and length i.e. l equal to R. Thus if we designate the number of cells as 4^m, the

letter m stands for the number of digits that we need to encode each cell. For instance, for 4R*4R rectangle, the number of cells is $64 = 4^3$ where 3 is the number of digits of DQTID. And the DQTID of a node (x_i, y_i) can be calculated as:

$$\text{DQTID} = \left[\left\lfloor \frac{x_i - x_s}{w} \right\rfloor (binary)\right] + \left[\left\lfloor \frac{y_i - y_s}{l} \right\rfloor (binary)\right] * 2 \quad (0)$$

In the above formula, (x_s, y_s) stands for the coordinate of the vertex at the NW corner.

Equation (0) calculates the X and Y address separately and then adds them together. Inside the bracket, binary number system is used while outside decimal numbers are considered. We can verify this formula from Figure 7. Take cell encoded as 013 as an example. Assuming coordinate of the NW corner vertex is (0,0) and a sensor node in 013 cell has coordinate as (3.5, 1.4), the we have

$$\text{DQTID} = \left[\left\lfloor \frac{3.5 - 0}{1} \right\rfloor (binary)\right] + \left[\left\lfloor \frac{1.4 - 0}{1} \right\rfloor (binary)\right] * 2$$

$$= [3(binary)] + [1(binary)] * 2$$

$$= 011 + 002 = 013$$

Basing on DQT construction, two useful models for range-based query and event-based query has been proposed by [7]. But there are some problems with the models: 1) static hierarchical configuration and cluster head selection lead to unbalanced energy consumption at various levels since high-level nodes are more frequently used thereby being prone to deplete 2) remaining energy of nodes is not taking into account in the cluster head validate algorithm which is not energy-efficient. Thus, I tailor DQT construction to my strategy to handle event and non-event query, which will be introduced in Section 3.2 and Section 3.3.

3.1.3 Assumptions

Considering the aforesaid problems and the convenience to properly apply the birthday paradox, we should make some basic assumptions for our query evaluation strategy, which are shown below:

- n sensor nodes are randomly uniformly distributed in a two-dimension rectangle plane and all of them will not move once they are deployed.
- Sensor nodes are not equipped with location identifying devices that they only know their own and their neighbors' information i.e. each node have a unique coordinate (x,y) and a DQTID (not unique for every node but used to identify a region).

- Communication radius of each node is $\sqrt{2}$ R to assume a connected network.
- Communication radius of each node is limited, but the network remains connected since nodes far away can be reached by multi-hop.
- Energy consumption is proportional to the number of transmissions.
- Every node can communicate directly with the base station but only do this when they are entitled this right i.e. to be selected as temporary proxy node.

3.1.4 Query Evaluation Strategy Process

The query evaluation strategy can be adaptive to different user requirements therefore it can take actions according to the queries received. Here it means that the user can query a predefined event or query a result with certain accuracy. In our specific use case, users can query if there is fire potential in the supervised area or can query the temperature of the region with certain accuracy degree (like the average temperature or a region is cool or cold etc.). To achieve this feat, the whole evaluation strategy process can be divided into two parts: **query issuing** and **data gathering**, which are illustrated as flowcharts below (**Figure 5** and **Figure 6** respectively).

The flow chart of query issuing indicates that before the query is issue, the deployment of sensor nodes should be finished first. Deployment mainly deals with sectioning the plane using encoding tricks introduced by DQT and calculating the DQTID for every sensor nodes. After all sensor nodes are deployed, they can initiate communications with neighbours to maintain neighbour information tables in their own memory, which can be used later when there is need to transfer data. After all is set, queries can be issue to seek interested information in the WSN. As shown in **Figure 5**, there are two sub-strategies to deal with different kinds of queries: event query and non-event query.

In my project non-event query algorithm is verified by a use case that users can query approximate average temperature which is divided into three grades: cold, warm, hot, or users can query the accurate average temperature and get an answer measured in centigrade. Event query algorithm is verified by a use case that users can query if there is a firm alarm in the supervised region. For the former use case, the process of selecting temporary head nodes is the most important part. Temporary head nodes act like cluster nodes that can gather data or distribute data in a given area by costing less communication and power consumption. The details of the process will be discussed in Section 3.2. And the most importance part of later use case is the process of uniformly distributing query and data replicas. If the query replicas and data replicas are both uniformly

distributed in the network, the rendezvous probability of them can be illustrated by birthday paradox (see Section 3.3).

Figure 5 shows the flow chart of issuing queries. Actions of this process are performed both by the management node and sensor nodes.

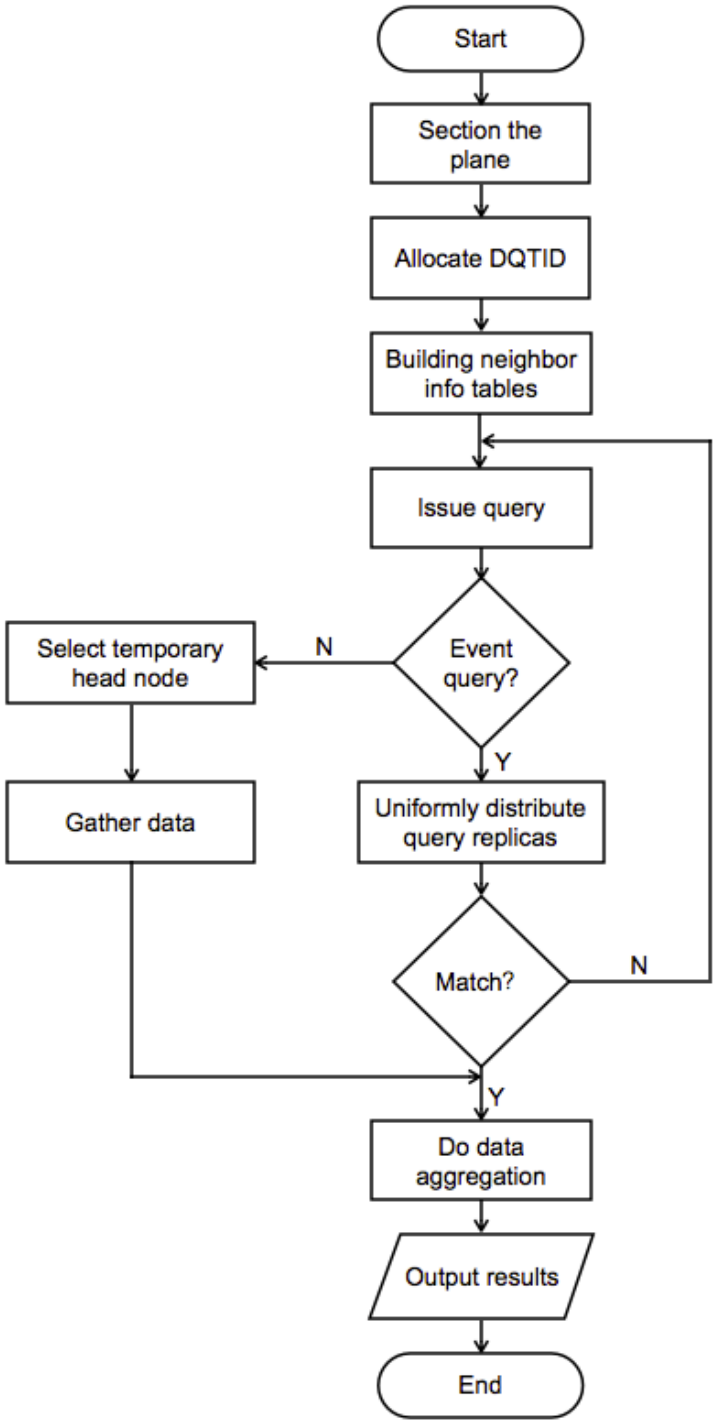


Figure 5: Query Issuing Flow Chart

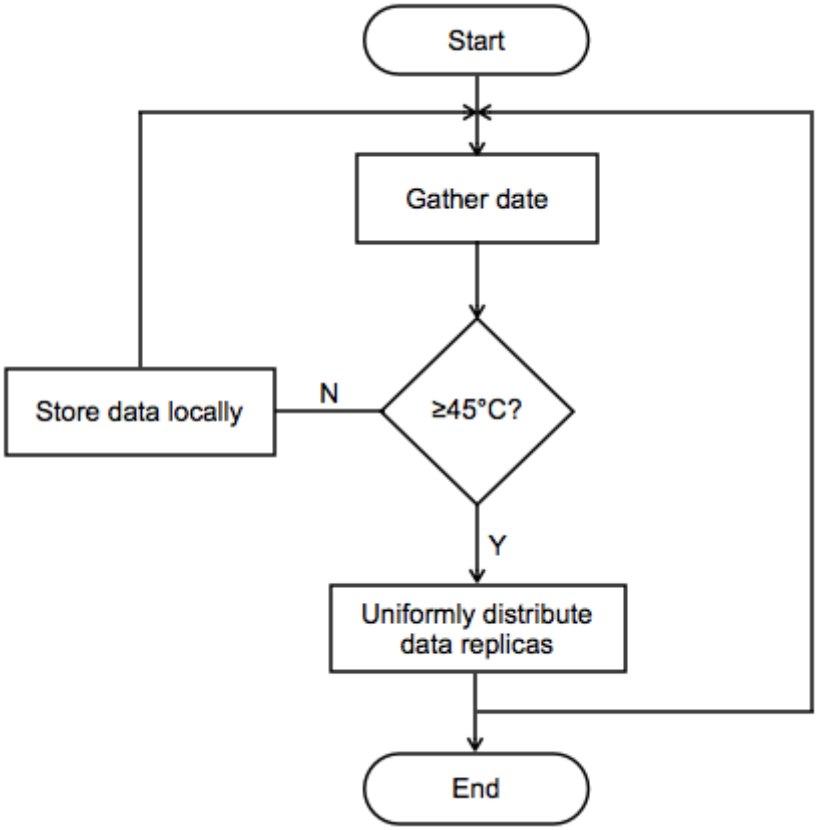


Figure 6: Data Gathering Flow Chart

The flow chart of data gathering: Figure 6 illustrates what actions should be taken by sensor nodes when the temperature is over 45℃ and under 45℃, since in this report, we only consider one type of data – temperature.

In the following two sections, details of the whole strategy will be illustrated by separating two kinds of queries.

3.2 Non-Event Query Model

3.2.1 Temporary Head Nodes Selection

As mentioned in Section 3.1.4, if the temperature is less than 45℃ then the updated data will be stored locally and only when corresponding queries are issue, temporary head nodes will start to gather information from nodes that they are in charge. Why those head nodes are temporary? In Section 3.1.2, we have looked at the problems that DQT structure possesses. It is easy to designate

nodes of a cell which is closed to the centre as cluster head, but it will over-frequently utilize some nodes such that they run out of energy fast. To balance the network energy consumption, we want to take energy and distance into account to select temporary head nodes in a sub-region each time there is a relevant query. By choosing a simple effective selection method, we can achieve a rolling effect that avoids over-utilizing some nodes. To reach this goal, we exploit **0-1 programming** here as our selection method, described as below.

We select temporary head nodes in each sub-region (NW, NE, SW and SE partitions) i.e. if we divide the supervised area into 64 cells, we have 5 temporary head nodes: one in each sub-region and one near the centre.

Decision variable x_i determines if a node can be selected as temporary head node.

$$x_i = \begin{cases} 0 & not\ select \\ 1 & be\ selected \end{cases}$$

Mathematical model here is as:

Using equation $\text{sum}\ (x_i) = 1$ to make sure that only one node can be select in each given area.

The objective function is:

$$\max obj = \text{sum}(x_i * z_i) \tag{1}$$

$$s.t. \begin{cases} z_i = \varepsilon\, E_i / E_{init} + (1-\varepsilon)\frac{d_{max}}{d_i} \\ d_i = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \\ \text{sum}\ (x_i) = 1 \end{cases} \tag{2}$$

where $d_{max} = \sqrt{2}R$.

As for the above mathematical model, ε is the factor that controls the weight of energy and distance to the given centre point. E_i is the remaining energy of the node $(x_i,\ y_i)$ while E_{init} is the initial energy. And d_i is the distance from node $(x_i,\ y_i)$ to the centre point $(x_j,\ y_j)$ of the given partition. In the first formula of equation (2), by normalizing the remaining energy and distance to the given centre point, we can make a simple criterion formula z_i to choose the temporary head node of a given sub-region.

3.2.2 Data Gathering

As mentioned in Section 3.1.4, if the issued query is not an event query, temporary head nodes

selection process is begin and afterwards data gathering should be performed by selected nodes. Following the above algorithm, for instance, in a 4R*4R plane we can have 5 temporary head nodes in possible locations shown in **Figure 7**.

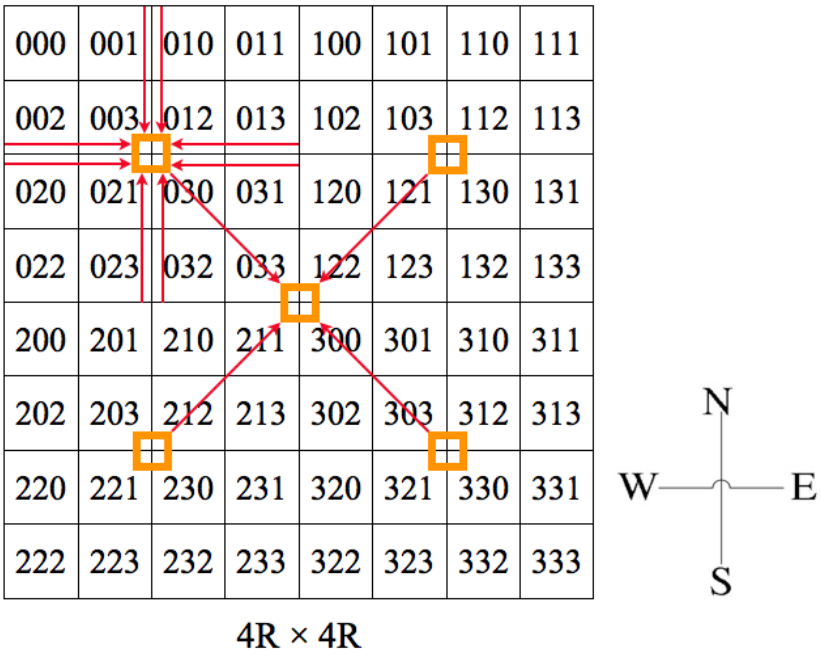


Figure 7: Possible locations of temporary head nods and data forwarding rout

The reason that the possible locations of temporary nodes would be like Figure 8 shown is that those locations are all near centre point of each sub-region. Nodes in these locations would cost less communication while gathering data, especially when the weight of distance normalization is much higher than the energy normalization. The NW partition of Figure 8 shows how data would be gathered by its partition temporary head node. The rest three partitions (NE, SW, SE) also follows same process and data is finally pumped to the centre temporary head node, which will be responsible for returning the result to the base station. Here data transfer is based on neighbour information tables held by each node. As mentioned before, after the neighbour information tables are built throughout the network, it does not cost extra communications thanks to DQT construction. The reason is that the selected temporary head nodes can find its neighbours through local computation. However, temporary head nodes selection would lead to extra communications but it can solve the hot spot problem to some extent, thus better balance the overall network energy consumption.

3.3 Event Query Model

In this chapter, we will discuss the mechanism used to handle event query. This type of query mainly deals with users requirements to know if a predefined event happens in a region. In our specific use case, the observers may want to know if there is fire potential in the supervised area. So we decide that if the temperature exceeds 45℃, data forwarding will follow a tree presented in Section 3.3.3. By doing this, the specific data replicas will be uniformed store in the WSN such that uniformly disseminated query replicas can meet them in some nodes according to birthday paradox. This method will greatly reduce response-time and power consumption.

3.3.1 Basic Theorems: The Birthday Paradox

The proposed strategy in this report is based on the birthday paradox, which concerns the probability that, in a set of n randomly chosen people, some pair of them will have the same birthday [9]. The mathematic relationship between the number of people and the probability of a pair that has the same birthday can be shown as **Figure 8**.

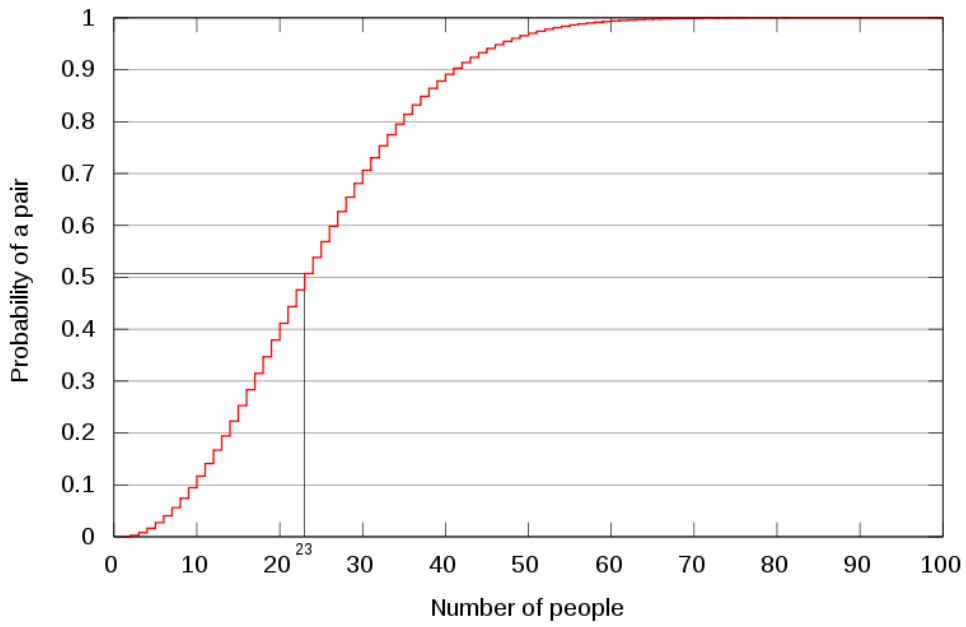


Figure 8: Probability of Same Birthday [9]

From Figure 8, we can see birthday paradox uncovers a fact that, the probability can be almost 100% when the number of people exceeds 100. And 99% probability is reached with just 57 people and 50% probability with 23 people. Actually, by the pigeonhole principle, the probability can be 100% if the number of people reaches 367 (since there are only 366 possible birthdays in total,

including February 29). Of course, these conclusions are based on the assumption that each day of the year is equally probable for a birthday.

Applying the birthday paradox to WSNs, we can see it is practical since the rendezvous of a data replica and a query replica is quite similar with rendezvous of a pair of people with the same birthday. To be more specific, let's assume that every node in a WSN is like a container that can receive both data and query replicas. The scenario can be described as **Figure 9**. When a certain number of both data and query replicas are uniformly disseminated in the network, the probability of rendezvous would vary as the number of replicas varies.

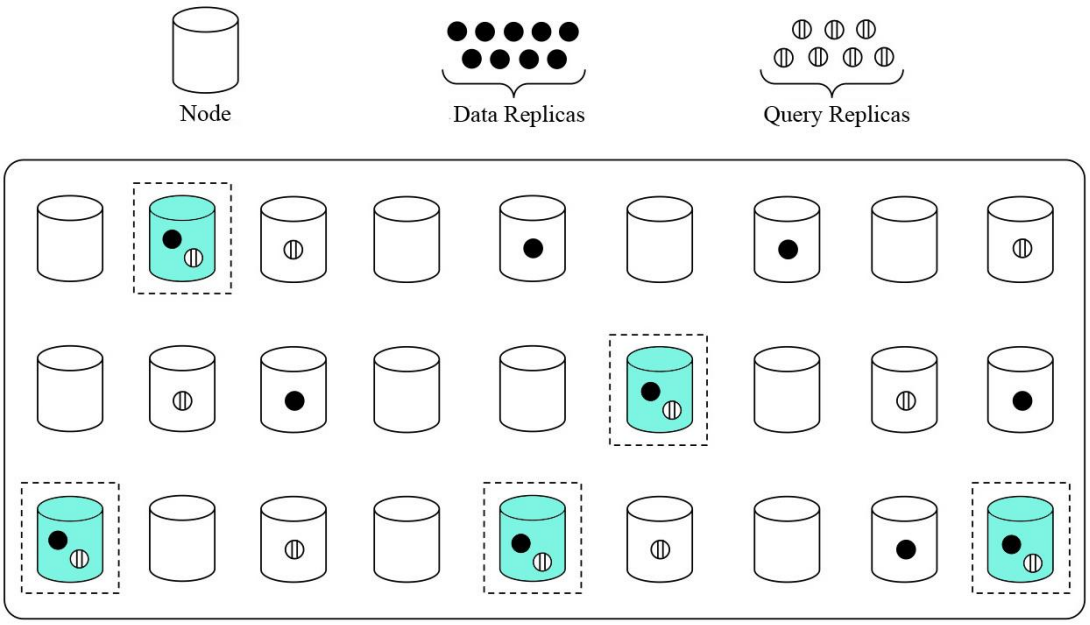


Figure 9: Rendezvous of Data and Query Replicas

If we designate the number data replicas as d and the number of query replicas as q, then according to birthday paradox, the rendezvous probability r of a (d, q) pair is as:

$$r = 1 - e^{-dq/n} \tag{3}$$

where n is the number of nodes in a WSN.

To do some transformation, let $dq = c^2 n$, then gives

$$r = 1 - e^{-dq/n} = 1 - e^{-c^2} \tag{4}$$

The match threshold would be

$$dq = n \ln \frac{1}{1-r} \tag{5}$$

And we can calculate relationship between r and c is shown as **Table 2**:

Table 2: Rendezvous Probability of Data and Query Replicas [13]

c	r
1	63.21%
2	98.17%
3	99.99%
4	99.999 99%

As we can see when c exceeds 3, the probability starts to be closed to 100%. How to determine the number of data and query replicas such that the probability of rendezvous can satisfy users' requirements? For instance, if we have a certain network of 625 nodes, since $d = q$, thus when we choose $c = 3$, it gives $d = q = 75$.

With this theorem, the main task of designing our query evaluation strategy is to propose a fast uniformed distribution algorithm and take the energy consumption into consideration. The details will be introduced in the following section.

3.3.2 Fast and Uniformed Data Distribution Tree

By exploiting DQT, we can build our fast and uniformed data distribution scheme. In [10] a method to build a hierarchical tree is presented, which is not suitable to be employed in our fast and uniformed data distribution process. So we tailor that method to our aim – to find a simple, fast distribution scheme to let query or data replicas be uniformly disseminated in the network, which is the prerequisite to employ the birth paradox.

We conceive a structured tree as **Figure 10**, which is only an example of quad-tree in NW sub-region basing on the construction of DQT. For the remaining three sub-regions similar structure trees are constructed too. Since the communication radius is $\sqrt{2}$R, sensor nodes located in cells marked as 003, 012, 021, 030 can communicate with each other directly i.e. they are in each other's neighbour tables. So by employing a two-level tree, sensor nodes can be traversed. This process is presented in the next section.

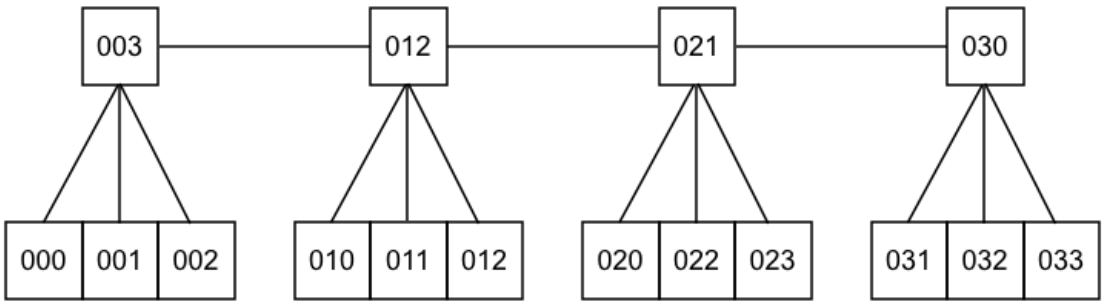


Figure 10: Example of quad-tree in NW sub-region

3.3.3 Data Forwarding

Data forwarding in my project is supported by the gossiping protocol. In the previous section we introduce a two- level quad tree in NW sub-region. Data forwarding is first taken in the sub-region that senses the accident initially and then be passed to the remaining sub-regions. Still take NW sub-region as an example. We assume the initiated sensor node is in cell marked by DQTID 010. First it pass the data to its upper level cell i.e. 012, then 012 cell forwards it to the same level cell: 003, 021 and 030, which will transfer the data to its subordinate cells correspondingly. Of course, there is more than one sensor node in a cell, so when a node in 010 cell want to communicate with nodes in 012 cells, it only need to pick up a sensor node from the group of nodes with DQTID of 012 randomly. The whole process is illustrated in **Figure 11**.

An accident temperature data is first sensed by a node within the cell marked as 010. It knows it superior is the cell marked as 012, thus it look for nodes with DQTID 012 in its neighbour information table and selects one of them to forward the data. Once the node in cell 012 receives the data, it distributes it to its subordinate cells: 011 and 012 and to its sibling cells: 033, 021 and 030.

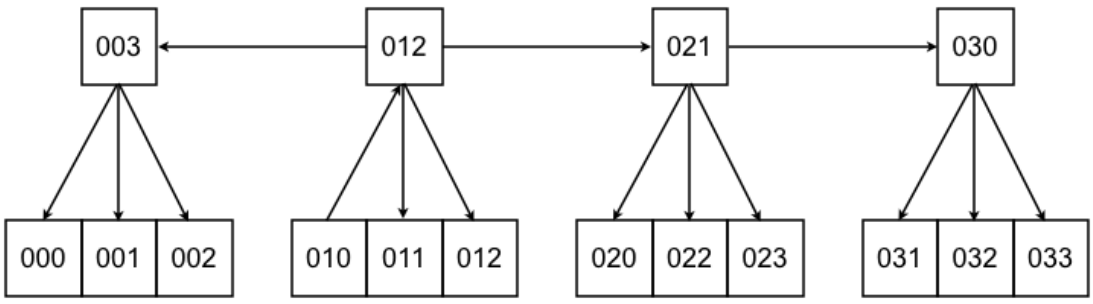


Figure 11: Data forwarding in NW sub-region

To disseminate data uniformly throughout the WSN, other sub-regions also should be taken into account. **Figure 12** indicates the process of transferring data from NW sub-region to NE sub-region. As we can see, we still assume the initiate sensor node is in cell 010 and data forwarding in NW sub-region has been illustrated above. When the data reach cell 033, it can reach cell 122 nearby, which is cell belongs to the NE sub-region. Then similar forwarding tree is constructed in NE sub-region.

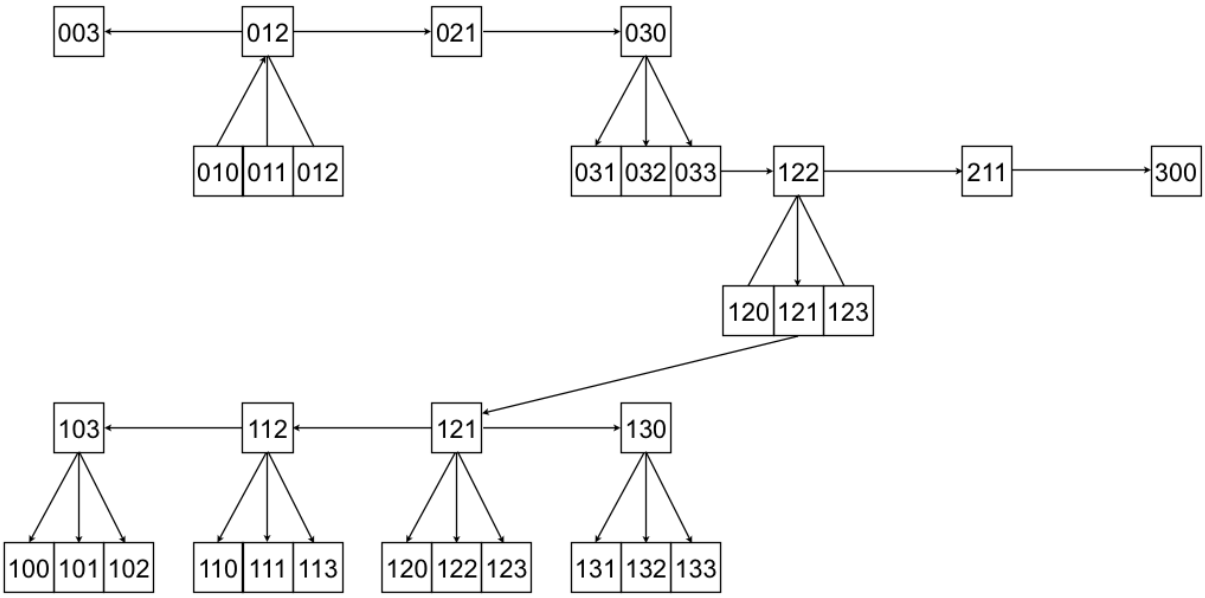


Figure 12: Data forwarding across NW and NE sub-region

Summarizing the above process, we can figure out that four cells in each partition are the first level cells while the others are the second level cells to be in charge by them. And the four cells in the centre area of the plane are used to deal with cross-border transmissions. By applying integrating gossiping protocol with the data forwarding tree discussed above, both data replicas and query replicas can be uniformly distributed in the deployment area in a fast, accurate manner. And according to the birthday paradox, at least one query replica can meet the data replica in one cell such that the query is successful. Actually, by sectioning the plane like this, tree structure of WSN is largely simplified and most computation can be done locally thereby improving the response speed greatly.

Chapter 4: Simulation Results and Discussion

In this research we investigate the performance of the proposed query evaluation strategy using Matlab2012a, truetime toolbox. My simulations and numerical analysis mainly focus on event querying and non-event querying and I analyse both cases' performance from three aspects: scalability, response-time and network power consumption.

Most of my simulations involve the basic scenario described in Section 3.1: I use DQT construction method to section the sensor deployment area and encode each cell, where nodes are uniformly distributed in the deployment area and the transmission range of each sensor node is diagonal line of each cell. In order to test scalability of my proposed strategy, I use four sets of nodes as **Table 3.** As number of nodes i.e. network size varies, three kinds of performance: scalability, response-time and network power consumption are discussed (see Section 4.1 to Section 4.3).

Table 3: Varied Network Size

Number of Nodes	Deployment Area (square meters)
12	60*60 (divided into 2*2 cells)
125	120*120 (divided into 4*4 cells)
625	240*240 (divided into 8*8 cells)
1600	240*240 (divided into 8*8 cells)

Other settings for my simulation are shown in **Table 4**:

Table 4: General Setting of Simulations

Items	Setting
Network type	802.15.4 (ZigBee)
Data rate	800,000 bits/s
Minimum frame size	100 bits
Transmit power	18 dbm
Receiver signal threshold	-48 dbm
Pathloss exponent ($1/distance^x$)	3.5
ACK timeout	0.001s
Simulation time	10 minutes

In the following, I will analyse the simulation results to verify that my proposed strategy has better

scalability, fast response and better balance the network power consumption basing three cases and percentage of transmission failure.

Case 1: query the approximate temperature, which is divided into three grades: Cold (below 10℃); Warm (10 to 28℃); Hot (over 28℃).

Case 2: query the accurate average temperature, which should be answered with an accuracy of up to two digits after the decimal point.

Case 3: query the fire accident, which should be answered with the DQTID of sensor nodes that have sensed a temperature data that is bigger than 45℃.

4.1 Scalability

I have defined scalability in Section 3.1.1: scalability means that also for large networks the 'necessary data' can be pumped 'fast enough' into the central data store. It requires the strategy can work well despite the network size may increase from hundreds of nodes to thousands of nodes. In my simulation, I verify improvement in scalability of my proposed strategy from 3 aspects: percentage of transmission failure, response time and network energy consumption.

First, I change the number of nodes as: 25, 125, 625, 1600 to see how the percentage of transmission failure changes. **As Figure 13 shown**, as the number of nodes increases, the percentage first increase slowly and when the number of nodes reach 1600, the transmission failure rate has a bigger amplification. However, the naive full-flooding algorithm does much worse than my algorithm. Full-flooding algorithm broadcasting data to all neighbours every time it receives it while algorithms of my strategy exploits gossiping and DQTID to forward data, which can reduce redundant data transmission to a large scale thereby achieving better scalability. Thus we can see that though full-flooding algorithm also has a slowly increasing failure rate when the number of nodes is not very big, its increasing trend is much bigger and when the number of nodes reach 1600, the transmission failure rate can even reach 15% in my experiment. Therefore my strategy would have a better scalability in sensor wireless networks.

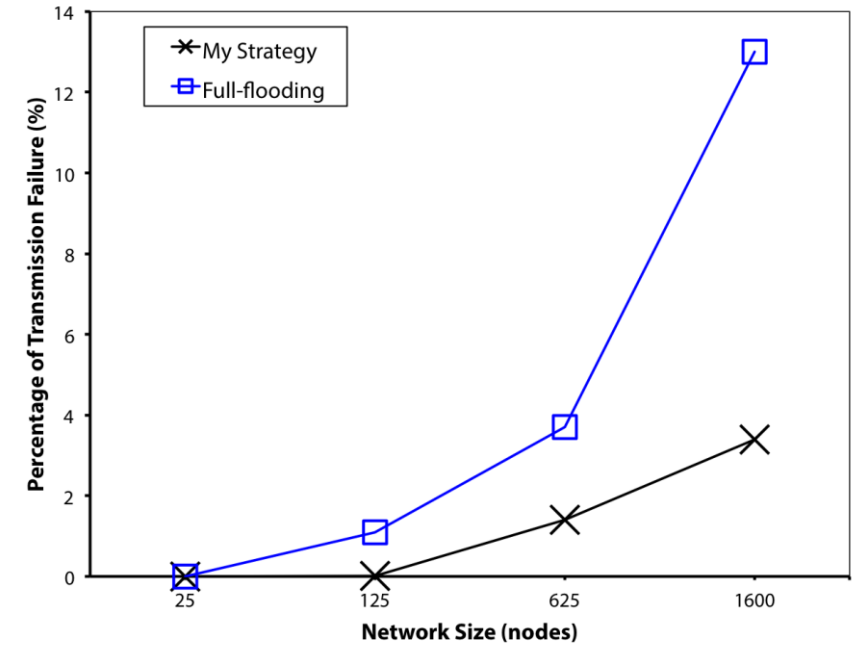


Figure 13: Comparison of Transmission Failure Rate

Then I analyse response-time and network energy consumption of the aforesaid three cases. Response-time and network power consumption indicates performance of the proposed strategy. A strategy of good scalability must have relatively good performance though the network size changes. **Figure 14** and **Figure 15** illustrate how my strategy performs in the predefined three cases.

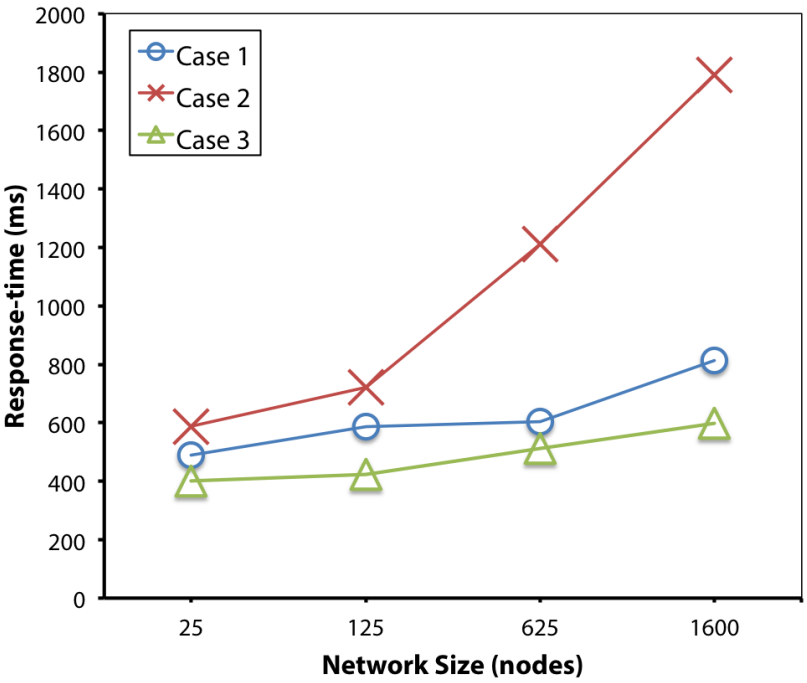


Figure 14: Response-time of Three Cases

In respect of response-time (shown as Figure 14), we can see that my strategy works well for case 1 and case 3, because both case 1 and case 3 involves much less sensor nodes than case 2. Especially for case 3: fire accident query, the response-time increase very slowly. This fact indicates that my strategy can be adapted to large scale sensor network regarding event query. By applying DQT techniques to uniformly distribution data replicas and query replicas, the speed can be greatly increase since the structure tree is much simplified.

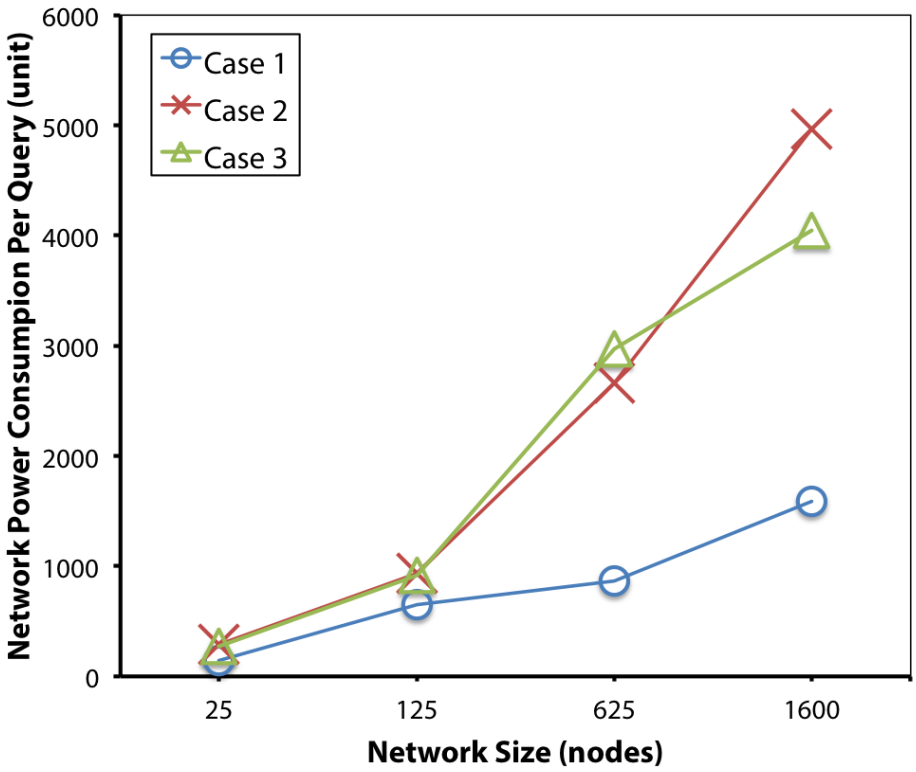


Figure 15: Network Power Consumption per query of 3 cases

As for the performance of network energy consumption, I use power consumption to deduce the communication cost. I have made it clear in the assumptions that the network power consumption is proportional to the communication cost of the whole network, because communication process consumes the most part of node energy. Hence, in my simulation, I define the initial energy of each node as 1000 units. At the beginning of a query, I calculate the remaining energy of the network and do the same task when the query is finished thereby getting the energy consumed by this query. In **Figure 15**, we can see the improvement of case 2 and case 3 is not obvious while case 1 consumed the least energy. The underlying reason is that I trade-off energy saving for response-time for the event query so it is normal that case 3 performs like this. In an event query, what users care most is the response-time. Case 2 involves most sensor nodes so it also performs not very well in response-

time but due to I use a mathematical model (0-1 programming) to avoid hot spot of the network. It is also a trade-off for communication cost.

4.2 Response-time

To discuss the response-time of my strategy, I compare my strategy with a naive method: full-flooding. As discussed in Section 4.1, full-flooding algorithm would cause much redundant transmission and if lack of an efficient algorithm to support the data forwarding process, response-time is hard to improve in a sensor network of large scale. In my simulation, I only implement the case of querying the average temperature of a region to see how full-flooding method and my strategy work. The result is as **Figure 16** shown. It indicates that my strategy outperforms the full-flooding especially the number of nodes is very huge.

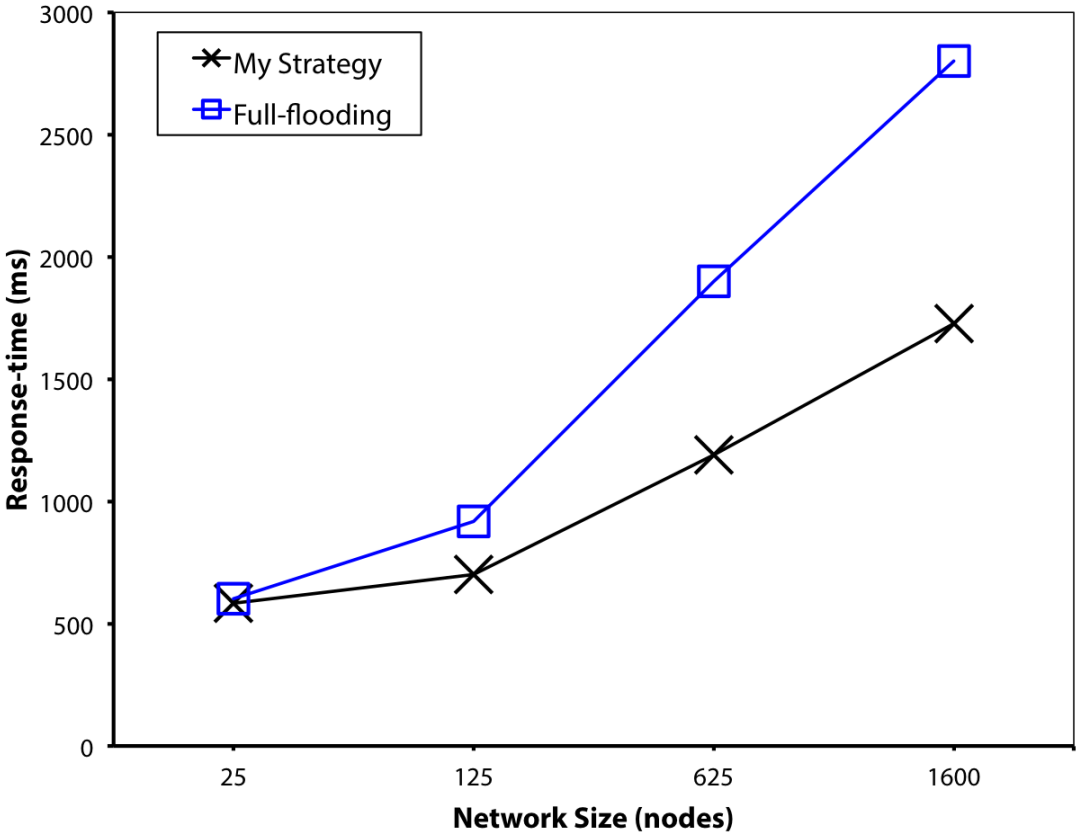


Figure 16: Response-time Comparison

4.3 Network Power Consumption

Energy saving is an important research aspect of WSNs, so network power consumption is also an important criterion to verify improvement of my strategy. I still use full-flooding algorithm to make comparison. From **Figure 17**, we can see that when network size is of 25 nodes and 125 nodes, the performance of both methods is similar. But as the number of nodes increases, performance differs. The trend of power consumption of full-flooding has a sharp slope while my strategy only has a slight increase. Hence, my strategy is much suitable for scalable sensor network. And as I describe in Section 4.1, I define the initial energy of each node as 1000 units. At the beginning of a query, I calculate the remaining energy of the network and do the same task when the query is finished thereby getting the energy consumed by this query.

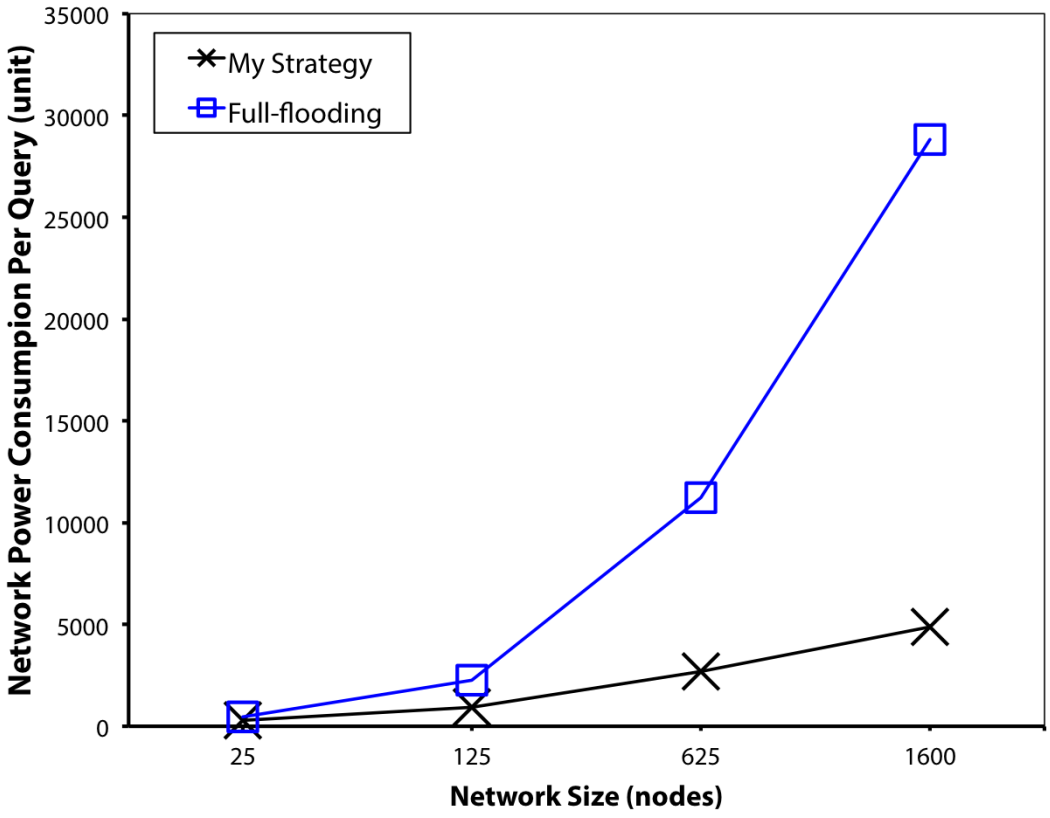


Figure 17: Energy Consumption Comparison

Chapter 5: Conclusion and Further Work

5.1 Conclusion

Given various constraints of Today's WSN such as dynamically changing topology, energy constraint and low computation capability of sensor nodes, WSN researchers have made great effort to routing protocols and algorithms in the past decade. All the efforts aim to find an optimal query evaluation strategy to achieve fast response speed, higher accuracy and lower power consumption. The recent years has seen great achievement in this field but there is still a long way to go due to the underlying problems of WSNs.

In this project, I presented an anytime-enabled query evaluation strategy which is suitable for use in scalable wireless sensor network. This strategy achieves good scalability as well as energy-efficiency. Also, it is an adaptive strategy that can achieve fast enough response speed according to different types of queries, especially for event query. The whole strategy is based on the DQT structure, a useful tool for sectioning the deployment area. In this report, two kinds of queries are considered: event query and non-event query, so the strategy consists of two sub-strategies to handle corresponding queries. For event query cases, birthday paradox is used to determine the number of data replicas and query replicas such that rendezvous probability of these two kinds of replicas is almost 100%. Afterwards, DQT structure is used to form a simplified structure tree to do fast uniformed distribution of replicas. This method achieves excellent performance in response speed as the simulation results shown. As for the non-event query, DQT structure and 0-1 programming is exploited. DQT structure is helpful for data routing while the mathematical model of 0-1 programming balances the energy consumption of the whole network, therefore solve the hot spot problem of WSN to some extent.

In summary, the novelty of the proposed strategy is that it applies an amazing querying infrastructure i.e. the DQT structure to the process of uniformed distribution and constructs data forwarding tree in a new manner. Besides, a simple mathematical model is proposed to select head nodes of each sub-region. Actually, considering the low computation capability of each sensor node, simple algorithms have their unique advantages.

5.2 Further Work

The proposed query evaluation strategy greatly improves the response time by taking actions fitting different kinds of queries and balances the overall network energy consumption. However, in practice, the sensor deployment area would not be two-dimensional and not a square. Thus, further work of this research should explore: 1) how to extend the DQT framework to 3-D space and 2) how to map from irregular shapes to squares. Though [7] has proposed an effective method to map an irregular deployment area to a square, it does not fit the need to uniformly distribute data throughout the WSN in our strategy. Because if we just extend the shorter width to make a square, some cells in the plane have no sensor nodes within them at all. If we still forward data to those cells, it can cost much redundant communication. Therefore, we hope to find a tailored DQT structure that can solve this problem in the future.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A survey on sensor networks,” *IEEE Communications Magazine,* vol. 40, no. 8, pp. 102–114, 2002.

[2] Silicon Laboratories Inc, “The Evolution of Wireless Sensor Networks”, N.p., n.d. Web.

[3] Kyungseo Park, “Architectures and Methods for Energy-efficient Querying and Indexing wireless sensor networks,” *The University of Texas at Arlington,* August 2008.

[4] B. Karp and H. T. Kung, “Gpsr: greedy perimeter stateless routing for wireless networks,” in MobiCom ’00: *Proceedings of the 6th annual international conference on Mobile computing and networking,* 2000, pp. 243–254.

[5] Mohamed M. Ali Mohamed and Ashfaq A. Khokhar, “Dynamic Indexing System for Spatio-Temporal Queries in Wireless Sensor Networks,” *12th International Conference on Mobile Data Management* 978-0-7695-4436-6, 2011.

[6] N. Sadagopan, B. Krishnamachari, and A. Helmy, “The acquire mechanism for efficient querying in sensor networks,” in *Proc. of the first International Workshop on Sensor network Protocol and Applications*, 2003.

[7] Murat Demirbas, Xuming Lu and Puneet Singla, “An In-Network Querying Framework for Wireless Sensor Networks,” *IEEE Transactions on Parallel and Distributed Systems,* vol.20, no.8, August 2009.

[8] "Ad Hoc." *Wikipedia*. Wikimedia Foundation, 13 Apr. 2013. Web. 15 Apr. 2013.

[9] "Birthday Problem." *Wikipedia.* Wikimedia Foundation, 13 Apr. 2013. Web. 15 Apr. 2013.

[10] R. Finkel and J.L. Bentley, “Quad Trees: A Data Structure for Retrieval on Composite Keys,” *Acta Informatica,* pp. 1-9, 2007.

[11] M. Demirbas and X. Lu, “Distributed Quad-Tree for Spatial Querying in Wireless Sensor Networks,” *Proc. IEEE Int’l Conf. Comm. (ICC ’07)*, June 2007.

[12] Wesley W, Kangasharju J, Leng C. BubbleStorm: resilient, probabilistic, and exhaustive peer-to-peer search [A]. SIGCOMM [C]. New York. USA, 2007, 49-60.

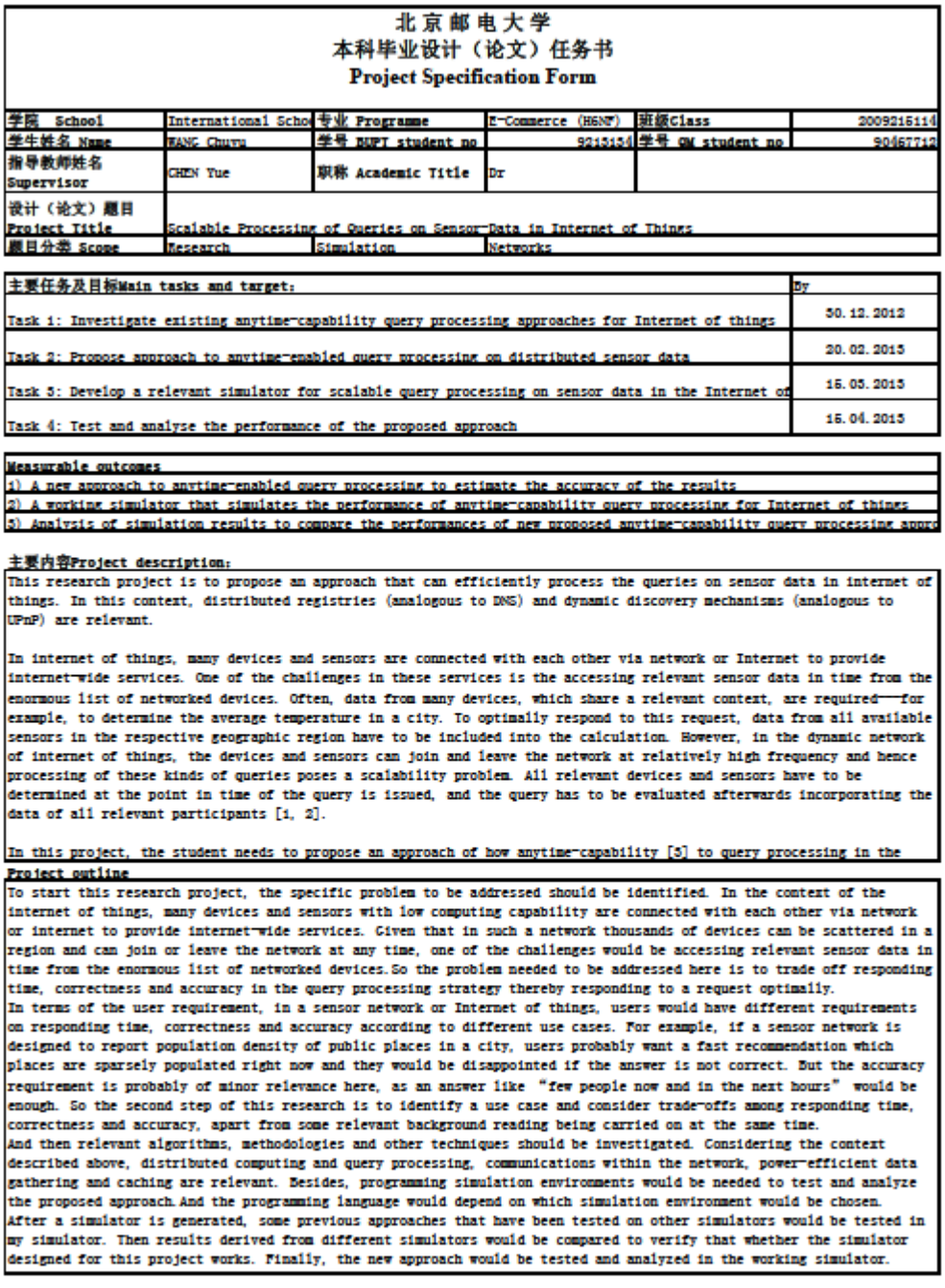
Acknowledgement

This work was especially supported by Martin Alexander Neumann, who is my academic supervisor from Karlsruhe Institute of Technology, Germany and gave great help for me from the beginning of this project. Alex leads me to the field of WSN by patient instruction via Skype meeting and emails. I really appreciate his devotion of time and admire his profound knowledge and gentlemanliness. Though it is a pity that I did not have the chance to go to KIT to meet him face-to-face due to the visa problem, I've deeply influenced by Alex's research spirit and professional attitude.

Yue Chen, my academic supervisor also greatly facilitates the project. Yue mainly gave general instruction that deals with project progress, viva test and simulation. Even she was not in Beijing, she would ask her doctor students to see if there is something can help. And Yue gave me much encouragement to continue the research when we know that we could not get the visa application work, which really impressed me.

One more person I really hope to thank is my good friend – Feng Ye. His acute mathematic sense, skillful Matlab operation ability and the passion to learn helped me overcome all difficulties occurred during the research work. Such an academic intimate has been fighting with me since ten years ago and we are still continuing on this road.

Appendix



北京邮电大学
本科毕业设计（论文）任务书
Project Specification Form

学院 School	International Scho	专业 Programme	E-Commerce (H6NF)	班级Class	2009215114
学生姓名 Name	WANG Chuyu	学号 BUPT student no	9215154	学号 QM student no	90467712
指导教师姓名 Supervisor	CHEN Yue	职称 Academic Title	Dr		
设计（论文）题目 Project Title	Scalable Processing of Queries on Sensor-Data in Internet of Things				
题目分类 Scope	Research	Simulation	Networks		

主要任务及目标Main tasks and target:	By
Task 1: Investigate existing anytime-capability query processing approaches for Internet of things	30.12.2012
Task 2: Propose approach to anytime-enabled query processing on distributed sensor data	20.02.2013
Task 3: Develop a relevant simulator for scalable query processing on sensor data in the Internet of	15.05.2013
Task 4: Test and analyse the performance of the proposed approach	15.04.2013

Measurable outcomes
1) A new approach to anytime-enabled query processing to estimate the accuracy of the results
2) A working simulator that simulates the performance of anytime-capability query processing for Internet of things
3) Analysis of simulation results to compare the performances of new proposed anytime-capability query processing appro

主要内容Project description:

This research project is to propose an approach that can efficiently process the queries on sensor data in internet of things. In this context, distributed registries (analogous to DNS) and dynamic discovery mechanisms (analogous to UPnP) are relevant.

In internet of things, many devices and sensors are connected with each other via network or Internet to provide internet-wide services. One of the challenges in these services is the accessing relevant sensor data in time from the enormous list of networked devices. Often, data from many devices, which share a relevant context, are required—for example, to determine the average temperature in a city. To optimally respond to this request, data from all available sensors in the respective geographic region have to be included into the calculation. However, in the dynamic network of internet of things, the devices and sensors can join and leave the network at relatively high frequency and hence processing of these kinds of queries poses a scalability problem. All relevant devices and sensors have to be determined at the point in time of the query is issued, and the query has to be evaluated afterwards incorporating the data of all relevant participants [1, 2].

In this project, the student needs to propose an approach of how anytime-capability [3] to query processing in the

Project outline

To start this research project, the specific problem to be addressed should be identified. In the context of the internet of things, many devices and sensors with low computing capability are connected with each other via network or internet to provide internet-wide services. Given that in such a network thousands of devices can be scattered in a region and can join or leave the network at any time, one of the challenges would be accessing relevant sensor data in time from the enormous list of networked devices. So the problem needed to be addressed here is to trade off responding time, correctness and accuracy in the query processing strategy thereby responding to a request optimally.
In terms of the user requirement, in a sensor network or Internet of things, users would have different requirements on responding time, correctness and accuracy according to different use cases. For example, if a sensor network is designed to report population density of public places in a city, users probably want a fast recommendation which places are sparsely populated right now and they would be disappointed if the answer is not correct. But the accuracy requirement is probably of minor relevance here, as an answer like "few people now and in the next hours" would be enough. So the second step of this research is to identify a use case and consider trade-offs among responding time, correctness and accuracy, apart from some relevant background reading being carried on at the same time.
And then relevant algorithms, methodologies and other techniques should be investigated. Considering the context described above, distributed computing and query processing, communications within the network, power-efficient data gathering and caching are relevant. Besides, programming simulation environments would be needed to test and analyze the proposed approach. And the programming language would depend on which simulation environment would be chosen.
After a simulator is generated, some previous approaches that have been tested on other simulators would be tested in my simulator. Then results derived from different simulators would be compared to verify that whether the simulator designed for this project works. Finally, the new approach would be tested and analyzed in the working simulator.

Risk Assessment

My project is a research project that deals with algorithm design and simulating it via MATALB. Thus, the software MATALB and my laptop have played an important role in my project. Here are the problems that I concerned.

Description of Risk	Description of Impact	Likelihood Rating	Impact Rating	Preventative Actions
Inappropriate operation on MATLAB	Level 1: Minor Being unfamiliar with the operation and grammar of MATLAB slows down the project and cause some unforeseen problems.	Level 2: unlikely	Level 3: Low risk	Learn to master MATLAB properly from the very beginning of this project.
MATLAB breaks down	Level 3: Very Serious Due to the unstable of MATLAB when the program is running, the codes which are unsaved could be lost.	Level 2: unlikely	Level 6: Moderate risk	Remember to save the codes every 5 minutes or before test new codes.
The number of nodes to be simulated	Level 3: Very Serious Different amount of nodes need to be simulated to verify the algorithm, from 25 to 1600, but the memory of my laptop may not be able to handle thousands of nodes.	Level 3: moderate	Level 9: Significant Risk	Write simple functions with few nodes first and increase the number step by step to evaluate if the memory can handle the node.

Environmental Impact Assessment

Since my project is all about algorithm research and simulation, which can be done by my laptop, there is little impact on the environment. But my laptop still cause some negative impact on the environment, because it needs to continue consuming electricity and generates a certain amount of CO_2 during the whole simulation process.