QoS ENABLED CROSS-LAYERED CLUSTERING FOR MITIGATING FLOODING QUERIES IN INTERNET OF THING NETWORKS

FAWAD ALI KHAN

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITI OF MALAYA
KUALA LUMPUR

2022

QoS ENABLED CROSS-LAYERED CLUSTERING FOR MITIGATING FLOODING QUERIES IN INTERNET OF THING NETWORKS

FAWAD ALI KHAN

THESIS SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITI OF MALAYA
KUALA LUMPUR

2022

UNIVERSITY OF MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: Fawad Ali Khan

Matric No: WHA150064 (17035745/1)

Name of Degree: Doctor of Philosophy

Title of Thesis: QoS Enabled Cross-Layered Clustering For Mitigating Flooding Queries In Internet Of Thing Networks.

Field of Study: Mobile Computing (Computer Science)

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work.
(2) This Work is original.
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyrighted work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work.
(4) I do not have any actual knowledge, nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every right in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work, I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action, or any other action as may be determined by UM.

Candidate's Signature Date: 1 August 2022

Subscribed and solemnly declared before,

Witness's Signature Date: 01 AUGUST 2022

Name:

Designation:

QoS ENABLED CROSS-LAYERED CLUSTERING FOR MITIGATING FLOODING QUERIES IN INTERNET OF THING NETWORKS

ABSTRACT

The Internet of Things (IoT) has received a lot of attention in recent years since it connects everyday things across a wide range of applications and domains. The IoT is intended to improve human lives through the rapid creation of resource-constrained gadgets and the rising connectivity of physical embedded devices that interact using present Internet infrastructure. To exchange queries among heterogeneous IoT networks, numerous sensors require bandwidth and network resources. Network flooding is a vital method for successful query exchange. However, the risk of the intended flooding queries is that they will result in unwanted and redundant network queries, resulting in increased network traffic. The leading cause of inefficient resources utilization is redundant, unwanted and flooding queries. Therefore, in this interconnected world of resource-controlled gadgets, the key issues are to mitigate redundant, unwanted and flooding queries. In this way, IoT devices need a lot of energy and take a long time to compute. More queries lead to increased bandwidth consumption and poor Quality-of-Service (QoS). Existing techniques are primarily concerned with how to speed up fundamental routing, have limited features, and only give QoS solutions to particular IoT layers. However, a sole QoS enabled cross-layered solutions for flooding suitable for both physical and network layers devices have not been studied yet. In this research, A QoS enabled cross-layered clustering (Cluster Based Flooding) an interoperable solution for network and sensor layer devices is proposed. The proposed system can minimize energy consumption, delay, network flooding, detect and eliminate redundant flooding queries by using a query control mechanism (QCM). The core idea behind the Cluster based flooding (CBF) is to split the network into various clusters. Inside the cluster Intralayer cluster (IALC) is responsible to maintain the local query information proactively. While outside the cluster

Interlayer cluster (IELC) is used to reactively transfer the routing queries outside the cluster. The CBF is a hybrid method that can be more effective compared to conventional systems in terms of query traffic generation. However, if proper redundant query detection and termination mechanisms are not used, the CBF may generate more control traffic than typical flooding techniques. In this study, we employed the Cooja network simulator to assess the QoS performance of the proposed CBF. Based on the simulation results, the proposed technique is superior to traditional flooding and state-of-the-art in terms of traffic delays, network throughput and energy consumption under various performance metrics. Further, this study also contributes a testbed that is based on real-time scenarios.

Keywords: Quality-of-Service (QoS) , Internet of Things, redundant queries, clustering, flooding.

QoS DIDAYAKAN PENGGUMPULAN MERENTAS LAPISAN UNTUK MENGURANGKAN PERTANYAAN BANJIR DALAM INTERNET RANGKAIAN BENDA

ABSTRAK

Internet Benda (IoT) semakin mendapat perhatian dalam tahun kebelakangan ini kerana keupayaan untuk menghubungkan pelbagai peranti dalam segenap aplikasi dan domain. IoT bertujuan untuk menambah baik kehidupan manusia melalui penciptaan deras peranti yang berkekangan sumber dan juga peningkatan keterhubungan peranti terbenam fizikal yang berhubung menggunakan infrastruktur Internet sedia ada. Untuk saling bertukar kueri dalam kalangan rangkaian IoT heterogen, banyak pengesan memerlukan lebar jalur dan sumber rangkaian. Pembanjiran rangkaian adalah satu kaedah penting untuk menukar kueri dengan jayanya. Walau bagaimanapun, risiko pembanjiran kueri adalah ia dapat mengakibatkan kueri rangkaian lewah dan tidak diperlukan, sekali gus meningkatkan trafik rangkaian. Punca utama penggunaan sumber yang kurang cekap adalah kueri yang tidak diperlukan, lewah dan pembanjiran. Oleh itu, dalam dunia yang saling terhubung melalui peranti dengan sumber terkawal, isu utama adalah untuk mengelakkan kueri yang tidak diperlukan, lewah dan pembanjiran. Melalui kaedah ini, peranti IoT memerlukan tenaga yang banyak dan mengambil masa yang lama untuk berfungsi. Peningkatan kueri mengakibatkan peningkatan penggunaan lebar jalur dan kualiti perkhidmatan (QoS) yang lemah. Kaedah sedia ada lebih menumpukan tentang cara meningkatkan kelajuan penghalaan di peringkat asas, mempunyai ciri-ciri yang terhad dan memberi penyelesaian QoS kepada lapisan IoT yang tertentu sahaja. Walau bagaimanapun, sebuah penyelesaian tunggal silang lapisan menggunakan QoS untuk pembanjiran yang sesuai bagi kedua-dua lapisan fizikal dan rangkaian masih belum diteroka. Dalam kajian ini, sebuah penyelesaian kluster silang lapisan menggunakan QoS (Cluster

Based Flooding) untuk peranti lapisan rangkaian dan pengesan telah dicadangkan. Sistem cadangan ini dapat mengurangkan penggunaan tenaga, lengah, pembanjiran rangkaian, mengesan dan menyingkirkan pembanjiran kueri lewah menggunakan mekanisma kawalan kueri (QCM). Idea utam di sebalik pembanjiran berasaskan kluster (CBF) adalah untuk memisahkan rangkaian kepada pelbagai kluster. Kluster antara-lapisan kluster dalaman (IALC) bertanggungjawab memelihara maklumat kueri tempatan secara proaktif. Sementara kluster antara-lapisan kluster luaran (IELC) digunakan untuk memindahkan kueri penghalaan secara reaktif di luar kluster. CBF adalah satu kaedah hibrid yang lebih efektif jika dibandingkan dengan sistem konvensional dari segi penjanaan trafik kueri. Walau bagaimanapun, jika pengesanan dan penyingkiran kueri lewah yang sesuai tidak digunakan, CBF berkemungkinan menjana lebih banyak trafik kawalan daripada kaedah pembanjiran biasa. Kajian ini telah menggunakan penyelaku rangkaian Cooja untuk menilai prestasi QoS dalam CBF yang dicadangkan. Berdasarkan kepada keputusan simulasi, kaedah yang dicadangkan lebih baik daripada kaedah pembanjiran biasa dan juga kaedah terkini dari segi kelengahan trafik, pengeluaran rangkaian dan penggunaan tenaga di bawah pelbagai penilaian prestasi. Tambahan lagi, kajian ini turut menyumbang sebuah tapak uji yang berdasarkan senario masa nyata.

Kata kunci: Kualiti Perkhidmatan (QoS), Internet Benda, kueri lewah, kluster, pembanjiran

ACKNOWLEDGEMENTS

First and foremost, I would like to express my profound gratitude to Almighty Allah for his goodness and mercy throughout my study. I want to extend my special thanks to my supervisor, Professor Dr. Rafidah Md Noor, for her immense contributions toward realizing all the works presented in this thesis. You made me see far beyond my peers by standing on your giant academic experiences and knowledge. Your time, continuous support, kindness, care, and thoughtful consideration made my journey less stressful. I would also like to thank Professor DR. Miss Laiha Binti Mat Kiah for being always disposed to help me in the uncountable obstacles that we eluded in the road to complete this work.

In the same way, I would like to express my special thanks to my mother, and my father Hashtmat Ali Qureshi, for their moral and financial support throughout my Ph.D. I am deeply grateful to my wife and son, for their love, care, prayer, and emotional support throughout my Ph.D. I am also thankful to Mr. and Mrs. Saifullah Khan for their motivational support. It would not have been possible for me to reach the hill of doctoral-level research work in a foreign country without your prayers and emotional support. I also would like to thank my Lil chums Zubair Ali Qureshi and Prof Sb for their continuous support to accomplish this task. It's my pleasure to extend my gratitude to Qurat ul Ain Mastoi, who always stands behind me in my difficult times.

Last but not least, I am indebted to my mother and my father for their sacrifices for me. I got my strength to continue my Ph.D. thesis from my family and their kind words which encouraged me to reach this stage.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

X_s	:	Duration of the mote
X_f	:	Processing time flooder mote
C	:	Speed of light
D1, D2, and D3	:	Distances
M	:	Maximum query request packets of neighbors
X	:	Net sum of motes
d_i	:	Distances between every mote
Pf	:	QueryLimitThreshold (QLT)
Z_0	:	Mote position at the beginning from (x_0 to y_0)
Z_n	:	Mote position at the end from (x_n to y_n)
m(t)	:	Set of queried motes
$m^c(t)$	:	Set of unqueried motes
B(t)	:	Set of peripheral or border motes
t, t_1, and t_2	:	Different time intervals
X1	:	Inspecting mote
X2	:	Flooder mote
I_c	:	Constant inspecting
I_p	:	Periodic inspecting
F_u	:	Utility function
F_d	:	Duration of the flooding attack
F_{dg}	:	Gain of flooding detection
P_{refa}	:	Payoff of the flooding
P_c, P_p	:	Payoffs for (Ic, Ip)
H_0	:	Null Hypothesis
H_1	:	Alternative Hypothesis
μ	:	Mean of sample values
DnC	:	Divide-and-Conquer method
SLA	:	Service-Level Agreements
Hy-IoT	:	Hybrid energy aware clustered protocol for IoT heterogeneous network
QoS	:	Quality of Service
S_D	:	Standard Deviation
∑	:	Summation of a data series
MSR	:	Mean squares for samples
MSE	:	Mean squares for errors
SSR	:	Sum of squares for samples
SSE	:	Sum of squares for errors
QCM	:	Query control mechanism
P	:	Probability

CHAPTER 1: INTRODUCTION

This chapter demonstrates a synopsis of the research which is conducted in this thesis. The research work is focusing on the QoS enabled cross-layered clustering for mitigating flooding queries in IoT (IoT) networks. The overview of the work is explained in several sub-sections as follows. Section 1.1 provides the introduction of the Chapter. Section 1.2 discusses the motivation of the Research Work, and the Statement of the Problem is presented in Section 1.3. Whereas Section 1.4 and Section 1.5 describes the Research Questions and Objectives respectively. Section 1.6 represents a significance and contribution of the study. Finally, Section 1.6 provides a layout of the thesis.

1.1 Introduction

The IoT has gained a tremendous fame in the recent years. Many of our everyday devices are becoming connected to us, spanning a wide range of characteristics such as autonomy, sensing, and contextual awareness (David et al., 2015). IoT is the outcome of the progress of the Internet, and the innovative creation of intelligent equipment helped in the development of recent prototype. The IoT is the next radical technology in transforming today's communication infrastructure into a wholly futuristic network (Krishnapriya, S and Joby, 2015). The IoT is planned to include a massive number of sensors that collect and transmit data on ambient conditions, such as physiological assessments, machine operational data, and so on. The IoT allows numerous sensors and things to connect with each other without the need of human mediation (Koike et al., 2016).

The prime purpose of IoT is to enable data sharing between devices and applications in the real-world environment (Laxmi, P and Deepthi, 2017). Consequently, the IoT is built on the combination of numerous communication systems, identification and

tracking technologies, different sensor and actuator networks, and distributed intelligent devices (Yan-e, 2011). These objects/devices are linked together and communicate with one another via the same network. These gadgets are connected to a sensor, allowing them to detect specific environmental variables, analyze the situation, and respond appropriately. In addition, IoT devices are scheduled to automatically take decisions or advise the user of the best choice (Huang et al., 2017). This interconnected network has the potential to offer significant advancements in the application-technology and services, resulting in economic advantages to worldwide corporate advancement. Several devices are connected to the Internet to communicate local data in information superhighway.

An IoT network comprises hundreds or thousands of small, dispersed autonomous devices known as sensors (Alqahtani et al., 2016). Sensors can work together to detect, quantify, and collect data on physical or environmental variables and then transfer that data into a single hop or multi-hop transmission to one or more sinks, depending on the local decision process. Sinks may receive duplicate and redundant data due to the link between spatial position and data characteristics. As a result, sinks read sensor input in a meaningful way and respond appropriately (White et al., 2017).

Moreover, the IoT has the capacity to provide a smart environment with significant time, energy, resource savings and a high Quality-of-Service (QoS). In the IoT , vibrant resource scheduling for diverse workloads is crucial for maintaining QoS, energy consumption on each mote, and data transmission traffic delay (Dhumane et al., 2016) (Alqahtani et al., 2016). Data transmission in an IoT network is prioritized, energy consumption, QoS, and delay are the most demanding needs for IoT networks (White et al., 2017). Furthermore, it is widely recognized that IoT has the capacity for a broad range of applications like health, agriculture, education, transportation, poultry, supply chain, farming, crop protection, and irrigation (Riadi et al., n.d.).

Heavy control packet exchange between sensor nodes for obtaining an efficient forwarding route to the base station results in reducing overall network lifetime.

Hierarchical transmission in WSN and IoT is considered one of the preferred ways for energy efficient routing of data packets toward the destination. In this kind of architecture, sensor nodes are divided into different layers with different tasks (Masoud et al., 2019).

Clustering is a technique where different sensor nodes divided into groups and sub groups, they transmit their sensed data to the CH node and they in turn forward these packets toward the sink in hierarchical fashion. Recently, different routing techniques were proposed for three layer clustering topology in Wireless Sensor Network (WSN) which outperform the basic two layer clustering hierarchy. The problem that remains in these approaches is the heavy control packet exchange between nodes after every round in order to choose efficient lower layer heads. Among these techniques is Hybrid Hierarchical Clustering Approach (HHCA). According to HHCA, the upper layer heads are centrally selected by base station, while sensor nodes only have to select lower layer heads distributivity (Ullah et al., 2019) (Lahane & Jariwala, 2021).

Every application necessitates numerous sensors to link and converse with one another, potentially lowering network QoS due to wasteful resource utilization, traffic delays due to superfluous messages/queries, and energy consumption because each device has explicit approach to the cloud resources (O. Liang et al., 2007; Salehi et al., 2013), as illustrated in Figure 1.1.

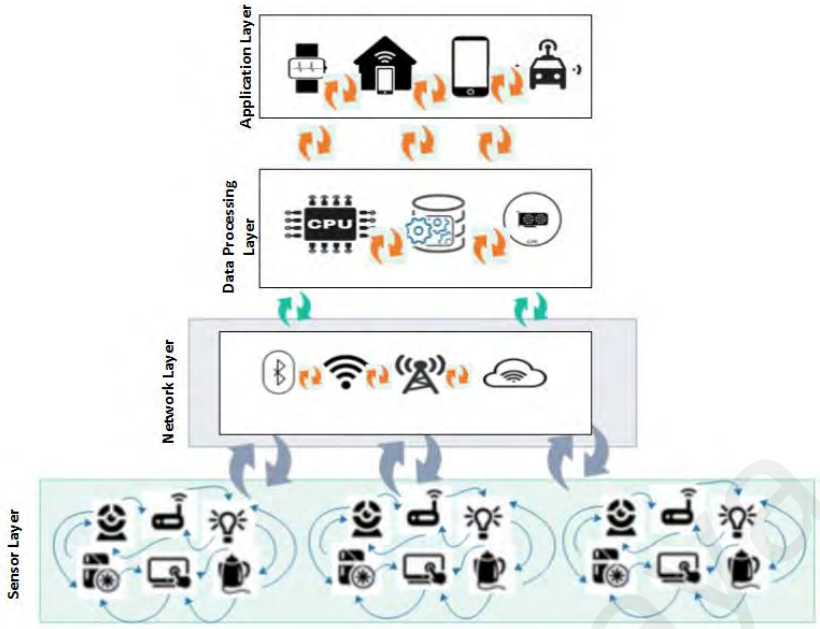


Figure 1.1: Layer Based Communication Model for Heterogeneous IoT Devices.

Cross-layer designed is an escape from the pure waterfall-like concept of the traditional communications model with virtually strict boundaries between layers. Figure 1.1 depicts a communication model with multiple sensor motes interacting with one another at the network and physical (sensor) layers of the IoT (Arkian et al., 2015). Cross-layer designed is an escape from the pure waterfall-like concept of the traditional communications model with virtually strict boundaries between layers. The cross layer approach transports feedback dynamically via the layer boundaries to enable the compensation of QoS parameters such as unwanted routing queries, delay, throughput or other mismatch of requirements and resources by any control input to another layer.

Strict boundaries between layers are enforced in the original networking model, where data is kept strictly within a given layer. Cross-layer removes such strict boundaries to allow communication between layers by permitting one layer to access the data of another layer to exchange information and enable interaction.

Furthermore, due to the energy-constrained nature of IoT devices, resolving these issues in IoT networks is demanding. Currently, no cross-layered mechanisms have been established for the identification and termination of redundant queries in this domain. The following subsection describe the challenges in network querying.

1.1.1 Network Querying

Network querying is regarded as a critical job in a Wireless Sensor Network (WSN) / IoT since it permits users to obtain information based on sensors. The only way to extract the essential information accurately and as per needs of the applications is to query the WSN/IoT. For instance, the authority can collect temperature or humidity data from specific locations to satisfy different goals such as statistical analysis, storage of records, events detection, etc., to meet requirements in a particular range or period in environmental monitoring WSN. In addition, at urgency, in cases of battlefield or in military use, it can be vital to determine the location of opponent's vehicles inside a certain perimeter as soon as possible. From all the above instances, it is detected that the query emitter needs specifically the particular sensor data meeting the query conditions on time location or range. As a result, data recovery from all WSN and IoT sensors are unnecessary and redundant in such instances. Furthermore, it can fail to produce an accurate query result in a timely manner. Therefore, it is vital to create a proper query execution methodology that considers all distinguishing WSN and IoT characteristics. WSN is typically considered as a dispersed and geographically scattered database that obtains data from environment dynamically rather than being entered by an operator. The specification of collaborative querying and job allocation through a straightforward interface that conceals the internal processes within the WSN. In addition, the majority of WSN real-time applications necessitate a quick and accurate query response. Apart from deploying sensors on mobile platforms, one of the most difficult challenges is

gaining access to sensor data, which necessitates the development of effective and distributed systems for query compiling, data organization, and storing. Decisions must be taken regarding the rate of recurrence and nature of data to be collected. Data can, for instance, be taken routinely at regular periods from unified WSNs, or need based, i.e., only when asked from the appropriate WSNs. Other essential issues in IoT and WSN applications include efficient query processing in context of processing time, ideal storage plan, and several operations for large-scale data. Using sensor technology, capturing enormous volumes of data has been very fast; querying and mining large-sized data is computationally expensive, especially when the analysis is required with adequate accuracy in real-time. As a result, implementing delay and accuracy-sensitive query processing approaches in WSNs, and the IoT provides several major research issues that inspire the work discussed in this thesis.

1.2 Motivation

While many studies have been done on various critical areas of IoT like architecture, network protocols, coverage, and so on, providing Quality-of-Service (QoS) of network queries in IoT remains a largely unexplored study area. This is due to the distinctive nature of IoT and WSN in comparison to conventional networks. Therefore, the compact definition the QoS parameters of an IoT query is unclear. Similarly, it applies to the query processing approaches that dynamically support them. The primary Quality-of-Service (QoS) metrics for query processing in heterogeneous IoT networks are energy consumption, delay, and network throughput. Even if low energy consumption is typically considered, the essential design need of IoT, real-time communication in time-sensitive applications such as safety monitoring, object tracing, health monitoring, mission-critical applications, etc., are nevertheless more and more significant. The

assurance of a query response accompanying a shortest likely delay is the fundamental real-time requirement of these applications. In addition, some applications can explicitly specify the time limit when the query is issued and require the best response in the long term. For instance, to detect a crucial hazard scenario like a bushfire, the application needs to know the surroundings temperature, the direction of wind, and other information in a matter of seconds. Due to the emergency, less accurate data can be accepted in such scenarios. In such instances, receiving an estimated but immediate response that meets the delay bound remains the highest priority for detecting the event and initiating the appropriate actions. As a result, maintaining timeliness in giving a real-time priority to a query response with the highest level of accuracy in terms of throughput is critical (Ruan et al., 2020).

Priority query is another critical Quality-of-Service (QoS) metric for IoT query processing such as environment monitoring, machine/structural health maintenance, etc. It is feasible to attain the best throughput in terms of accuracy by gathering the most recent data from all network's sensors to respond to every application-generated query (Thakare et al., 2020). However, given the resource-restricted types of the sensors, this strategy of gathering the entire sensory data at the time of query implementation is not always practical or cost-valuable. Therefore, the most efficient and effective use of energy restricted IoT resources and the least possible query delay are needed to achieve the best resource utilization.

Again, depending on the conditions, some applications require sensor data with shorter delay or higher precision or a mix of specific latency and accuracy requirements. For instance, very accurate sensor data are necessary to maintain the desired environment in typical working settings in a food storage controller. However, changes must be reported immediately (i.e., the high temperature that causes a fire), and the system needs

approximate sensor data to address an incident to regain from any abrupt and unusual change in one or more recorded parameters. (i.e., delay is the top priority).

Furthermore, sensors necessitate network resources and bandwidth to exchange queries among dissimilar IoT networks. For a successful exchange of queries, network flooding is a vital probing strategy. However, the downside of the intended flooding queries is that they will result in unwanted and redundant network queries, resulting in high network traffic. The leading cause of wasteful usage of resources are redundant, undesired, and flooding queries. In this networked world of resource-controlled gadgets, the key issues are alleviating redundant, unwanted, and flooding queries. In addition, because of unwanted and redundant queries, IoT devices use additional energy and require computational time, which leads to more bandwidth consumption and miserable QoS. Existing techniques are primarily concerned with how to speed up fundamental routing, have limited features, and only give QoS solutions to specific IoT layers. However, a sole QoS enabled cross-layered solutions for flooding suitable for both physical and network layers devices has not been studied yet (Premila et al., 2015).

The research presented in this thesis is a step towards improve the QoS of IoT networks by preventing and mitigating the unwanted and redundant flooding queries for both physical and network layers devices and can become the part of future IoT-QoS technologies.

1.3 Problem Statement

In recent decades, the IoT has gained a lot of attention. However, IoT is planned to include a massive number of sensors that collect and transmit data on ambient conditions, physiological assessments, machine operational data, etc (Özdoğan & Ayhan, 2019). Due to the large number of connected devices or sensors and exchange of information on network the redundant and unwanted queries becomes the major cause of

network flooding which leads to an inefficient utilization of resources and may reduce the QoS (Conti et al., 2014)(O. Liang et al., 2007)(J. Jin et al., 2012). Network flooding continues even after the destination found which may affect the overall performance of the network(Benenson et al., 2006)(Gluhak et al., 2011).

IoT devices are more susceptible to redundant and unwanted flooding queries, which may disrupt data transmission, causing them to delays, require more bandwidth and energy to transmit the query to the destination which may reduce the QoS of IoT network in context of energy consumption, cost, delays and network throughput. There is also a lack of mechanism to identify the priority queries from the network (Talal et al., 2019) (Qiu et al., 2012), (Kumar & Chaurasiya, 2019), (Cheng et al., 2018), (Baddeley et al., 2019), (Yamazaki et al., 2020).

There is no compatible solution for priority and redundant undesired flooding queries among all existing studies for physical and network layer devices. In addition, present techniques primarily focus on accelerating basic routing with limited functionality and providing a QoS solution only to a certain IoT layers (Ullah et al., 2019).

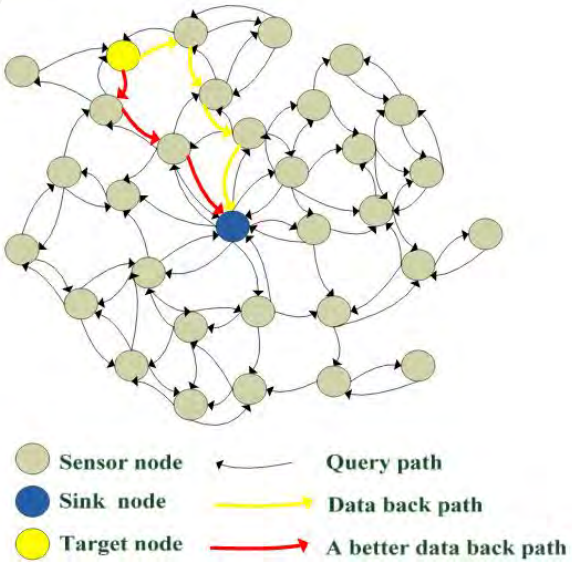


Figure 1.2: A Level-Based Flooding Search Strategy of Internet of Things.

In Figure 1.2 sink / sender mote searching for target / destination mote and target mote send data back reply to sender node in disordered way. Intermediate motes rebroadcast the query automatically which leads to flood the whole network. Generate heavy network traffic and redundant queries by utilizing excessive energy and bandwidth which may leads to more delay and overall degradation of QoS. Note that among all the existing techniques, there is no interoperable solution both for physical and network layer devices. Based on the discussion, it is concluded that IoT demand a QoS enabled cross-layered clustering approach that will help in monitoring network flooding, detect and terminate the redundant queries and reduce the energy consumption, delay and prioritize the queries as it has been neglected to date (Tandon et al., 2021), (Delgado-Rajo et al., 2020).

1.4 Research Question

This study addresses the following research questions to achieve the objectives.

RQ1: *How can we analyze and identify the limitations of existing mechanism used to prioritize, detect and terminate the redundant / unwanted flooding queries in sensor and network layer of IoT networks?*

RQ2: *What mechanism is required to prioritize, detect, and terminate the redundant and unwanted flooding queries for sensor and network layer to enhance the QoS of IoT network?*

RQ3: *How can we formulate cross-layered clustering for redundant and unwanted flooding queries to be developed?*

RQ4: *How can a real-time QoS enabled testbed can detect and terminate the redundant and unwanted queries in IoT network and to reduce the number of duplicate/overlapping queries in IoT networks to improve QoS.*

RQ5: *How does the proposed mechanism improves the network performance in terms of energy consumption, delay, and throughput?*

1.5 Research Aim and Objectives

This research aims to address the problem of redundant queries and network flooding in the IoT networks and to propose a QoS enabled cross-layered clustering approach. Following objectives are defined to be achieved to attain the aim of this research.

1. To investigate the state-of-the-art solution and identify the issues and limitations to prioritize, detect and terminate the redundant and unwanted flooding queries over the sensor and network layer of IoT network.
2. To develop a cross-layered Cluster Based Flooding (CBF) technique for priority and redundant queries. Two new algorithms are introduced as below:
    - Interlayer Clustering (IELC) algorithm for network layer that uses advance query control mechanism (QCM) for detecting and terminating the redundant and unwanted queries and network flooding.
    - Intralayer Clustering (IALC) algorithm for physical layer that maintains priority queries information locally.
3. To formulate the cross-layered Cluster Based Flooding (CBF) using Sets in Prob B.
4. To design real time QoS enabled Query Control Mechanism (QCM) testbed used to detect and terminate the redundant and unwanted queries in IoT

networks. The QCM testbed aims to reduce the number of duplicate/overlapping queries in IoT networks to improve QoS.

5. To evaluate our Cluster Based Flooding (CBF) approach using simulation tools under realistic scenarios and compare the results with the state-of-the-art approaches in the literature as well as validate the results using a statistical analysis tool.

1.6 Significance and Contribution of the study

The significance and key contributions of this dissertation are as follows,

1. An exhaustive and deep review of literature for the state-of-the-art solutions in WSN/IoT to devise a comprehensive thematic taxonomy. The study analyzed each state-of-the-art query solution to identify the distinguishing WSN/IoT features utilized for each solution and the exact problem addressed by a particular technique together with the simulation or emulation environment of the corresponding technique. The critical discussion extended the knowledge of the domain of the current query processing trends in the WSN/IoT networks, the major strengths of potential, and the research gaps that required thorough investigations. This study concluded that a sole QoS enabled cross-layer solution for flooding (suitable for both physical and network layers devices) was not addressed previously.

2. Design of QoS enabled cross-layered clustering technique to mitigate flooding queries in IoT networks. The QoS enabled cross-layered clustering was significantly implemented as interoperable solution both for physical layer and network layer devices. Since the cross-layered CBF divides the whole network into different clusters, being local query information proactively maintained by the IALC, CBF was investigated as potentially more efficient against traditional schemes in terms of query traffic generation. However, the CBF was found insignificant in the absence

of appropriate redundant query detection and termination techniques since the CBF generate more control traffic compared to standard flooding techniques. Thus, the adopted Interlayer clustering (IELC) composed of advanced query detection and termination techniques (QCM), linked signal strength and Query Limit Threshold (QLT) values for detecting flooding. The technique was validated as capable of minimizing the energy consumption, network flooding, and identifying and eliminating unwanted and redundant routing queries in IoT networks. The study also investigated this technique more accountable for checking further locality and query detection strength in an IoT network during flooding. This study observed the strength of query detection for verifying any variation concerning the signal strength of query packet, and the QLT. The QoS enabled solution outperformed the existing solutions.

The outcome of this research contribution was published [Khan, F. A., Noor, R. M., Mat Kiah, M. L., Noor, N. M., Altowaijri, S. M., & Rahman, A. U. (2019). QoS Enabled Layered Based Clustering for Reactive Flooding in the IoT. *Symmetry*, *11*(5), 634].

3. Design of a Formal Method for CBF Employing Event- B Criteria to Deeply Examine the CBF in IoT Context. This study designed formal method inspired by formal verification, significantly enhanced the quality of the verification system, backed by rigorous mathematical proofs. Since the existing studies lacked the important formal validation of cross-layered routing protocols, the formal specification design of cross-layered cluster-based flooding CBF at event B proved the correctness of the route discovery mechanism. Having a refinement-based method, it significantly improved the way to add system details to the corresponding model progressively, resulting in modeling and authentication easier for the user.

4. Development of QoS Enabled QCM Testbed. This study developed a QoS enabled

QCM testbed to detect and mitigate the redundant and unwanted queries in IoT networks. The QCM testbed significantly reduced the number of duplicate/overlapping queries in IoT networks to improve QoS by a prompt calculation of all the overlapping clusters within the query space. The testbed successfully implemented the smart query detection and mitigation to manage the redundant queries issue to provide a better QoS.

5. Inferential Validation of Implemented Outcomes of Research Study. The last research contribution of this study is the successful validation of achieved outcomes of comparative analysis of our solution with the existing studies. The study incorporated the inferential analysis and validation of QoS enabled QCM mechanism employing ANOVA and t-tests. The research outcomes accomplished the 95% confidence interval, and the probability was found less than 0.05, demonstrating the significance of achieving the rejection of NULL hypothesis.

The outcome of this research contribution was published [Khan, F. A., Noor, R. M., Kiah, M. L. M., Ahmedy, I., Yamani, M., Soon, T. K., & Ahmad, M. (2020). Performance Evaluation and Validation of QCM (Query Control Mechanism) for QoS-Enabled Layered-Based Clustering for Reactive Flooding in the IoT. *Sensors*, *20*(1), 283].

1.7 Layout of Thesis

The remainder of this thesis is presented below.

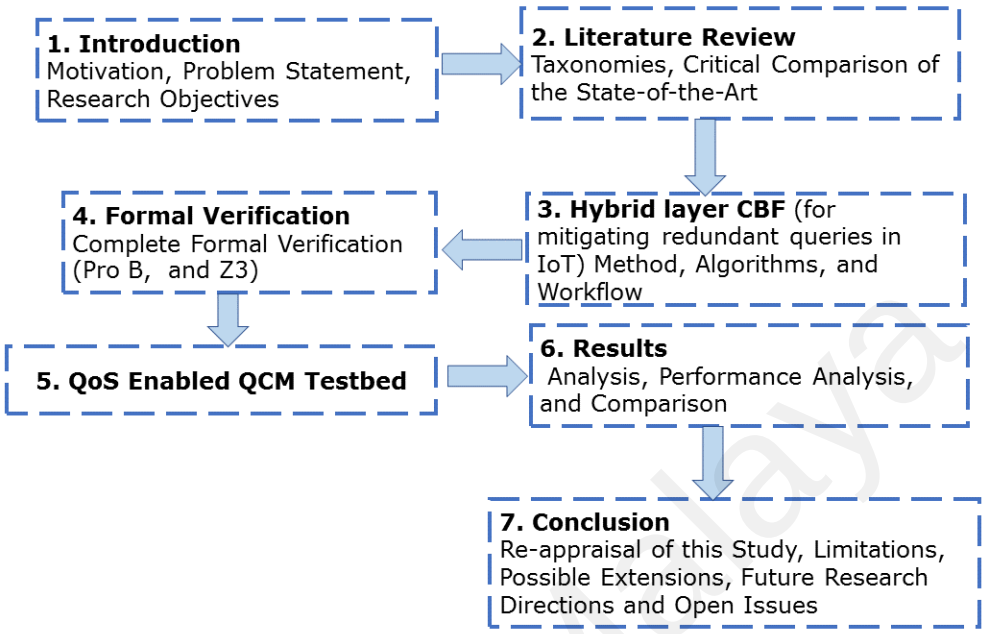


Figure 1.3: Thesis Layout

Chapter 2: This Chapter examines several existing research on sensor data models, query routing paths, and QoS of query processing approaches, as well as risks and challenges related to IoT Layers. The review focuses on our research objectives and covers relevant research into a single WSN and multiple WSNs connected with the IoT. The advantages and disadvantages are discussed for each method, and recommendations for future research are identified.

Chapter 3: This Chapter proposed a QoS enabled cross-layered clustering technique for mitigating flooding queries in IoT networks. It explains the phases in the Cluster Based Flooding (CBF) along with the algorithms IALC, IELC and QCM presented in each phase. The distinct features of the CBF are also highlighted.

Chapter 4: This Chapter presents a formal analysis of the cluster-based flooding (CBF) using Event-B method and as a case study use to examine the CBF in IoT.

Additionally, this Chapter demonstrate system requirements and environment assumptions taken during development. It depicts the whole process of formal development, including the formalization with Event-B and the validation with the ProB.

Chapter 5: This Chapter presents QoS enabled query control mechanism (QCM) testbed used to detect and terminate the redundant and unwanted queries in IoT networks. The QCM testbed aims to reduce the number of duplicate/overlapping queries in IoT networks to improve QoS. The query control mechanism is aware of all the query information. Therefore, all the overlapping clusters in the whole query space can be easily calculated.

Chapter 6: This Chapter reports the results obtained from different experiments and analyzed the effectiveness of our method. This study also compares and contrast experimental results of this research with the benchmark results of the state-of-the-art methods. The statistical validation is performed to know the significant differences between the QCM and state-of-the-art methods.

Chapter 7: This Chapter completes the thesis by explaining how the research objectives have been achieved. It also lists the limitations of study along with the future research directions of the proposed method.

CHAPTER 2: LITERATURE REVIEW

The Chapter previews the Wireless Sensor Networks and IoT architecture, followed by the essential concepts to help readers comprehend the notions of WSN/IoT. This chapter demonstrates a thorough review of the query processing, and associated challenges in heterogeneous WSN/IoT Networks. This Chapter devise a contemporary taxonomy of the reported redundant/unwanted queries in cross layers to clarify the important categories of QoS implications (for each layer of the IoT and wireless sensor network). Additionally, this Chapter comprehensively analyzes the possible unwanted and query threats impacting the QoS. It also focuses on a specific layer in conjunction with a compact solution to design QoS enabled solution to mitigate the redundant queries.

The subsequent Chapter demonstrates the state-of-the-art solutions in WSN/IoT considering the most primitive to the most modern trends. The structure follows the fundamental solutions categories and presents the critical analysis and discussion in formulating a thorough thematic taxonomy. This Chapter also analyzes each state-of-the-art query solution to recognize the differentiating WSN/IoT features employed for individual solutions. It relates the exact problem solved by a specific technique presented with the simulation or emulation environment of the related technique and finally extend the critical discussion based on the domain knowledge of the current query processing trends in the WSN/IoT Networks, the major strengths, and the research gaps required for thorough investigations.

The subsequent of Chapter is as follows. Section 2.1 provides a comprehensive overview of WSN, including its services and potential areas of application. Next, the concept of how distributed and heterogeneous WSN collaborates in the IoT can significantly extend the services provided by the IoT layer and extend the redundant query

threats that the layer faces. Section 2.2 presents an importance of IoT network in detail. Section 2.3 centers on the state of the art on in-network query processing in WSN /IoT. The query handling issues in the distributed and heterogeneous WSNs towards IoT are discussed in Section 2.4. Section 2.5 is charted by a review on the Quality-of-Service (QoS) of WSN/IoT query processing along with cross-layered architecture and challenges. Finally, Section 2.6 concludes with summarizing the Chapter.

2.1 Background

This Section provides a comprehensive overview of WSN, including its services and potential areas of application. Next, the concept of how distributed and heterogeneous WSN collaborates in the IoT can significantly extend the services provided by the IoT layer and extend the redundant query threats that the layer faces.

2.1.1 Local / Stationary Wireless Sensor Network

A local/fixed Wireless Sensor Network (WSN), also called stand-alone WSN, consists of hundreds of small dispersed autonomous devices called "Sensor Nodes" (Yick et al., 2008). In cooperation with sensors, physical or environmental conditions can be sensed, measured, and collected locally. Some WSNs may be equipped with multiple fusion nodes spread between the sensors for specific purposes, such as data collection, data aggregation, or some other pro-active computational operation. Fusion nodes are usually a tiny bit strong than the usual sensor node and used to appropriately control a group of neighboring sensor node. Due to the local decision mechanism, sensors can transmit sensed data to one or more gateways and sink nodes via the fusion nodes in one hop or multi-hop communication process. Sink or gateway nodes have more computing capacity, battery life, and transmission range and relay data in return to the base station as presented in Figure 2.1.

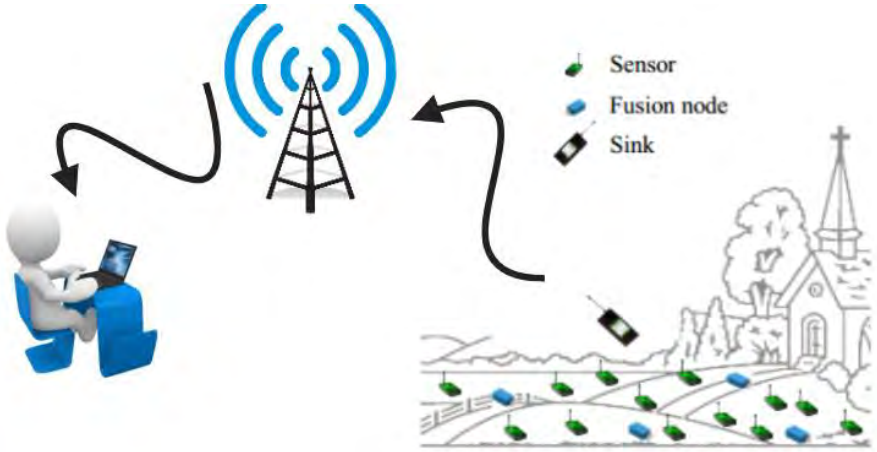


Figure 2.1: Basic WSN Architecture

WSN/IoT infrastructure is either unstructured, with a complicated number of sensor nodes employed ad-hoc, or organized, with fewer sensor nodes installed in a pre-planned manner and proper coverage of the area. Different types of WSN influence different design factors (such as fault tolerance, network size, cost, operating environment, topology, hardware limitations, power consumption, and transmission media). WSN has many distinguishing features that set it apart from other traditional networks such as wireless, MANET, Bluetooth, etc. Sensor nodes can integrate different sensors and transceivers in a resource-limited environment, and They can also organize, transmit and exchange information with end-users in their surroundings. Nodes are generally static upon deployment, and some can self-organize, referring to the system's ability to attain the necessary organizational structures with no human involvement. Each node is powered by a small low-capacity battery and cannot be replaced after a deployment, limiting the architecture of the data flowing through the network to preserve battery life and prevent users from overwhelming. Additional important features of WSN can be defined as intensive computing, data-centric, application-specific architecture, and cross-layered optimization in network protocols (Mukherjee et al., 2021).

The most popular wireless sensor network applications (Akyildiz et al., 2002) can be

categorized as follows and presented in Figure 2.2.

2.1.1.1 Simple Monitoring

Following are few examples of monitoring application scenarios:

- Collection of Environmental Data

Researchers collect sensor data from a series of sensors installed in an area for a number of months or years to identify long-term and seasonal trends in the atmosphere and then analyze the data offline (Cerpa, Elson, Estrin, et al., 2001; Mainwaring et al., 2002).

- Military Surveillance

Through sensor networks, leaders and commanders may track the position, condition, and availability of soldiers, vehicles, equipment, etc.

- Environmental Bio-complexity Mapping

With low cost and operating overhead, sensing technology can detect any form of spatial or temporal resolution of a geometric field (Cerpa, Elson, Estrin, et al., 2001; Cerpa, Elson, Hamilton, et al., 2001; Keitt et al., 1997).

- Human Physiological Data Monitoring

Sensor networks can capture and preserve human psychological data for extended periods, which can then be used for medical research (Noury et al., 2000).

Tracking of Farm Animals | Green House Environmental Monitoring | Fire Hazard Detection

Figure 2.2: Applications of WSN

2.1.1.2 Surveillance of Security or Emergencies

The following are some examples of security or emergency handling applications:

- Surveillance on the Battlefield

Sensors can be placed along every route on the battlefield to keep an eye on all activities.

- Detection of Nuclear, Biological, and Chemical Attacks

In all types of nuclear, biological, or chemical anomalies, sensor networks are used as an alert mechanism.

- Detection of Forest Fire

The fire's source can be detected by remote deployment of sensor nodes in a forest.

- Detection of Flood

A variety of sensors such as rainfall, water level, and weather sensors are used to detect floods.

- Healthcare Diagnostic

Sensor networks can track and detect the behavior of the elderly and thus help doctors to recognize the symptoms on time (Young Han Nam et al., 2002).

2.1.1.3 Object Tracking

Object tracking is another application for sensor networks. It is necessary to track the whereabouts of valuable goods or persons in various situations. Some examples of tracking applications are as follows:

- Control system for Inventory

The sensor network's object tracking function helps to monitor the number of items in a vast inventory and manage them.

- Military targeted Intelligence sensor networks may be used to detect military anomalies.

- Medical monitoring: heart rate, blood pressure, allergies, and other data can be detected using tiny, lightweight sensors that can be affixed to each patient. Doctors may also carry a sensor node to help them find their way around the hospital.

2.1.2 Distributed Wireless Sensor Networks

Today the extensive use of Internet networks worldwide is carried out by devices such as computers and cell phones directly by humans anytime and anywhere. However, the field of information and communication technology has recently introduced a new dimension: accessibility for everything, at any time and from any place. Not only the Internet of the future will enable people to communicate with one another and access information, but it will also allow machines to communicate with one another and with people in their immediate vicinity. It opens up a range of development platforms concerned with increasing intelligence in daily communication than with faster broadband (L. Tan & Wang, 2010). The future Internet is entering a new age of

pervasiveness known as the IoT, in which new modes of communication between humans and things, as well as between things, will emerge. It is reasonable to describe IoT as "The things have identities and virtual individuals operating in smart environments using intelligent interfaces for social, environmental and user context communication " (Zorzi et al., 2010). According to the IoT concept as mentioned in Figure 2.3, the gadgets must be capable of sensing to be aware of their state on numerous physical characteristics such as temperature, humidity, light, speed, proximity, and so on. In addition, cooperation between various WSNs is mandatory to achieve the ubiquitous view of IoT. For instance, controlling a car's speed (a node in a vehicle sensor network) may require road traffic and weather conditions. The former information is retrieved from the vehicle sensor network, while the latter is obtained from environmental monitoring sensors. A driver may also search from the same vehicle for the status of the security devices/home devices that have to be installed on the home network.

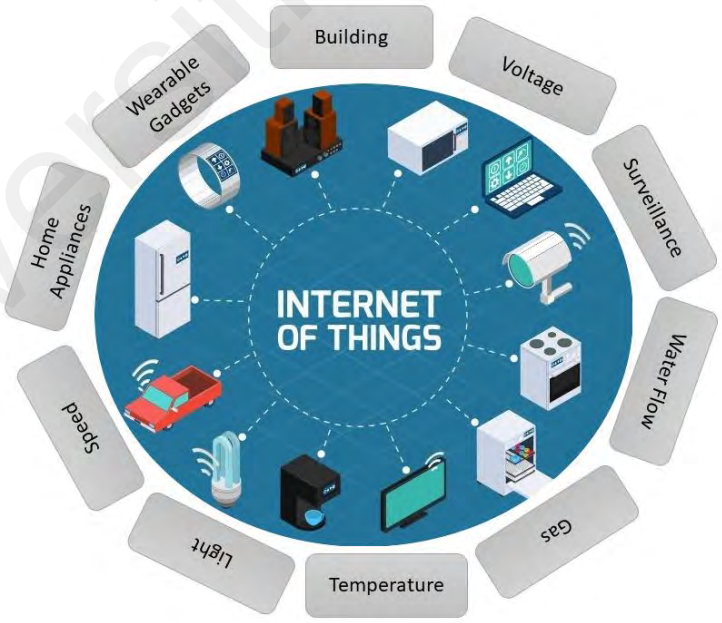


Figure 2.3: Everyday Objects Connected to the Internet

This highlights the significance of distributed and collaborative WSNs in IoT. It introduces the notion of the Sensor Web, characterized as a collection of comprehensive, universally distributed diverse sensors related to one or more WSNs that are linked by a communication material and share data via interoperable interfaces. By using various types of queries, users and web applications can access and control the sensors. This work is concerned with query processing in WSNs and IoT.

2.2 IoT Networks

The rapid growth of IoT, grid computing, cloud computing and distributed Internet-based systems recently led to the explosion of data creation in nearly every area of engineering and business (Hajizadeh & Jafari Navimipour, 2017; Jafari Navimipour & Fouladi, 2017; X. Jin et al., 2015). Also, an increasing quantity of physical objects are being connected rapidly, indicating the IoT concept (Piccialli et al., 2017; Qin et al., 2016; Whitmore et al., 2015). With the introduction of wireless networking, the Internet, and ubiquitous computing, a new concept known as the IoT has emerged; IoT consists of physical devices that can be tracked and managed over the Internet (Mao et al., 2016; Moschakis & Karatza, 2015; Z. Yan et al., 2014). Various actuators and sensors linked to the Internet through a wireless sensor network can monitor billions of things in the IoT (Abdollahzadeh & Navimipour, 2016). The IoT's key characteristics are connectivity, sensing, and accessibility between things (Levi & Sarimurat, 2017). IoT connects objects to the internet via a variety of technologies, including cellular technology (2G/3G/4G/LTE/5G), Machine-to-Machine (M2M), and radio features such as Bluetooth (IEEE 802.15.1), Wi-Fi (IEEE 702.11), and ZigBee (IEEE 802.15.4). Regardless of IoT device heterogeneity (Baccelli et al., 2014), data from IoT applications (such as home automation, smart buildings, and energy management services) can be easily combined, connected, contrasted, and integrated to achieve the set objectives (Piccialli et al., 2017).

Small batteries and energy supplies often operate tiny gadgets in the IoTs (Baker et al., 2015). Several IoT applications are designed to track critical circumstances such as fire detection, smoke detection, building health surveillance, and intrusion detection in undesired network latency or jitter cases. In such applications, the network must be stable enough, and the routing of this data must be optimized to deliver data to a designated system within a defined period. Since most nodes in a multi-hop routing scheme sleep to conserve energy, the nearest nodes of the sink should be compelled to wake up to gather and send data to the sink node without delay and without sacrificing energy efficiency (Liu et al., 2016). As a result, long-term applications like continuous surveillance systems are vital to lengthen their life (Choi et al., 2015). Moreover, the rate of data obtained by the base station is usually tremendous (Rahman et al., n.d.). As a result, data aggregation from diverse locations is an efficient approach in a network where nodes are resource and energy-constrained (Chao & Hsiao, 2014) (Accettura et al., 2013) . The query aggregation strategy's primary goal is to efficiently aggregate and gather data packets to control energy consumption, prolong the network lifetime, traffic bottlenecks, and data correctness (A. R. Khan & Chishti, 2020), (Prakash et al., 2006). Furthermore, removing redundancies and reducing the volume of transmitted data would save network resources (Dhand & Tyagi, 2016). The effectiveness of aggregation depends on the structure of the network and the sensing data size. Since the scale of the sensing data is so large, it is vital to reduce the network's top communication (F. Xie & Ye, 2015).

In addition, the flooder motes can attack the aggregator nodes during the query aggregation process (Merad Boudia et al., 2015) Thus, the base station cannot guarantee the accuracy of aggregated data if the intermediate node is down. (Parmar & Jinwala, 2016).

2.3 Query Processing in WSNs/IoT

This section addresses the relevance of WSNs, and IoT-based queries and the specific features and challenges related to WSN and IoT query processing. WSN seeks to provide users with access to relevant information derived from data gathered by various sensor nodes. In real-world applications, a significant number of sensors are employed to track any physical environment (Meguerdichian et al., 2001). Such networks, therefore, produce huge amounts of data. However, query processing in WSN is needed to extract relevant information from the massive amount of sensor data. The primary reason for generating queries in WSN / IoT is as follows:

- Query processing can reveal complex patterns in unstructured sensor data, allowing the information of interest to be identified.
- End users can communicate with sensors by querying the sensor network without worrying about the complexities of networking.
- A query may be scheduled flexibly to collect data on demand or at defined intervals, depending on the application requirement.
- An event is a pattern or notable change that occasionally appears in the observed environment in different forms (P. Wan & Lemmon, 2009) like continuous or gradual. Query processing in WSN helps to detect any unusual event in the environment.

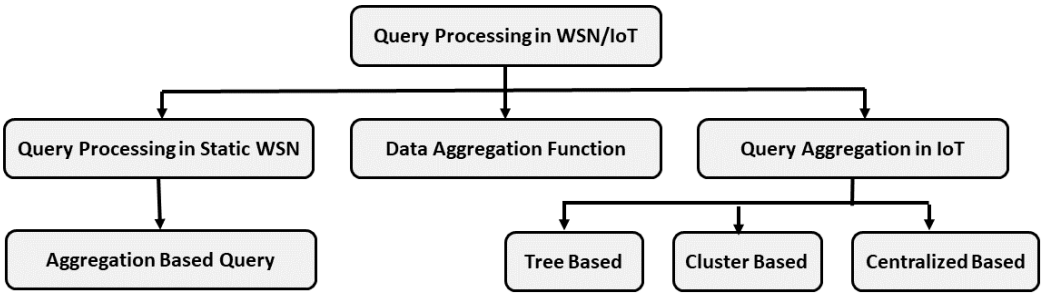


Figure 2.4: Query processing in WSN/ IoT

The subsequent section elaborates a summary of challenges stated in existing state-of-the-art studies for query processing tasks in single and distributed WSNs. Figure 2.4 depicts the general outline of the discussion, which will be expanded on one by one in the following sections.

The subsequent section elaborates a summary of challenges stated in existing state-of-the-art studies for query processing tasks in single and distributed WSNs. Figure 2.4 depicts the general outline of the discussion, which will be expanded on one by one in the following Sections.

2.3.1 Query Processing in Static WSN

As mentioned previously, WSNs are composed of sensor, fusion and sink nodes which forming a three-tier architecture. Where sensor, fusion and sink nodes are responsible for forming the bottom level, middle level, and top-level network hierarchy. In query based WSN, regardless of query type, high-level user queries are received at the sink, updated, and directed towards the appropriate sensors through the fusion nodes. As shown in Figure 2.5, sensor nodes process queries and return query results to sinks in the reverse hierarchy. In network query processing is a query processing model that pushes query into the sensor network, closer to the data source (Noury et al., 2000)**.** WSN queries can be planned to execute in various ways, such as an aggregation-based query, depending on the query execution strategy.

2.3.1.1 Aggregation Based Query

As previously mentioned, WSN queries require data from sensors that meet spatial, temporal, or value range specifications. Many sensor nodes are highly resourced limited and need to maintain as little energy consumption when communicating data and processing queries.

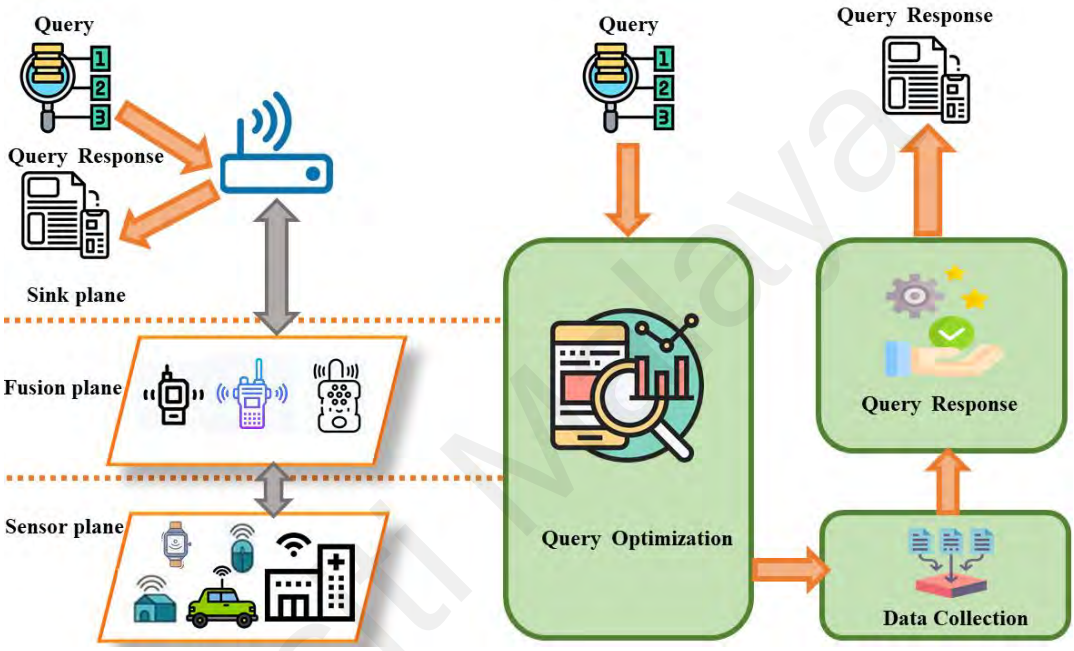


Figure 2.5: Query Aggregation in WSN/IoT Network

In most sensors, data transmission consumes the more power compared to data processing. Therefore, it is very desirable to reduce the amount of data through local processing while providing query response. The user entity would be linked to the root node, which would issue queries, and the sensors would cooperate to produce an accurate result. According to Fasolo et al. (2007) in-network aggregation of query response data can be divided into two approaches based on various factors such as application type, available bandwidth, network parameters, etc.

- **In-network Aggregation with Size Reduction:** This method incorporates data from

various sources to minimize the amount of data be transmitted. Assume that a sensor gets multiple temperature measurements from two different sensors. It may aggregate data according to application requirements, such as average or maximum temperature, instead of forwarding two readings and thereby minimize the amount of transmitted data by sacrificing the accuracy of data. It is typically not possible to restore all the original data ideally after the aggregation process.

- **In-network Aggregation without Size Reduction:** Instead of more processing, this approach lowers the quantity of transferred data by merging several data packets into one. A sensor, for example, collects pressure and temperature data from two separate sensors. Since the data types vary, it is impractical to merge them into a single data set, such as an aggregate or maximum value. As a result, it may combine all data packets into one, reducing the volume of data transmitted. At the sink, this method preserves the original data.

2.3.2 Query and Data Aggregation in IoT

The core objective of the query aggregation is to enhance the network lifetime and also reduce the energy consumption (Tripathi, A., Gupta, S., Chourasiya, 2014). However, any node may store, aggregate, and send aggregated queries, received from subsequent nodes or produced by itself over a certain length of time. (Upadhyayula & Gupta, 2007). This eliminates redundant and unwanted queries from raw data and reduces communications expenses (Z. Li et al., 2017) . *Here, more than one query is considered as data in the network.*

Nodes in the IoTs usually are resource-constrained and battery limited (Raza et al., 2017) . To save energy and resources, data need to be aggregated (P. Zhang et al., 2018). The data aggregation is a process by which specific nodes or simply a single node combined the results of other nodes (Goudarzi et al., 2019), (Nguyen et al., 2021). The

node handles the collected data to reduce transmission either with the base station or with an outside user having the authorization to connect to the network (Pourghebleh & Navimipour, 2017), (Sanyal & Zhang, 2018), (Kiran Maraiya, Kamal Kant, n.d.). Figure. 2.6 depicts the aggregation of data through a process in the IoT. By selecting the most appropriate route, the collected data is transmitted to the sink (Dagar, M., Mahajan, 2013). In general, data aggregation approaches have the following several benefits:

- It aids in improving the quality and accuracy of data, which is accomplished across the entire network (Mishra, 2012).
- Since the data collected from nodes contains specific redundancy, this procedure is needed to reduce unnecessary data which can also decreases traffic load and conserves the resources of the nodes. (Mishra, 2012) .

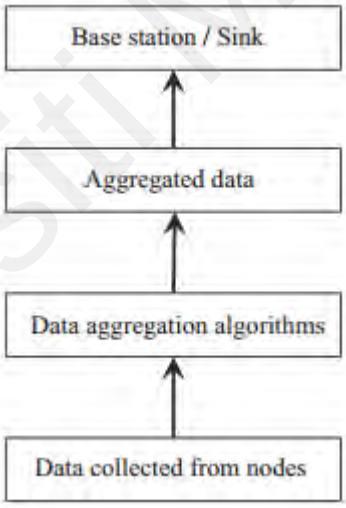


Figure 2.6: Data Aggregation Mechanism

This Section covers the most critical existing data aggregation techniques, along with their variations, advantages, and drawbacks. This Section elaborates the three categories of aggregation mechanisms for IoT that are based on tree, cluster, and centralized aggregation frameworks.

(a) Tree-based

All nodes in the tree-based method are supposed to be deployed in the shape of a tree, implying that data aggregation will be performed by a hierarchical and intermediate node (Dagar, M., Mahajan, 2013). Figure 2.7 depicts the data aggregation mechanism in this technique. Each node needs a parent node in transferring the sensed data. The data communication process initiates at the leaf nodes and terminates at the sink and is performed by the parent nodes via the aggregation process (Mishra, 2012). The main feature of the tree method is to offer energy efficiency (Dagar, M., Mahajan, 2013). The following subsections describe the chosen tree-based mechanisms.

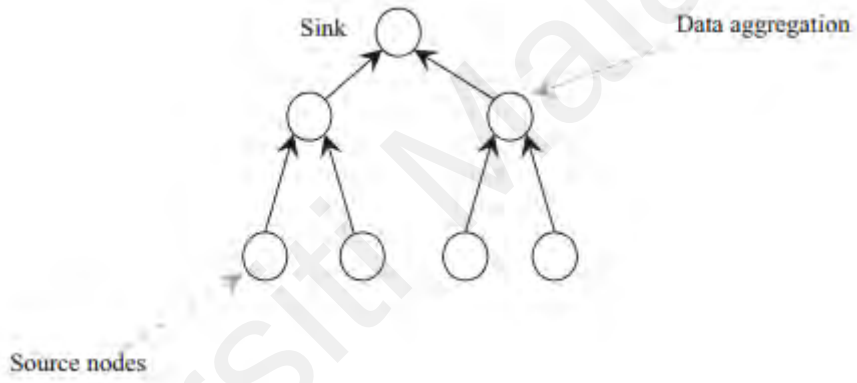


Figure 2.7: Tree-based Mechanism

Search technique for IoT services via hierarchical nodes has developed by (Fredj et al., 2013) which comprising intelligent spaces in which IoT systems reveal various capabilities.. The tree-based structure is a kind of practical illustration. The search technique employs clustering and data aggregation employing a quasi-metric. This approach identifies all the services, which reply to a request and are fit for matching costs, compared to other methods. Furthermore, depending on the request's description, it imposes a limit on the number of nodes to which it can be transmitted. Consequently, all nodes have a significantly lower probability of flooding. However, this mechanism does

not study the costs of retaining the discovery platform. It is resource-intensive in a variable scenario. In addition Z. Zhou et al. (2014) proposed a tree index approach which is highly efficient while forming sensor nodes to a tree architecture which is skewed in their distribution of sensor nodes.

The method's key contributions are integrating proximate sub-regions having similar message forwarding distances between two sub-regions. In comparison to current comparable methods, this technique reduces the amount of dead space in upper-level sub-regions. Therefore, queries consume less energy than the existing index tree-based frameworks.

According to experimental results, this process is more energy efficient. It takes no account of the heterogeneity of the nodes. A lifetime balanced data aggregation technique is designed by (Z. Li et al., 2017) for the IoT network. Which is based on end-to-end delay requirements regulated by specified application. The proposed approach extends the life of an IoT network and reduces end-to-end network restrictions while preserving the delay in required data delivery. To balance the lifetime of adjacent devices with no raising the end-to-end delay, the aggregation delays of adjacent devices are balanced together collaboratively. Furthermore, aggregation delays are only adjusted locally across neighbors. Consequently, dealing with network heterogeneity and complexities quickly is regarded as one of its capabilities. Aggregation delays are adjusted as soon as the lifespan between neighboring devices gets imbalanced because of connectivity. It can control aggregation behavior, for instance, packet loss, route changing, and so on, dynamically. Thus, it is practical in realistic situations. Finally, the heterogeneous networks will make a compromise within network life and the end-to-end latency. In this mechanism, the precision of the results is also enhanced by the problem of battery leakage. However, the effects of working with multiple sinks are not improved.

In addition, Hitchhiker a feature linking model is designed by (Ramachandran et al., 2016) to assist multifunctional data aggregation and management inside IoT. The bindings are graded as a high or low priority in this mechanism. In this mechanism, metadata is used to provide multi-hop data aggregation through component bindings. To facilitate low priority end-to-end routing queries, it can adopt a central meta manager to discover routing query requests in multi-hop networks. This method also offers certain advantages, such as lower energy usage, latency, and the capacity to tolerate data loss. However, it don't address the accuracy and node heterogeneity issues. Sruthi & Geethakumari, (2016) proposed a method for IoT data aggregation that is both efficient and secure. The computational and communication shortcomings of the IoT network are addressed in this mechanism, while security features are linked to the creation of a perfectly secure data aggregation system. This system thus ensures the safety and is subject to heavy traffic loads. Zimos et al. (2016) proposed data aggregation technique, which is suitable for the vast implementation of IoT devices for air quality monitoring. The proposed mechanism reduced mean-squared error significantly while recovering the data against distributed and other compressed sensing approaches employing experimental findings.

The proposed method reduces required network traffic, data rates, and device life expectancy. The proposed design equally demonstrates tolerance in case of noise occurring in measurement and communication. However, extreme calculation does not incorporate sensor nodes. On the other hand, latency is not considered. Furthermore, Koike et al. (2016) developed a data combination approach, as well as the requirements and application for deploying it over a wide-area network to assist IoT traffic. From an architectural point of view, this method produces overlay networks that lower the load of

data transmission in the router and builds a logical network based on information sent over the Internet.

This method merely aggregates packets and makes no changes to the data contained within them. As a result, this technique is reversible and safeguards the payload's information. The suggested aggregation method decreases the load on the underlying network and energy utilization. It, on the other hand, has a high latency. Finally, (Alghamdi et al., 2016) Proposed a secure approach for aggregating IoT data. An elliptic-curve-based seed exchange technique and a Hilbert-curve-based data transformation are used to encrypt sensor data in the given mechanism. As a result, determining the flooder nodes is quite challenging.

According to the performance review, this technique surpasses existing approaches in terms of privacy and energy savings. A tree-based design was adopted to carry out intermediate aggregations. The studies suggest that the data protection, performance, precision, and data integrity requirements should meet the desired method of aggregation. This measurement is used to detect the space and conduct an algorithm tradeoff. Compared with current methods, the life of the network and the aggregated rate of data participation in this process have increased but are subject to high traffic.

The selected tree-based study is discussed in this section. These mechanisms have solved energy and network existence problems, but latency, data consistency, QoS, and problems with redundancy in the future should be considered. The key advantages and disadvantages of each of the studies are compared in Table 2.1.

Table 2.1: A Comparison of Tree-Based Mechanism in Data Aggregation

Studies	Method	Strength	Limitation
(Fredj et al., 2013)	A secure data aggregation method	Highly scalable	Low energy consumption

(Z. Zhou et al., 2014)	Energy-efficient index tree	Minimal consumption of energy	Lack of assortment
(Ramachandran et al., 2016)	Component binding model	Minimal consumption of energy Minimum delay Supported fault tolerant	Accuracy and heterogeneity are not considered
(Sruthi & Geethakumari, 2016)	An efficient and secure data aggregation method	Highly secure	Generate more traffic
(Alghamdi et al., 2016)	A secure data aggregation scheme	Stable lifetime of network Minimal consumption of energy Highly accurate High secure	Generate more traffic
(Koike et al., 2016)	Packet aggregation scheme	Generate minimum traffic Minimal consumption of energy	Produce more delay
(Zimos et al., 2016)	An efficient and secure data aggregation method	Stable lifetime of network Highly scalable Generate minimum traffic	Latency is not considered
(Z. Li et al., 2017)	Lifetime balanced data aggregation method	Stable lifetime of network Highly scalable Highly accurate Support heterogeneity Minimal consumption of energy Minimum delay	Don't support more than one sinks

(b) Cluster-based

This Section addresses cluster-based procedures of data aggregation in the IoT and selected studies for cluster-based mechanisms. The methods described are finally compared and discussed.

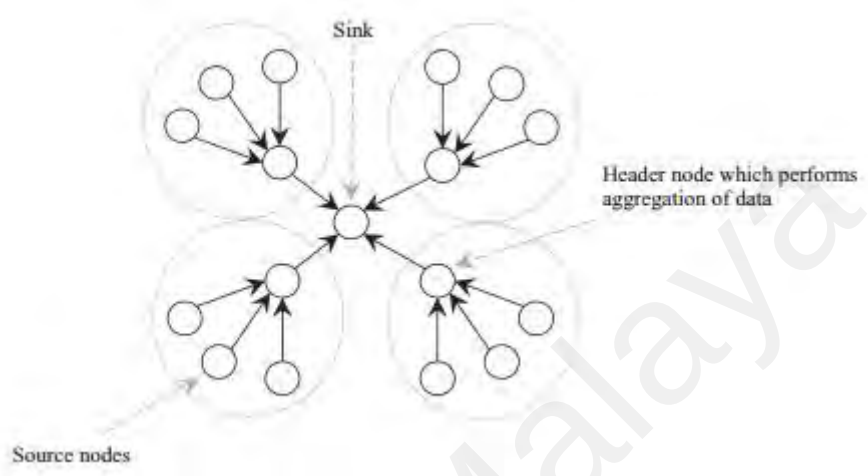


Figure 2.8: Cluster Based Mechanism

The cluster-based method splits the network into several clusters. Every cluster is made up of several sensor nodes. Every cluster keeps a header node, recognized as a cluster-head. Additionally, bandwidth overhead can be reduced with the number of transmitted query packets (Sirsikar & Anavatti, 2015). Figure. 2.8 depicts the data aggregation mechanism in this technique.

For data aggregation in the IoT the authors in (Liu et al., 2014) proposed trust analysis tool based on a node behavior detection. A trust record queue is used to record node trust records and malicious detection, which portrays the kind of trust evaluation. This mechanism lowers communication costs between storage and nodes and ensures failure tolerance, although it does not take account of the heterogeneity of the nodes. In addition, Jiang et al. (2015) have proposed a stable and scalable IoT storage system that

meets data mining and analytics requirements with extensive aggregate data in context of flexibility, stability, and reliability.

The framework is built around a revamped secret sharing scheme that ensures data protection without the need for complicated key management. At the device level, a distributed IoT storage foundation is organized to provide reliability and scalability. The multiple IoT storage servers' cloud be combined to increase capacity or to isolate for greater flexibility. The proposed mechanism tolerates packet loss, but it has limitations due to the dispatcher's low availability and heavy traffic load.

F. Xie (2014) suggested a Chinese Remainder Theorem (CRT) employing an IoT data aggregation coding algorithm. This method is appropriate for data sensing. The benefit of the CRT transform is that it compresses large sensing data into several residues. The aggregator node will receive all sensing data in the proposed mechanism, conduct aggregation, and transfer the results to the decision server. Also, the proposed mechanism transmits sensing data to residual data. The mechanism decreases traffic load and increases the reliability of data transmission while the final aggregation results are of low accuracy. In addition, González-Manzano et al. (2016) have offered an IoT-friendly aggregation method that provides the possibility of multi-attribute accumulation clusters allowing value correlations that protect privacy.

This mechanism allows data to be aggregated in one procedure concerning multiple attributes of each entity to ensure validity and privacy of data. Furthermore, the proposed system can cope with broad situations that allow malicious handling of aggregated data to be detected. Privacy protection, collision resistance, verifiable aggregation, and correlative aggregation are among the aims of the proposed mechanism. This technique provides an associated combination that allows the sink to achieve both the total amount and the association between the values of the attribute. An incorrect

discovery procedure is also given for aggregators to keep away from redundant aggregators. This approach is suitable for use in a broad IoT environment with a core server and several nodes that work in an application based on the assessment results. The primary downside of this scheme is that it has higher latency.

Finally, a cross-layer design for data aggregation is proposed by (Alkhamisi et al., 2016). This technique functions without a static infrastructure for mobile ad-hoc environments. This technique is an interoperable method for both application and network layers and is favorable in managing fault and failure tolerance. This technique has the advantage of minimizing traffic load and conserve energy consumption in real-time.

The previous section analyzed some selected cluster-based studies. In addition, each study included a description of the process, advantages, and weaknesses. The most important advantages and disadvantages in each study are comparatively described in Table 2.2.

Table 2.2: A Comparison of Cluster-Based Mechanism in Data Aggregation.

Studies	Method	Strength	Limitations
(F. Xie, 2014)	Chinese remainder theorem-based algorithm	Generate less traffic	Lack of accuracy
(Liu et al., 2014)	A novel trust-based secure data aggregation	Highly accurate Highly secure Support fault tolerance	Lack of heterogeneity
(Jiang et al., 2015)	Secure and scalable IoT storage system data aggregation	Highly scalable Highly Secure Support fault tolerance	Generate less traffic
(Alkhamisi et al., 2016)	Cross-layer Framework	Generate less traffic Minimal consumption of	Latency is High

		energy, Support fault tolerance	
(González-Manzano et al., 2016)	Privacy maintaining aggregation protocol	Highly scalable Highly Secure Support fault tolerance	Latency is high

(c) Centralized Mechanism

This mechanism permits each node to transmit data to a central node by adopting a shortest possible route. All nodes transmit their triggered query packets to a powerful node that connects all the other nodes. This node refers as header node.

The header node combines data from all other nodes, and the aggregation process results in a single packet (Sirsikar & Anavatti, 2015). Figure 2.9 depicts the centralized data aggregation mechanism.

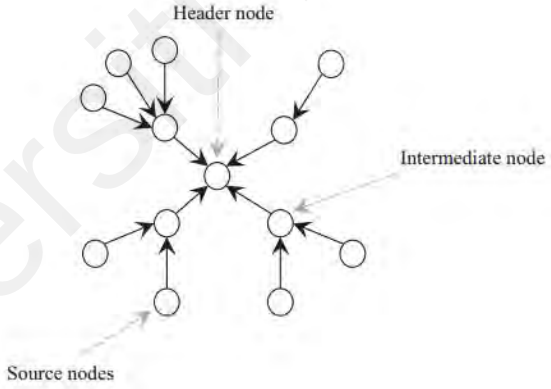


Figure 2.9: Centralized Mechanism

Sándor et al. (2015) proposed an IoT platform architecture for data aggregation that integrates WSN and standard ICT infrastructures with publish-subscribe data distribution capabilities. It has also provided a comprehensive risk analysis that considers reliability, accessibility, and privacy. The experimental impact of a case study is assessed

at the laboratory scale of IoT-based applications. On the other hand, the sophistication of data aggregator systems reveals various flooding threats to these infrastructures.

Zhu et al. (2017) proposed a service-oriented distributed architecture to gather information from multiple data nodes common to several IoT applications. Each manufacturer provides services for its products. In this architecture, the information collected is stored in the data node by itself. This method is scalable since every query node is responsible for its own product and each data node is responsible for only the data it collects. Requests for query/registers are answered as addresses of linked nodes can be settled with the product URIs directly. The mechanism's assortment issues can be resolved with semantic and service-oriented technologies. This mechanism would minimize network traffic and will serve as a versatile mechanism for data sharing through various applications. However, it has the issue of single-point failure.

Wan et al. (2019) designed a multidimensional data indexing scheme that is both energy and time-efficient and is structured to respond to range queries. Specifically, to have a more effective routing at lower latency, the proposed approaches used for data indexing are utilizing hierarchical indexing structures using Binary Space Partitioning (BSP), such as K-means clustering and Voronoi-based methods. The algorithm of the Voronoi Diagram (VD) reduces the total energy consumption and the response time for query requests. This work is restricted because the VD data indexing model can only be used for general query operations in O (log n) time cells, e.g., for processing the location-based service. The proposed work lacks real-world prototype for evaluating its usefulness.

Currently, the IoT-based WSN is an immense ongoing research area due to multiple applications and services in the numerous fields. In this regard, a large amount of data can be sensed by the sensor nodes, and some of the data are redundant. This

redundancy degrades the performance of the network by creating some demerits. To overcome this issue, (Kumar & Chaurasiya, 2019) presents a duplication elimination technique to eliminate duplicate/overlap queries. It performs data mining to pick the relevant details until forwarding data to a base station or a cluster head. This method removes redundancy from data query packets sensed by nodes before sending them to the cluster head or base station for pre-processing. In the IoT, the network of sensor nodes operates in two ways: First, the data are sensed and collected spatially by its neighboring nodes. Second, the data are sensed and observed temporally at the given intervals of time. The proposed approach performs better in both cases. Specific characteristics and properties of the sensed data must be present to avoid redundancy. The proposed solution enables load balancing, traffic accounting, and data management while providing Quality-of-Service (QoS) for various applications and services. The main drawback of this approach is that it doesn't support dynamic scenarios and is only limited to cover few QoS metrics. In (Dietzel et al., 2016) the authors have developed a resilient aggregation framework that uses existing communication redundancy to recognize and filter fake aggregates with the help of data consistency tests. In this model, an attacker cannot monitor all these paths but can still affect most incoming paths to a destination based on network topology. As a result, the filtering mechanism's architecture restricts information distribution to node-disjoint paths, potentially reducing the impact of flooder nodes. It employs a filtering mechanism with clustering to identify and delete conflicting information. It is limited to single-point failure, covers few QoS parameters. The most significant advantages and disadvantages of each study are presented in Table 2.3.

Table 2.4 summarizes the mentioned data aggregation techniques and their essential features, including energy consumption, fault tolerance, accuracy, latency,

heterogeneity, network longevity, scalability, protection, and traffic load, as well as the impact of these factors either be beneficial or detrimental.

Table 2.3: Comparison of Centralized Mechanism in Data Aggregation.

Studies	Method	Strength	Limitation
(Sándor et al., 2015)	Security assessment of modern data aggregation platforms	Highly secure	Lack of fault tolerance
(Dietzel et al., 2016)	Filtering mechanism	Scalable	Support few QoS, Support only single point failure
(T. Zhu et al., 2017)	Distributed service-oriented architecture	Highly secure Support heterogeneity Generate less traffic	Lack of fault tolerance
(Kumar & Chaurasiya, 2019)	Duplication elimination technique	Load balancing, traffic accounting, and data management	Don't support dynamic scenarios and is only limited to cover few QoS metrics
(S. Wan et al., 2019)	Multidimensional data indexing scheme	Energy and time efficient, low latency	Does not have a real-world prototype

Table 2.4: Overview of data aggregation Methods along with its features.

		QoS Parameters										
Type	**Studies**	**Heterogeneity**	**Traffic load**	**Energy consumption**	**Scalability**	**Security**	**Cross-layered**	**Network lifetime**	**Redundant Query**	**Latency**	**Fault tolerant**	**Accuracy**
Tree	(Alghamdi et al., 2016)	N	N	Y	N	Y	N	Y	N	N	N	Y
	(Sruthi & Geethakumari, 2016)	N	N	N	N	Y	N	N	N	N	N	N

	(Ramachandran et al., 2016)	N	N	Y	N	N	N	N	N	Y	Y	N
	(Koike et al., 2016)	N	Y	Y	N	N	N	N	N	N	N	N
	(Zimos et al., 2016)	N	Y	N	Y	N	N	Y	N	N	N	N
	(Z. Li et al., 2017)	Y	Y	Y	Y	N	N	Y	N	Y	N	Y
	(Z. Zhou et al., 2014)	N	N	Y	N	N	N	N	N	N	N	N
	(Fredj et al., 2013)	N	N	N	Y	N	N	N	N	N	N	Y
Centralized	(T. Zhu et al., 2017)	Y	Y	N	N	Y	N	N	N	N	N	N
	(Sándor et al., 2015)	N	N	N	N	Y	N	N	N	N	N	N
Clustered	(Alkhamisi et al., 2016)	N	Y	Y	N	N	N	N	N	N	Y	N
	(González-Manzano et al., 2016)	N	N	N	Y	Y	N	N	N	N	Y	N
	(F. Xie, 2014)	N	Y	N	N	N	N	N	N	N	N	N
	(Jiang et al., 2015)	N	N	N	Y	Y	N	N	N	N	Y	N
	(Liu et al., 2014)	N	N	N	N	Y	N	N	N	N	Y	Y

Researchers focused on QoS metrics such as energy consumption, tolerance for faults/failures, security, scalability, and network traffic in the selected studies. However, many data aggregation techniques fail to consider accuracy, latency, network life, the cross-layered, and redundant query issues.

Query transmission is typically believed to make up a small portion of total data transmission in an IoT sensor network. However, in certain instances, the presumption

does not hold, which requires aggregation of queries before they are spread to the network.

2.3.3 Aggregation Functions

Query transmission is typically believed to make up a small portion of total data transmission in an IoT sensor network. However, in certain instances, the presumption does not hold, which requires queries aggregation function before they are spread to the network. The aggregated functions (Polyvyanyy et al., 2017), (Sun et al., 2017), (Huysmans et al., 2008), (Shafagh et al., 2015) of the different applications are closely linked to the specific sensor. They can be categorized as follows:

- **Lossy and Lossless Aggregation:** By using a lossy aggregation function, the fundamental values are lost after combining. Furthermore, as compared to sending all readings uncompressed, the transmitted data can lose precision. On the other hand, the lossless method enables compression by retaining the original data, allowing the entire readings to be thoroughly reproduced from aggregated form.
- **Duplicate-Sensitive and Insensitive**: A sensor node can be needed to accumulate multiple copies of the same data. It should determine whether to discard duplicate data depending on the feature type or not. For instances, if an aggregate function calculates the mean value of received data, duplicate values will affect the result. Conversely, duplicate-insensitive only considers the lowest value among the data. Furthermore, various devices may be better suited to different types of operations (Lu et al., 2008), which must be accomplished when designing aggregation functions.

2.4 Querying Distributed WSNs towards the IoT

As previously stated, integrating various enabling technologies such as recognition, sensing, and communication technologies, as well as retrieving relevant information from them, is needed to bring the IoT concept to life in the real world. The core objective of this section is to deliver an overview related to distributed WSNs in IoT (Alam et al., 2010; Gračanin et al., 2006; W. Li et al., 2011; Mainetti et al., 2011; Sánchez López et al., 2012), what are the difficulties and strategies in managing massive and heterogeneous sensor data in the IoT, and what are the methodologies and processing blocks of performing the query on that sensor data (Figure 2.10).

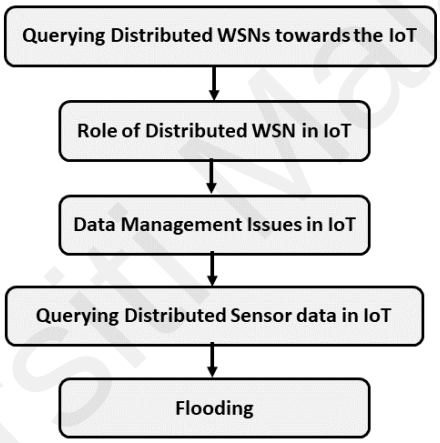


Figure 2.10: Issues Regarding Distributed WSNs towards IoT

2.4.1 Role of Distributed WSNs in IoT

Since the IoT should be able to link a large number of disparate objects through the Internet, there is a strong reliance on a dynamic layered architecture (Ikram et al., 2015; Johnson et al., 2017; C. Zhang et al., 2017). The IoT architecture must ensure its operations, which connects the virtual and physical worlds (Mustafee Navonil, 2015)(Su et al., 2015). The IoTs links a massive number of devices, resulting in significantly increased traffic and even more data storage requirements. As a result, a new IoT architecture is needed to resolve several issues such as scalability, QoS, interoperability,

and reliability etc. Also, technological growth, different types of new businesses, and application models determine IoT progress (Tsai et al., 2014) .

The traditional IoT layers are shown in Figure. 2.11 (Khan et al., 2012). The following is a depiction of these layers:

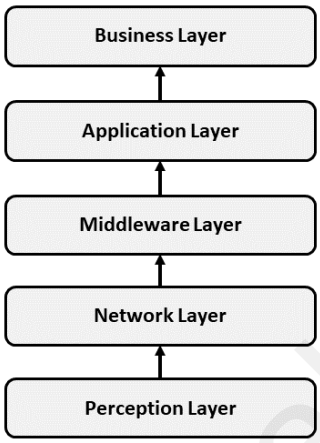


Figure 2.11: Layers of IoT

- **Sensor/Perception layer:** According to (Al-Fuqaha et al., 2015) the Object or edge technology layer refers to the perception/Sensor layer or sensing layer. It's the bottom layer, often known as the hardware or physical layer. This layer manages data gathering with the help of tiny connected sensors (Liao & Hsiao, 2014). This layer converts information into signals to transmit over networks for related applications (J. Tan & Koo, 2014).
- **Network Layer:** The network layer aims to link everything and allow anything to share data (Atzori et al., 2012; S. Li et al., 2015). This layer secures data transmission from source nodes to the core unit (R. Khan et al., 2012). WLAN, Bluetooth, Zigbee, 3G, UMB, infrared technology, and so on are the primary technologies used in this layer (Wu et al., 2010).
- **Middleware Layer:** This layer integrates service and request according to address and name. It enables IoT application developers to work with

heterogeneous objects without focusing on a specific platform (Al-Fuqaha et al., 2015). The information is retrieved from the network and saved in the database by using this layer (R. Khan et al., 2012).

- **Application Layer:** Typically, information from other layers is integrated and evaluated in this layer (Tsai et al., 2014). This layer provides high-quality services to meet the needs of customers queries (Al-Fuqaha et al., 2015).
- **Business Layer:** The business layer can handle the whole activity and service by developing flowcharts, business models, and related designs (Al-Fuqaha et al., 2015). The achievement of IoT technology is determined by the value of applied technology, creativity, and the business model's viability. (Wu et al., 2010).

The following Sections go through the existing studies on querying distributed sensor data in the IoTs.

2.4.2 Querying Distributed Sensor Data in IoT

As previously mentioned, the IoT includes a massive amount of sensor data from distributed and heterogeneous WSNs. Existing research looks at different methods for querying sensor data. Some approaches store sensor data in the upper layer of the IoT hierarchy and pose queries against it. In contrast, others send the query directly to the cloud or base station, which collects the appropriate sensor data. The following section addresses several extensive works on priority and flooding queries.

2.4.2.1 Priority Queries

Recently, many application scenarios like military surveillance, infrastructure protection and environmental monitoring have been accomplished by IoT.

In some cases, applications in WSN/IoT are time restricted, leading to the need

of real-time scheduling of data packets. Nowadays, many communication package schedule policies in WSN/IoT apply algorithms in view of First-Come-First-Served (FCFS). However, there are many problems in FCFS; including node-to-node communication packages transport delay as well as starvation in real-time packages. Besides, these schemes cannot respond dynamic inputs quickly (Wang & Zhang, 2016). In data centers, the occurrence of timeout for priority queries dramatically causing problems like query completion time. To mitigate timeouts, the transport protocol should try to maintain a small switch queue to avoid the packet loss and recover lost packets quickly. Recent work suggests using Explicit Congestion Notification (ECN), Round Trip Time (RTT) or the in-network signal to achieve that. However, these solutions either still suffer from many timeouts when the number of concurrent flows becomes larger or require the nontrivial hardware support (Ruan et al., 2020)

The authors in Rajendranath & Hency (2019) proposed PRITRAPS (Priority-based Task aware Pre-processing and Scheduling) mechanism that is employed in real time scenarios of industries. In which different applications units are accessing the gateway unit to measure and monitor the parameters of different service types. PRITRAPS employs priority among the tasks to reduce the network load.

The scheduling algorithm employed in PRITRAPS is EDF (Earliest Deadline First). The pre-processing task unit decreases the number of tasks by choosing the tasks having similar spatial and temporal requirements. The residual energy of the sensor nodes can help the scheduler for deciding the sensor nodes in respective of query task requirements. The scheduler finds the best potential nodes and

assigns them to the query task for processing. This method is not energy efficient and lacks Qos parameters.

The authors in VasudevanVijay et al. (2009) suggested reducing RTOmin to the granularity of microseconds. However, this method requires the specific kernel version (2.6.18 or later) and may cause the massive spurious retransmission (J. Zhang et al., 2013). Other protocols such as SAB and TFC control the queue length using the customized switch. The basic idea of SAB is to uniformly allocate the switch buffer to each flow. The allocation result is fed back to each flow as its congestion window. In this way, the injected traffic of all flows cannot exceed the switch buffer. One target of TFC is to achieve zero-queueing. It represents the link bandwidth resource as tokens and allocates them to flows some time. TFC guarantees that the aggregated injected traffic would not exceed the capacity of the pipeline. In these two protocols, packets are rarely lost and the probability of timeout is greatly reduced (J. Zhang et al., 2014)(J. Zhang et al., 2016).

2.4.2.2 Flooding Queries

Flooding queries is usually used for route discovery, route maintenance and topology update in IoTs. In large-scale WSNs and IoT, this flooding causes such excessive message collisions that the network efficiency is reduced. However, the flooding queries has obvious advantages over the location-based unicast/multicast in complexity and economic cost without additional equipment.

Three critical mechanisms can be used to scan for nodes with emergency data: flooding, managed flooding, and random walk (Cui, 2009), (Benenson et al., 2006), (Benenson et al., 2006). The random walk has a lot of latency, so it's not suitable for time-sensitive applications. The simple flooding mechanism overloads the network with

redundant query packets, generating heavy load and rapid energy depletion.

As a result, the authors in (Qiu et al., 2012) present Level-Based Flooding as a new search mechanism (LBF). It can calculate the shortest path between the sink and target nodes. The nodes in LBF are organized into a hierarchy, and the sink node is aware of the node levels. When a sink node receives a query, it may broadcast the query cost-effectively based on the nodes' level information.

In Yuan et al. (2013), the author proposed a level and cluster-based routing approach for a wireless sensor network. This approach divides the whole network into different levels as per hops to the central node. Every node must have a level number. A sensor node forwards information to a sink node more efficiently using level information, and a sink node can quickly locate other sensor nodes which can helps to balance the network. Unlike all other cluster routing methods, instead of sensing, a cluster head schedules jobs for sensor nodes in the cluster based on their remaining resources. This study also presents several algorithms for developing, querying, and scheduling a wireless sensor network. The proposed research considered factors such as energy consumption, interoperability, and rapid response. The shortcoming of this study is observed that it only concentrates on energy usage and was not beneficial in mobility scenarios. Do not accept cross-layered solutions and are only limited to network layer systems.

The study in Cheng et al. (2018) mainly focused on minimizing the delay and energy consumption while constructing the flooding tree taking into account duty-cycle activity and unstable wireless links. It demonstrates the presence of delay and energy consumption during flooding. The problem is formulated as an undetermined-delay-constrained minimum spanning tree (UDC-MST) problem with an a posteriori known delay constraint. Since the UDC-MST problem is NP-complete, a distributed Minimum-Delay Energy-efficient flooding Tree (MDET) algorithm is designed to create

an optimal energy tree with flooding delay bounding. This research achieves a potent combination of flooding delay and energy efficiency.

Baddeley et al. (2019) introduced a novel synchronous flooding (SF). SF protocols dynamically meets specific SDN control criteria. By using Atomic-SDN, it offers substantial performance improvements over other low-power IoT networks while implementing in SDN. SF methods are capable of providing latency and reliable assurance to SDN controllers at the local mesh scales. Atomic-SDN increases SDN control by orders of magnitude in terms of reliability, latency, and energy consumption. In addition to this analysis, the spread of control messages across the flood network can be facilitated with unique control schedules. The Atomic-SDN enables the SDN layer to function without knowing the topology information and benefits from the spatial and temporal diversity of flood protocols.

The authors in Yamazaki et al. (2020) have proposed a simple route request (RREQ) flooding system based on the rest of the node's power without using control messages and complicated tasks. Initially, the limit of node density shows the significance of proposed plan in context of energy efficiency (bits/J), considering energy utilization and throughput. Further, since the proposed system assumes the flood times as constant, all the nodes would almost carry same time to drain the battery. Consequently, if the nodes are static, they last longer than in the traditional approach of proposed method. The drawback of this work is that it is not well-performing in complex and uncertain scenarios. It is only based on a few QoS parameters and was not favorable in dynamic environment. Do not endorse a solution for the cross-layered but only for the network layer.

The authors in (Abdulridha et al., 2019) developed FSFS approach that can be classified under Flat approaches; it is simple and very quick in delivering the packet to the sink and can be used for routing and propagation issues. Depending on a simplistic system within WSN that provides all nodes of a simple routing table, FSFS takes improvement over Flooding, Gossiping, and Floosping. It presents a sweet way to load balance in power consumption among the sensors with the least number of sending operations. The conclusion that should be mentioned here is that, as far as the number of nodes in WSN grows larger, FSFS gives high achievement. On the other hand, FSFS is running only with structured Topology (Mesh) which means each sensor node has a predetermined place and recognizes its neighbors. It is suitable for the monitoring applications in a civil or military environment; on the other hand, FSFS still suffers from redundant information at the base station. Table 2.5 summarizes different flooding queries mechanism fall under different categories and their essential features,

Table 2.5: Analysis of various flooding queries mechanisms

Protocol	Classification	Memory Requirement	Localization	Data Aggregation	Clustering Manner	Intra-Cluster Topology	Redundant Query
LBF	Control overhead reduction	Low	No	Yes	Proactive	Multi-hope	Yes
2L-OFFIS	Control overhead reduction	Low	Yes	Yes	Proactive	Multi-hope	Yes
ARPEES	Energy consumption mitigation	Low	No	Yes	Reactive	Single-hope	Yes
MDET	Energy consumption mitigation	Low	Yes	Yes	Reactive	Single-hope	Yes
DMSTRP	Energy consumption mitigation	Low	No	Yes	Proactive	Multi-hope	No
FSFS	Energy consumption mitigation	Low	Yes	No	Reactive	Single-hope	Yes

RREQ	Energy consumption mitigation	Medium	Yes	No	Proactive	Multi-hope	No

Memory Requirement. The memory requirements of the whole network depend on whether each node has to store some query or routing information, such as the query packets which are waiting to be forwarded, neighbor information, cluster information, route information and so on. This can be represented by a polynomial which is related to the parameter n concerning the number of the nodes. For instance, a method of event-based clustering is proposed in (Quang & Miyoshi, 2008), this method requires the nodes nearby the event store their neighbor information, we assume that the events occurs in the whole network, and thus all the nodes of the network need to store the neighbor information instead of particular nodes. With the network density enhancing caused by the increase of the network size, the nodes need to store many more information. Due to the limited memory capacity of the large-scale WSNs, however, how to efficiently utilize these storage resources is of great significance for enhancing.

Localization. Position information is of great help to enhance the accuracy and the efficiency of routing protocols. In (Jamalipour & Azim, 2016), the nodes can get the position information, and that makes the directed transmission substitute for broadcast communication of the control packet. Therefore the control overhead is decreased.

Data Aggregation. The advantage of hierarchical networks over flat networks is apparent, because in the former network data aggregation could be conducted at cluster head nodes. These nodes collect the sensed messages from its member nodes, and remove the redundant part, thus reducing the total messages towards the sink nodes. By this means, the network energy efficiency is improved.

Clustering Manner. "Proactive" means that the clustering of the network is operating before the network operates. Because the clustering is carried out in the entire network and it needs a long time to maintain, it will create more energy cost than "reactive" clustering which is triggered on demand, such as the occurrence of some event. In some emergent cases, the performance of "reactive" routing is not time-sensitive enough.

Intra-cluster Topology. In a cluster, the single hop topology can reduce the end-to-end delay to a certain degree, whereas a significant advantage of the multi-hop topology is energy-efficiency. Especially the topology of the spanning tree, which consists of the multi-hop structure, not only reduces the transmission energy through decreasing the average transmission distance, but also alleviates the collisions in clusters with a schedule scheme utilizing the tree structure.

Cluster Head Election. According to the different objectives of each protocol, these protocols have different ways of electing the cluster heads. "Residual energy" is chosen as the criteria to select cluster head to ensure that the cluster head has enough residual energy to process and deliver data packets. That makes the nodes energy-balanced to a certain degree.

Multi-Path Routing. Multi-path routing means the traffic is delivered along several paths in order to balance the energy consumption of sensors along the single path. By this method, the query packets could still be delivered successfully in the case of path failure, thus ensuring the reliable delivery of query packets. However, a deficiency is that much more overhead may be incurred owing to several sensor nodes must be selected as the next hops. In hierarchical routing protocols, some sensor nodes are grouped to efficiently relay the sensed data to the sink. The cluster-head plays the specialized role of performing

data aggregation and sending it to the sink on behalf the nodes within its cluster. Thus, how to form the cluster is a more interesting and essential research issue concerning such protocols so that the energy consumption and various communication metrics such as latency are optimized. In addition, due to the number of sensor nodes is substantially increased in large-scale WSNs, the nodes nearby the sink will assume more query forwarding tasks so the energy of these nodes is depleted rapidly. That makes the hierarchical routing protocol design challenging.

Redundant Query. Redundant queries ensures reliable data for decision making. Reliable data plays a very important role in the analysis, monitoring and forecasting of system behaviour whereas bad quality data may provide erroneous result in decision scheme. In Wireless Sensor Network (WSNs), nodes are densely deployed in a region to collect information. Sensors sense the similar data and forwards to sink. This similar data sometimes leads to redundancy at the sink. The redundant data results in more accuracy, reliability and security whereas elimination helps in energy saving as most of the energy of sink node gets waste in dealing with the redundant data. Data accuracy still needs to be preserved even if there is increase in network cost and/or time. Therefore, there is requirement of a mechanism in which we can extract information from the redundant data and be able to provide a more consistent, accurate and reliable data set in an energy efficient manner (Verma & Singh, 2018).

It can be concluded that the query flooding is usually used for route discovery, route maintenance and topology update in most of the routing protocols mentioned. In large-scale WSNs, this flooding causes such excessive message collisions that the network efficiency is reduced. However, the flooding has obvious advantages over the location-

based unicast/multicast in complexity and economic cost without additional equipment. Therefore, research on flooding technique is necessary.

2.5 QoS in WSN/IoT Query Processing

The term Quality-of-Service (QoS) relates to the reliability of a service provided by a network to the related application. A collection of observable attributes typically describes the QoS level depends on the application type (Kumar Kumar et al., 2019). This section addresses QoS-aware query support in IoT and presents significant works that have been done on QoS-enabled cross layer query execution in IoT Networks.

2.5.1 QoS Support for Query in IoT

The QoS parameters of the IoT network can be seen from different perspectives and dimensions, including bandwidth, latency, packet drop, prevent jitter and interference. Therefore, QoS should be described in a different way for diverse technologies. It is challenging to accomplish QoS effectively in wireless networks because of the segment gap caused by management and resource distribution of shared wireless media (Gubbi et al., 2013). J. M. Liang et al. (2013), proposed the 3GPP LTE-A mechanism ensuring traffic bit rate, packet delay, and loss rate with IoT devices in an energy-saving QoS context. To optimally use the LTE air interface resources, the authors in (Piri & Pinola, 2016) measured packets of various sizes in the LTE uplink. The results showed that packets of smaller size achieved approximately half the throughput of larger-sized packets. This result allows packet aggregating to optimize many QoS parameters such as latency, packet loss, jitter, and bandwidth usage required by a large number of small query packets on the mobile edge of the IoT gateway. Duan et al. (2011), developed QoS architecture, which provides a framework for the control of translation from top to bottom layer. The cross-layer management facility and brokers of this architecture often

feature lower layer controls. With the help of this architecture, researchers may further optimize the QoS of IoT. Different QoS methods for achieving QoS in IoT are described in Table 2.6

Table 2.6: Summary of Different Methods Providing QoS in IoT

Citation	Objectives	Strength	Evaluation	Limitations
(M. Zhou & Ma, 2013)	To provide QoS requirements of IoT composite services	The algorithm is fast enough to meet real-time requirements of IoT.	Testbed	Uncertainty analysis of QoS is not performed in this method.
(L. Li et al., 2014)	To present a QoS scheduling model (based on three layers) for service oriented IoT.	This method minimizes the resource costs to optimize the scheduling performance. It also presents QoS support to definite applications of IoT and enhances IoT network's lifetime.	Simulation	In case of congestion, packet loss, delay, and issues of control mechanism.
(Vithya & Vinayagasundaram, 2014)	To provide QoS routing method by establishing priority criterion in the network.	Focuses on prioritizing packets under priority queue to gain best transmission with low latency.	Testbed	The priority criteria take highest number of cluster frames, making it critical for low priority frames to wait for turn indefinitely.
(Awan et al., 2014)	To examine the QoS in context	Robust to both traffic	Simulation	This method is only an

	of delay, matching traffic generated over the network.	variations and topology failures. Provides an analytical model for evaluating the performance of smart devices under different traffic states to meet the QoS constraints. Uses buffer management, makes high priority traffic continues its arrival by impel out low priority traffic to circumvent loss of emergency related data packets.		analytical model, which is not validated in real time scenario.
(Aazam et al., 2016)	To increase QoS based on previous Quality of Experience (QoE) and Net Promoter Score (NPS) records.	Fog computing provides the solution by bringing cloud resources to the edge of the underlying IoT and other end nodes. Provides better	Simulation & Testbed	Vigorous in predicting the consumed resources by diverse devices.

		reliability and reduces jitter.		
(X. Xu et al., 2018)	To improve QoS by communication link reliability and reduce delay for loss-and-delay WSN/IoTs	Reducing delay, improving reliability, balancing energy consumption aimed at increasing the transmission power of nodes	Simulation	Only limited to optimize few QoS parameters
(X. Li et al., 2018)	To Reduce energy consumption and guarantee delay under corresponding QoS requirement constraints.	Reducing delay, improving lifetime, increasing energy efficiency without performance degradation of data transmission	Simulation	Only limited to optimize few QoS parameters
(Kyung & Kim, 2020)	The QoS-aware flexible mobility management scheme that classifies flows into four classes	Flexible network resource utilization, differential handover for different flow classes, absence of service degradation.	Simulation	Based on heuristic approach
(Shafique et al., 2020)	The SDN-Based Application-aware Distributed	Load balancing, application-aware data transmission,	Simulation	Only focus on bandwidth utilization

	adaptive Flow Iterative Reconfiguring (SADFIR) routing protocol	heterogeneity aware		
(Latif et al., 2020)	To Reduce nodal propagation delay, maximizing throughput, improving network lifetime, and minimizing energy consumption.	Reducing the number of re-transmissions, good energy conservation, enhancing throughput	Simulation	This method is only an analytical model, which is not validated in real time scenario.

In the traditional IoT query propagation paradigm, sensors transmit queries to the network's access point or central gateway motes. Table 2.6 provides a brief argument for the use of the testbed. This access point handles requests and then forwards them to the appropriate network destinations using the underlying routing system. The traditional IoT query propagation paradigm has certain drawbacks, such as the fact that sensors may send queries that are redundant or duplicate, or that a single sensor may convey an undesirable query intended for another application or sensor. This is because there are queries that overlap in multiple clusters. The overall amount of energy that is used consequently increases when the size of the query is increased. In such situations, device resources (in terms of bandwidth or energy for the sensor node) are lost due to too many redundant network query transmissions that can result in obvious degradation in QoS transmission (Fathallah et al., 2019). It is essential to notice that none of these testbeds solely focused on elimination of redundant and unwanted queries in IoT networks to enhance QoS. There

is a need of a testbed equipped with query control mechanism (QCM) that focus on mitigation of multiple and overlapping cluster queries in IoT networks.

2.5.2 QoS Enabled Cross Layer Architecture, Design in the IoTs (IoT): Issues and Possible Solutions.

IoT's layer-based architecture will hierarchically manage queries by ensuring the QoS in all layers. IoT needs to compromise between query delays and reliability based on application requirements in line with the demand for a single sensor network. The work addresses service time, service delay, service accuracy, service load, and service priority, among other QoS specifications. It proposes a QoS architecture for IoT that describes how the quality assurance criteria are transmitted and converted from top to bottom. Detailed layer based QoS architecture in IoT is depicted in Figure 2.12.

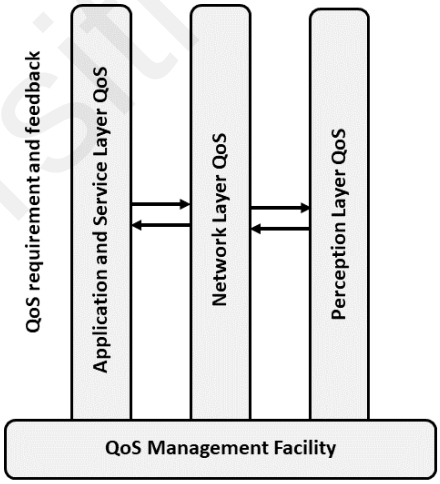


Figure 2.12. Layer Based QoS Architecture in IoT

Upper layers communicate QoS specifications to ground layers, and lower layers, in turn, share QoS feedback to higher layers. The architecture includes a QoS management facility, QoS brokers, and separate QoS requirements for each layer. For all

three layers, QoS management offers cross-layer features. The broker's job in the network and perception layer is to address QoS requests from the upper layer and translate them for use in the local layer.

More issues occur as of the increased quantity of smart devices in IoT, necessitating the use of cross-layer models to address. The cross-layer models discuss several issues related to the increased use of smart devices and applications on the IoT. The cross-layer architecture is simply coordinating different layers in integrating resources to create an extremely adaptable network (Foh et al., 2007; Papandriopoulos et al., 2008).

Cross-layer architecture is typically essential in addressing communication issues among IoTs devices (IoT). As a model, the cross-layer architecture aids in removing the precise boundaries of the OSI networking model, enabling data access from another layer. Data exchange on the IoT is employed on several layers, making it a challenge to share data across multiple layers. By using the architecture model for cross-layers, different layers share information to communicate more practically. Cross-layer communication protocols are regarded as one of the most important methods for improving interlayer communication in IoT. Some of the main issues relating to cross-layer networking protocols are energy management in wireless networks and bandwidth for better performance. The conventional layered structure of the IoT has been criticized for the major flaws discovered with the usage IoT technology has increased. Authentication, encryption, georeferencing, and timing are now possible with IoT infrastructure by adopting Trustful Space-Time Protocol (TSTP). The TSTP has been essential in removing data duplication across the system, allowing messaging to be more efficient. In contrast to the standard TCP/IP model, the TSTP cross-layer model improves the filtering process for the efficiency of IoT

data sharing.

2.5.2.1 Cross Layer Design Model

The main challenge or issue with the TCP/IP model is that it offers end-to-end connectivity for the application layer only. In this case, the application, transport, network, data link layers, and physical layer are the five major concern layers. The TCP/IP offers a small contact, as only the two neighboring layers are communicated (Ma et al., 2004)(Magagula & Chan, 2008). It is impossible to communicate between any other layer that isn't an IoT neighbor on the Internet. The communication issues are referred to as information sharing because layers, not neighbors on the IoT, cannot exchange information under the TCP/IP model. Moreover, information exchange between two layers leads to several other problems in the IoT (Luo et al., 2010). In the absence of a framework for data sharing between layers, the issue is generally obscured so that during the diagnosis, it cannot be identified. Many other issues occur in IoT in the event of noise on the network (Bandyopadhyay & Sen, 2011). It demands reconnection from one level to the next, which takes a lot of time to overcome the main problems. The cross-layer architecture is adopted to address these critical challenges

2.5.3 The Architecture of Cross Layer Platform in the IoTs

The IoT architecture is critical in resolving concerns such as Quality-of-Service (QoS), device stability, integrity, and privacy. Several different IoT architectures have been proposed. Simple layered architecture is one example. Three to five layers are possible in a simple layered architecture. The perception layer, network layer, and application layer build up a three-layered architecture. The five-layer architecture consists of objects, abstraction of objects, service management, application, and business layer. Sensors such as Zigbee and RFID are used in the perception layer of the IoT architecture (Jing et al., 2014). The compilation and storage of data are the responsibility of these

sensor units. There are specific sensor devices installed for network data collection and storage. The IoT architecture also includes a network layer. This layer manages to transfer data from the observation layer to the higher layer. The network layer is also essential in ensuring confidential data and information acquired by sensor devices. The end-to-end IoT architecture is shown in the Figure 2.13. It comprises of four network components: wireless sensor modules, data connectivity, a cloud management framework, and a management portal.

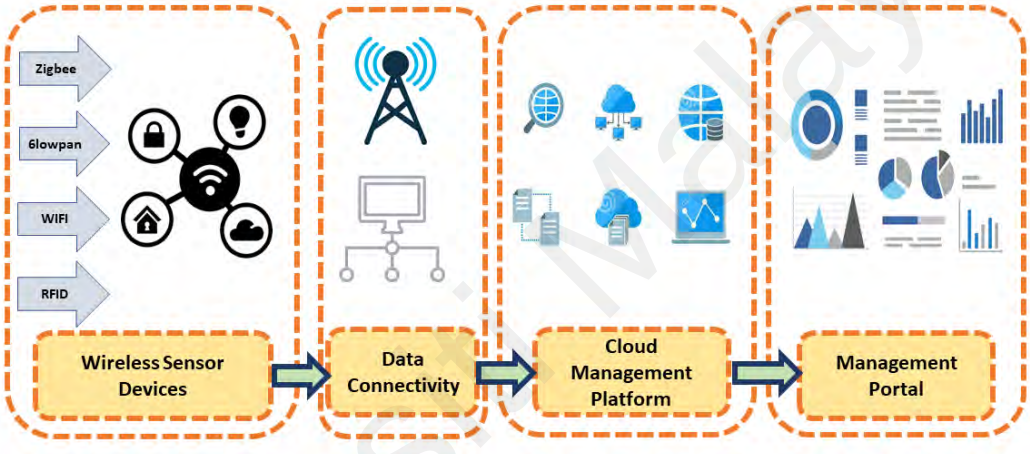


Figure. 2.13: IoTs (IoT) Architecture (End-to-End)

The communication functionality can be described as a cross-layer design and control as an adaptive solution to the system using a resource allocation approach. In IoTs, a centralized optimization model is critical for managing parameters at the physical layer. These primarily include channel error correction as well as modulation. Cross-layer architecture aids MAC and error management in the link layer, while it aids addressing and routing in the network layer (Z. Yan et al., 2014). Via these roles, the device can achieve optimal results in line with the independent goal functions of the cross-layer designs on the IoTs.

In the IoT, cross-layer architecture is also essential for multi-objective optimization. Essentially, the IoT is expected to provide differentiated application services, such that the Quality-of-Service (QoS) provided to various applications. As a result, multi-objective optimization is used to solve a wide range of problems in the IoT. Multiple competing goals within the IoT are overcome with the multi-target optimization. These are primarily delays, energy utilization, and end-to-end packet errors.

The cross-layer technology is made up of layers that assist in improving the performance of the IoTs. The application layer, network layer, and sensor layer are the three main layers of the cross-layer architecture. Each layer is essential in improving the performance of the IoT platform. The sensor layer of the cross-layer architecture, for example, serves as the application foundation for the IoT platform. In the IoT, the network layer roles as a connection between the sensor and the application layer. This layer also contributes to computational technology. The application layer of the cross-layer architecture is liable for displaying the IoT business to the customer. These layers have a vital role to play in the IoTs. Essentially, cross-layer architecture assists in coordination between the network layer and the application layer regarding the data collection and cleansing. The key objective of the strategy is to provide the different IoT users with both fluid and intelligent services. Following are essential problems concern with the cross-layer design.

(a) A. Routing and Communication Among Heterogeneous Devices:

Although IoT technology has been effective in improving intelligent-based communication worldwide, it also has many issues and challenges to be faced. The IoT defines global cyber-physical networks, allowing various applications such as e-health, transportation and tracking of commodities. This application is based on the heterogeneity of the hardware for the IoT requirement for different routing and

communication applications. The cross layer communication schemes have been used to solve the problems encountered in the use of IoT applications (R. Xie et al., 2012)(Guan et al., 2010). The modified Grey Wolf optimizer framework is one of the most critical cross-layer functionalities in solving IoT problems. The optimizer assists in determining the best routing paths and communication criteria. In IoT, the interrelationships between the different functionalities of the application layers are investigated. In realistic scenarios in the physical layer, the cross-layer model is used to validate and optimize the application (Yu & Leung, 2002). The novel updated Grey Wolf Optimizer architecture resolves the standard layered solutions to the IoT to achieve optimal global connectivity. Furthermore, the Delta diagram facilitates global end-to-end connectivity in IoT. Essentially, the method improves both the IoT protection matrix and hardware synthesis.

(b) Application of 6L0WPAN:

The implementation of the 6L0WPAN is also an essential issue in the IoT. Unlike Bluetooth and ZigBee, personal wireless networks help to compress headers in IoT and also compression (L. Da Xu et al., 2014). The data loss in the network caused by low power can be identified using the cross-layer architecture model. The cross-layer mechanism is included to allow the border routers to obtain power from the power grid to detect not only the issue of data loss but also to optimize network communication (Ameigeiras et al., 2010). The cross-layer model is essential considering that it helps routers to collect the complete data context. Cross-layer approaches prevent the packet drop in stage during network communication.

(c) Data Sharing Among IOT Devices:

In the modern world, the fundamental IoT has been significantly changed to consider several other things. For example, the current IoT has been extended to consider the vast number of communications, intelligence, and numerous other remotely

accessible objects, such as smartphones, sensors, and vehicles. Although IoT is being built in a challenging environment where big data is managed, it is vital that complex data management practices, such as cross-layer architecture, become inevitable with increased data sharing/communication in IoT (Z. Li et al., 2010). Big data and cloud computing techniques will be needed to handle more data. Wireless sensor networks are commonly used in IoT since they link many wireless sensor devices. With the increased number of IoT devices, it is critical to manage efficiency and performance adequately. The IoT is expected to evolve dramatically in the next century, with more data being shared by different devices. The cross-layer architecture is proposed to assist in improving the efficacy and reliability of data sharing within the IoT (Bu et al., 2012; R. Xie et al., 2012).

(d) Privacy and Scalability:

It is worth noting that the modern world is heavily reliant on the use of smart devices. As a result of this trend, more devices will be available to meet the human needs soon. As a result, as the use of smart devices grows, more problems will inevitably arise. Security, privacy, and scalability are just a few of the anticipated challenges (Sheng et al., 2013). Since smartphones are designed to be used by a single user, security is a significant concern. The implication is that smartphones contain private and personal information. Security, access controls, confidentiality, and authentication are all addressed using the cross-layer architecture. To build a cross-layer architecture for authentication and authorization, non-repudiation is needed. Encapsulation in IoTs provides authentication and data confidentiality. The datagram transport layer protocol is used to authenticate IoT devices.

(e) Energy Consumption in IoT application:

Energy conservation is one of the significant problems with WSN technology in the IoT. Scavenging or rechargeable batteries can provide energy for the WSN. Whatever

the case, energy management needs to be improved efficiently. Solution : Besides reducing communication and calculating loads, cross-layer designs can be implemented to minimize energy consumption (L. Zhu et al., 2012), (L. Zhu et al., 2011). The WSN communication layers within the IoT affect the impairments of the intrinsic wireless medium, low power radio connections, medium interferences leading to packet losses and path loss.

(f) Middleware for IoT:

With technical advances, a modern IoT architecture has emerged that is more successful in managing major issues within IoT applications. The new IoT architecture includes pattern identification services. This is evident in the physical layer, as well as the middleware and applications layers. The new IoT architecture allows the solution to apply the distinguished algorithms to various environments and devices. In contrast to receiving raw data from the physical layer and others, the application can directly retrieve data from the middleware. This means that the IoT infrastructure would work better. This is frequently due to the new application's ability to retrieve all contextualized data from other layers, regardless of the neighborhood. As a result, the Linksmart middleware represents a significant advancement in the management of the IoT architecture. The diagram below depicts the general design of the middleware between the physical and application layers. The physical layer consists of sensors that collect weather data, event data, bus service data (Souza, AMC da and Amazonas, 2015).

The different layers that make up the Linksmart application serve a variety of functions. For example, the physical layer of the Linksmart middleware cross-layer design hosts the resource layer for smart devices and sensors. This is related to the Linksmart middleware to facilitate data transfer and algorithm detection. Within the IoT scheme, the middleware layer is responsible for pattern recognition and configuration.

Since it contains the configuration parameters for both the resource managers and the application layers, this layer is essential because it allows cross-layer communication.

On the other hand, the application layer is vital for carrying bidirectional communication and the various configuration features for cross-layer communication. Cross-layer communication is an essential aspect of the architecture (Yu & Krishnamurthy, 2007). The architecture allows nodes in IoTs to access, execute, and transmit information through each physical node. The network access points, the middleware layers, and the application nodes all communicate effectively throughout the network. It is a distinguishing feature in IoT that is critical in improving inter-node connectivity across the application layers, middleware, and physical layers.

2.5.4 Cross-Layer QoS Strategies

This section presents various QoS technologies in IoT from the MAC, network, and cross-layer models. The threats and opportunities in each of the layers are evaluated for these QoS strategies. Finally, the future research directions for QoS strategies for research and implementation are addressed before concluding this section.

2.5.4.1 Service-Differentiated Real-Time Communication Scheme (SDRCS).

SDRCS is powered by events, it routes real-time traffic via a cross-layer packet architecture that integrated real-time routing into a modern priority MAC scheme. Based on this architecture, the protocol approximates distributed packet speed classification and traffic control. It also localizes decision-making by giving priority to packet transfers to optimize packet speed (Xue et al., 2011).

This approach can be used to prevent the degradation of bandwidth by un-scheduled data packets. It is event-based and can be easily adapted to changes to the network. In addition to locating the multi-channel transmission, no additional hardware

is required. (Bhandary et al., 2016). This approach lacks several QoS parameters, has a high computational cost, and does not provide a solution for redundant query packets.

2.5.4.2 Network Layer QoS Support Enforced by a Cross-Layer Controller (NLQS)

This scheme allows for the distinction of packet-level service depending on the throughput, packet error rate, and latency (Melodia & Akyildiz, 2010). This method can improve QoS at the network layer. It includes an interlayer control unit (XLCU) for networking functions in physical, MAC, and network layers to be configured and controlled (Bernard et al., 2019). This method allows the interactions between cross-layers to be managed without weakening device upgradability, simplicity, and modularity (Melodia & Akyildiz, 2010).

2.5.4.3 Cooperative MAC Protocol for Multihop Networks (MCMAC).

M-CMAC, CoopMAC, is designed to assist low data rate stations with high data rate stations in forwarding traffic for broadcasting. (Jacob & Shamna, 2015). Helpers have often selected keeping the two fast-hop transmissions by replacing with one slow-hop transmission. Every node has a cooperative table (CT) of possible helpers, including the destination and helper MAC addresses and the Euclidean distance and total distance through the helper. It ensures a higher throughput compared to IEEE 802.11 DCF (Jacob & Shamna, 2015).

2.5.4.4 Cluster-Based Cooperative Routing (CBCR) Protocol.

The CBCR protocol includes a multi-hop data-forwarding feature at the link layer, which is implemented with cooperative links that use M-CMAC. This protocol includes two phases: the selection step of the routing relay and data transmission.

i. **Routing Relay Selection Stage:** All node sends periodic beacon messages to its neighbors containing the node's MAC address. Each of these creates a relay table that lists all the neighbors with whom it can communicate. In case of changes in

its entries after its previous broadcast, the Node will further broadcast its next list. The MAC addresses of nodes adjacent to node X are located in column one of its relay table, and the row of the neighbor node contains MAC addresses for neighbors of the adjacent node. Each node selects routing relays independently based on its relay table. The number of nodes connected by a relay node is determined by the number of nodes that it connects (Jacob & Shamna, 2015).

ii. **Step of Data Forwarding:** a node having query should transmit packets by first confirming the recipient being in the same cluster. If the receiver owns a helper, the packet is forwarded to them, but the packet is sent directly to the receiver if they are unavailable. When the intended recipient is within a separate cluster, the relay table is examined to see whether the recipient is accessible via other routing relays. If reachable, packets are sent directly or through a helper, to the routing relay if the relay contains one. When a destination is not reachable through relays, the node broadcasts packets to all relays, it has the advantage to multicast the packets (Jacob & Shamna, 2015).

2.5.4.5 Adaptive Cross-Layer Forward Error Correction (ACFEC).

The ACFEC model exchanges data packets between nodes through the access point (AP) that runs the infrastructure mode whereby the FEC is included in the multimedia data. (Rao & Shama, 2012). These data are processed via the RTP packets by encapsulation using a streaming server. An adaptive FEC controller detects the packet class from the RTP header and retrieves the packet header from UDP. The encoder creates some error-correcting packets, which is calculated by the block's source packet number. The controller monitors multimedia transmissions using MAC failure data, and if a transmission fails, its counter is increased by one. The controller uses the failure counter to adjust the packet number produced after a block is transmitted (Bernard et al., 2019).

If packets are lost, the redundancy rates are changed, and extra packets are generated to replace the lost ones and satisfy the receiving node's requirements (Rao & Shama, 2012). The FEC packet number is increased or decreased to meet the receiver's requirements and prevent packet errors. This is accomplished by detecting packet losses accurately and adjusting redundancy thresholds. It guarantees better QoS by reducing the packet loss and adjusting the redundancy rate. (Sun et al., 2011)

2.5.4.6 Balanced Cross-Layer Fuzzy Logic (BCFL)

(M. Li et al., 2013) developed a new fuzzy logic-based routing algorithm (BCFL) that takes the distribution of cross-layer parameters as an input to the fuzzy logic inference scheme. Based on the value of distribution, each cross-layer parameter has a dynamic weight. The concept incorporates the following innovations:

It reduces algorithmic complexity significantly; the size of its distribution list determines the parameter's weight. It is built on straightforward if-then rules that remain constant even as the constraints increase. It is easily adaptable to changes in network conditions. The algorithm can be used to choose a CH in the protocols of cluster routing (M. Li et al., 2013).

2.5.4.7 Minimum Hop Disjoint Multipath Routing Algorithm with Time Slice Load-Balancing Congestion Control Scheme.

MHDMwTS is a two-stage routing protocol consisting of route construction and path recognition phases. There are three disjointed paths: primary, alternating, and backup paths, each with different sources. The path build-up process begins when the source node requests to build a route to the nearest hop neighbor (Alamri & Abdullah, 2016). Initially, the source activation process starts in the first stage by adding its numbers and timestamps to the desired path construct node and sending it to the next least hop-

count node. This process continues until the least latency sink has received the necessary data to construct the main path (Sun et al., 2011). In the second stage, route extraction occurs when a new packet from a different path arrives, comparing the extracted path with the primary path. In the case of a shared node, the packet needs to reject, or an alternate path will be sought as a backup by comparing previously taken paths. In the third phase, the sink returned the ACK packet to a sender with path information and time data after the timestamp is calculated. This protocol is limited to cover only a few QoS parameters, i.e. latency and congestion control (Bernard et al., 2019)(Alamri & Abdullah, 2016).

2.5.4.8 Cross-Layer Optimal Design (CLOD)

Authors in (M. Li et al., 2013) proposed a CLOD for data link layer scheduling, network layer routing, and transport layer congestion control under the assumption of fixed link capacity. Energy performance is increased by means of congestion management. The congestion of node at the transport layer is reduced by compressed sensing (CS), which reduces the transmitted bits, While the optimal allocation of resources reduces the congestion at the data link layer (J. Yan et al., 2016). It increases energy efficiency by extending the network lifespan and performing congestion management.

Table 2.7: A summary of some reviewed cross-layer models

Protocol	Layers	Aims	Comments	QOS parameters
SDRCS	MAC/PHY	Routing real-time traffic	Transmits multimedia traffic via cross-layer packet forwarding	Throughput, Latency
SUIT	Application / Transport	Transmission of non-real-time video	Uses fuzzy logic to regulate congestion	Transmissio n rate, delay
CLASS	Any two	Design serves as a framework for implementations of different application scenarios	Scalable, efficient, flexible with very low propagation delay	Propagation delay, jitter
NLQS	PHY/MAC / Network	Permits packet-level service differentiation as a function of throughput, packet error rate, and delay	Network layer QoS is enhanced	Throughput, delay, packet error rate
CMVT	Application / Network	Encodes video streams to video data frames via MPEG-4 encoding and does route discovery and data transfer	Greedy forwarding and rollback are used to find source-to-sink paths	Energy
CoopMAC -	PHY/MAC	Offers spatial diversity among the three nodes	Helper ID signal finds out how feasible cooperative communication	Throughput, rate, delay
M-CMAC	PHY/MAC	Increases end-to-end throughput and packet delivery ratio	Euclidean distances between nodes are calculated and converted to the data rate for a link	Throughput, packet delivery ratio
CBCR	PHY/MAC	Minimizes control overhead and time consumed in establishing the cooperative paths than M-CMAC	Energy consumption is more uniformly distributed in a network enhancing network lifetime	Throughput, packet delivery ratio, energy

MAC-PHY	MAC/PHY	Meant to enable PHY layer cooperation and maximize gains of cooperation at the MAC layer	Leverages both spatial diversity and coding gain	Throughput, delay
MAC-Centric	MAC/APPL	Meant to support QoS needs in new video apps and enable MAC layer differentiation for H.264 partitioning	Targets multimedia applications by using the MPEG-4 scheme	Delay, packet loss rate
ACFEC	MAC/Network (UDP)	Meant to enhance the quality of video streaming over 802.11 WLANs and overcome packet losses	Adjusts redundancy rates to overcome channel fluctuations and detect and reduce packet loss	Packet loss rate
MHDMwTS	—	Meant to provide reliable data transfer with the multipath routing and load-balancing congestion control method in WMSNs	More reliable than basic routing schemes for transport multimedia data	Latency, package transmit rate
BCFL	—	Introduces dispersion into fuzzy logic-based routing and sets every cross-layer parameter with a dynamic weight	Can be used to select a CH in clusterbased routing protocols and proposes a dispersion formula	Node utility, dispersion
CLOD	Datalink/ network/ transport	Prolongs network lifetime and achieves congestion control and designed for lightly loaded WSNs	Assumes fixed link capacity and integrates compressed sensing technology	Throughput, average energy, CS error ratio

Table 2.7 summarizes these cross-layer models and makes a few observations. In real network, multiple layers perform various roles and provide various essential services. A layer can only interact with its neighbors. Because of the numerous features in wireless

communication, the layered model degrades device efficiency. Routing protocols share QoS parameters in the layers of IoT to maximize performance. However, cross-layer models as mentioned in Table 2.7 is a vital solution to handle the query flooding. The cross layer design is more scalable, efficient, flexible with very low propagation delay, can regulate congestion, improve QoS and can be used to improve routing efficiency by terminating redundant queries during the communication process (Bhandary et al., 2016).

In CLD, the physical layer is extremely important. To improve QoS, rate adaptation and channel allocation occur at the physical layer through signal processing. CLD provides solutions for power management, minimizing energy consumption, managing network flow and congestion, and fault tolerance, making it an attractive option for designers considering other layers. It is also desired to design CLDs to influence QoS network activity, terminate unnecessary and redundant query packets, and minimize network flooding in the IoT (Bernard et al., 2019).

Network flooding has been studied in a range of fields. For instance, Hy-IoT was proposed for a hybrid energy-aware clustering protocol to the heterogeneous IoT network (Sadek, 2018). To manage a heterogeneous IoT network, Hy-IoT delivers a real-world cyber IoT architecture centered on clusters. It also provides an effective means of picking cluster heads, enhancing the use of motes energy, therefore increasing the network life and the transmission rate for the packets to the base station. A vital issue in this method is dealing with redundant queries in both the population of motes and the network density in order to integrate an IoT controller. Haddad et al. (2017) also proposed a three-level architecture for IoT redundancy control. To regulate service query redundancy on three scales: macro, meso, and micro scale in IoT networks correspondingly, the framework employs the Explicit Spatio-Temporal model. Several other essential elements, like the

functional architecture, algorithms generating redundancy data and related complexity, and the framework's proactive redundancy control mechanism, are not considered. (Abdelaal et al., 2016) developed a strategy for increasing energy efficiency in QoS-Constrained WSNs using the Divide-and-Conquer (DnC) method. The fundamental principle underlying DnC is that the QoS parameters should be controlled while giving the right amount of network life. However, a real testbed is required to assess the suggested technique more realistically.

Alqahtani et al. (2016) Proposed End-to-End (E2E) QoS design and supervising plan for IoT networks. The authors employed Service Level Agreements (SLA) to find the flooding problems in numerous IoT devices like smart environment, smart water, smart water, smart metering, smart agriculture, smart farming, industrial control, e-health, logistics, home automation and domestic. SLA don't accommodate unified/standard processes to collect the needed metrics across-layer and from various providers for E2E.

2.6 Conclusion

This Chapter explains the concept, underlying research strategies and simultaneous works in context to query processing in both WSN and IoT Networks. It analyzes QoS issues of query execution mechanism of IoT and WSN by devising a thematic taxonomy that presents various parameters. It provides insight into various redundant and unwanted queries that affect the network resources. It discusses state-of-the- art techniques used to detect flooding queries. Moreover, a brief discussion is presented to highlight possible solutions for efficient network flooding in IoT and WSN. Finally, various challenges are highlighted regarding handling the redundant routing queries.

Various defensive mechanisms are designed and developed to handle flooding in IoT networks. As previously mentioned, redundant and undesired transmissions flood the

network, potentially increasing network queries. It also increases the use of network resources and decreases QoS over the link.

With the increased use of smart devices, more challenges emerge in the performance of the IoT. The cross-layer design is normally used to resolve prominent problems in the IoT, comprising of such as security, privacy, energy consumption and efficiency among other error. To address these challenges, cross-layer designs have been embraced. The proposed cross-layered design enables the query exchange with the rest of the layers thereby creating a better Quality-of-Service (QoS).

Based on the shortcomings of the present techniques, which focus mainly on how to enhance the basic routing queries scheme for IoT devices, have limited feature and only provide individual QoS solution to IoT Layers. However, a sole QoS enabled cross-layered solutions for flooding suitable for both physical and network layers devices are not being addressed previously. This study proposed a QoS enabled cross-layered Cluster Based Flooding solution for IoT network. The proposed technique is a solution that is compatible with both physical and network layer devices.

CHAPTER 3: CROSS-LAYERED CBF (FOR MITIGATING REDUNDANT QUERIES IN IOT)

This chapter presents a QoS enabled cross-layered clustering technique for mitigating flooding queries in IoT networks. The cross-layered cluster-based flooding mechanism can provide interoperable solution for the devices of network and sensor layers. This method split the network into various clusters. Inside cluster, the Intralayer cluster (IALC) is used to proactively maintain local query information, on the other hand Interlayer cluster (IELC) is used to reactively acquire routing queries to the receiver outside the cluster. Cluster based flooding (CBF) is a hybrid method that can be more effective compared to conventional systems in terms of query traffic generation. However, if proper redundant query detection and termination mechanisms are not used, the CBF may generate more control traffic than typical flooding techniques. For the purpose to minimize the control traffic, energy consumption and network flooding IELC uses an advance query control mechanism (QCM) which corelate the signal strength with a predefined threshold value (QLT) QueryLimitThreshold to identify and terminate redundant and unwanted routing queries.

The mote localization and query detection strength of an IoT network is checked in flooding using this approach. The strength of query detection is examined for verification and any fluctuation in the signal strength of the query packet and the QLT. The mote's position stability is dependent on the location information of its valid mote neighbors, combined with its fixed QLT, and monitored over time at various intervals. This study also carried out a formal analysis using the salient features of Set Theory and Game Theory (Abdalzaher et al., 2016) that helps to identify diverse network flooding patterns in the sensor and network layers of the IoT Networks.

This Chapter is organized into five sections. Section 3.1 provides detailed methodology of the study. Section 3.2 starts with an overview of flooding strategy. It followed the design principle of the proposed method in Section 3.3, which provide a description of the state transitions along its event generation for each phase. Section 3.4 and 3.5 elaborates the (CBF and QCM) with detail description of each of its phase. Also, each phase explains along with its algorithms to have an insight of its working steps. Finally, this chapter is concluded in Section 4.6.

3.1 Methodology

The entire study is divided into five phases, as depicted in Figure 3.1. This study investigated the QoS implications of the entire WSN and IoT networks with the exact state of the art solution pondering the most basic to the latest trends. A review and analysis are carried out of the QoS vulnerabilities, unwanted and redundant network queries, and potential IoTs Network (IoT) challenges. The study uncovers the contemporary layered based clustering of the reported redundant and unwanted queries, and associated challenges to the IoT in context of main categories of QoS implications relating to each IoT layer. Moreover, it highlights the possible redundant and unwanted flooding queries impacting the performance of individual layer and suggests the compact solution in design of QoS enabled IoT.

Existing cutting-edge outcomes are also critically examined in developing a comprehensive thematic taxonomy. Furthermore, this study examines each cutting-edge QoS solution to determine the distinguishing IoT aspects used, and the problem addressed by a specific technique, as well as the simulation or emulation environment of the related technique. This research also examines the impact of each state-of-the-art QoS-based query solution on the appropriate IoT layers. The critical examination of existing state-of-the-art flooding solutions broadens domain knowledge of current IoT QoS trends,

significant strengths of prospective IoT, and research gaps that require further exploration. It is evidently noticed that flooding query remains the fundamental issue and a challenge that affects the QoS of the IoT network.

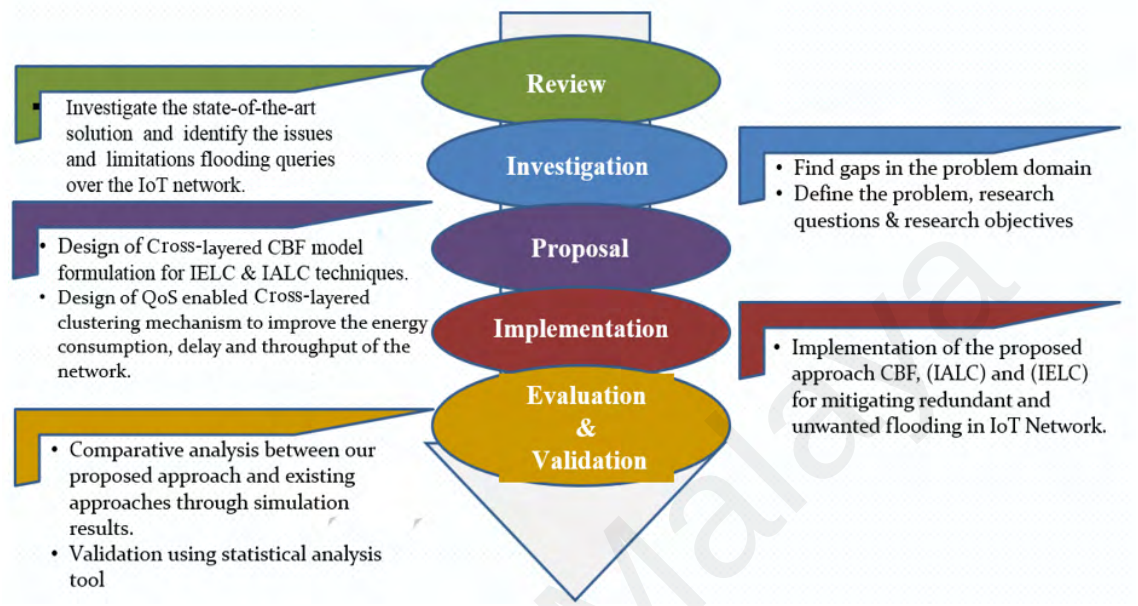


Figure 3.1: Research Methodology

A complete investigation is carried out in the real-world scenario of IoT to establish the problem. This study conducted a formal analysis of the prominent features that assist identify various network flooding patterns in the sensor and network layers of IoT Networks using the Event-B method (Set Theory and Game Theory).

The research validates the system exploiting formal proof on a theoretical mathematical model of the system that comprehensively verifies and confirms the expected behavior of the CBF Approach, as opposed to simulation and testing. This research also exhibited and assessed the impact of redundant flooding on IoT controllers and the detection accuracy behavior and analysis of some of the most current state-of-the-art. This study examines different techniques to address redundant/unwanted communication inside the IoT network to comprehend the sequence of measures taken during flooding and propose the cluster-based flooding (CBF) technique. Both physical and network layer devices can

use the CBF technique because it is an interoperable solution. CBF splits the network into various clusters. Intralayer clustering (IALC) keeps the local queries proactively, while interlayer clustering (IELC) ensures that the routing queries to destinations beyond the cluster are obtained reactively. CBF is a hybrid technique that has the capability to be more efficient in context of query traffic generation than classical schemes. CBF is vulnerable to cause more control traffic than traditional flooding techniques without adequate redundant query detection and termination mechanism. Interlayer clustering (IELC), composed of an advance query detection and termination technique (QCM), employs strength of link signal, and a query limit value to detect flooding. It can minimize the network flooding, energy consumption, identification, and elimination of redundant/unwanted flooded queries in IoT networks. The findings of the simulation reveal higher performance in the context of traffic delay, throughput, and energy consumption against state-of-the-art techniques under different performance metrics compared to conventional flooding and state-of-the-art systems.

This study also exhibits statistical performance and evaluation of query control mechanism to minimize energy consumption, delays, and network throughput. In Particular, it assessed the performance gauge of Query Control Mechanism (QCM) for QoS-enabled layered-based clustering for flooding on the IoTs. This study used statistical methods to determine that the QCM algorithm beat the existing techniques for identifying and eliminating redundant flooding queries. Such as Divide-and-Conquer (DnC), Service Level Agreements (SLA), and Hybrid Energy-aware Clustering Protocol for IoT (Hy-IoT). The study inferentially analyzed for performance- measures of algorithms in context of three different scenarios, i.e., energy consumption, delays, and throughput with

different intervals of traffic, malicious mote, malicious mote with realistic condition, Scenario based on a varied mobility speed varied simulation area and varied pause time

The results showed that the QCM algorithm performed significantly as against the existing algorithms depicting a statistical probability value "P" less than 0.05. It indicated that the performance of QCM achieved the 95% confidence interval. Thus, the study inferred the performance of the QCM as substantial against other algorithms.

3.2 Flooding Strategy

This Section presents a modelling strategy for flooding in IoT networks. It elaborates the interchange of query messages among distinct motes and specifies the number of query streams. In addition, system modelling provides an overview of mote functionality and query flow among end-users and motes (Bourke et al., 2014; Elsayed et al., 2013; Yang et al., 2006). Figure 4.1 depicted the system model of redundant/flooder, sender/sink, and destination motes. The flooding is only feasible once the flooder mote is located between the receiver and the transmitter. According to the system's geometric shape, the flooder's transmitted signal gets received by the destination before it completes transmitting or establishing a new connection. Equation (1) denotes the fraction of each mote interval that must stay unflooded for successful communications and represents the different regulated distances D3, D2 and D1.

$$D2 + D3 \leq (\lambda X_s - X_f)c + D1 \quad (1)$$

X_s denotes mote duration i.e sum of valid motes, where λ is a constant value allocated to every authentic mote in the network and it varies between 1 and 10, Xf denotes the flooder's processing time which is the time it takes the flooder motes to strike the network. The distances between the motes are denoted by *D1, D2*, and *D3*, while the

speed of light is denoted by c, as seen in Figure 3.2. The flooder mote constantly observes the communication channel, and when it detects a query packet transmission, it quickly emits a radio signal to generate a receiver-side collision. The flooder mote can transmit enough energy to lower received bits of query messages, leading to cyclic redundancy (CRC) failure. A flooding attack usually meets the following criteria: tremendous energy efficiency (use very minimal energy), low detection probability (near zero), high levels of DoS, i.e., to disturb communications to the intended or maximum degree and to be resistant to flood control techniques of the PHY layer, i.e., to block the signal processing techniques to handle the flooding. In all situations, a flooder mote tries to preserve efficiency according to these criteria. This allows the flooder mote to adopt stable approaches with behaviors in the sensor and network layer to maintain a minimal detection opportunity.

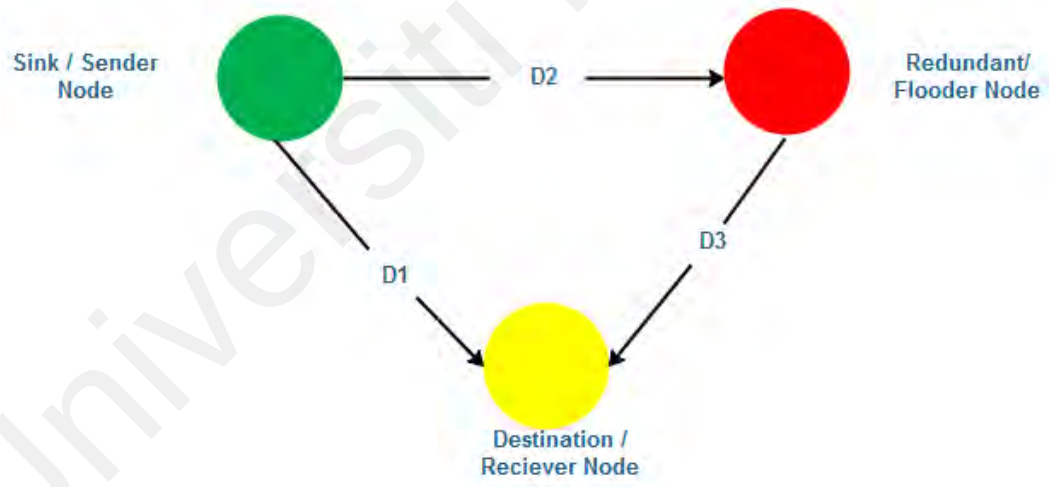


Figure 3.2: Sink Mote, Destination Mote, and Redundant Mote System model.

3.3 Cluster Based Flooding (CBF) for IoT

The CBF's fundamental principle is to divide the whole network into several routing clusters. Proactive maintenance with the aid of route query exchange

and update query packets is done by Intralayer clustering (IALC). In the case of a broken or established connection between the directly connected neighbor mote, the MDP-level MAC begins route updates which are specified as the directly connected neighbor mote and share a communication link. Mote discovery and media access control protocols provide the services to identify the neighbors of all connected motes. On the other hand, Interlayer clustering (IELC) provides services of reactively transfer query packets to the motes which resides outside of the cluster by means of route query reply. IELC sends routing queries to its borders or outlying motes via a broadcast delivery service. IELC uses IALC tables to keep track of updated route clustering information for peripheral motes.; based on the QueryLimitThreshold (QLT) value, this information is then utilized to evaluate if the query for destination mote relates to their cluster. QLT enables the maximum motes transmitting capacitance to be regulated to send the maximum number of query packets and enables network flooding detection to occur. Figure 3.3 elaborates the CBF architecture.

3.3.1 CBF Assumptions

The following is a list of the CBF network assumptions:

- *n* number of sensor motes are deployed randomly.
- All motes provide the same capacity with respect to functionality,

  every mote has an IP address and may function as a sensor gateway, allowing query messages to be exchanged.
- The mode of communication is single-hop and multi-hop for all connected motes.

Any mote can launch a flood attack. The flood begins when it detects any activity and begins interrupting the communication.

The flooder and regular motes have the same capabilities, however the flooder mote may additionally generate duplicate query messages (i.e., random flooding queries).

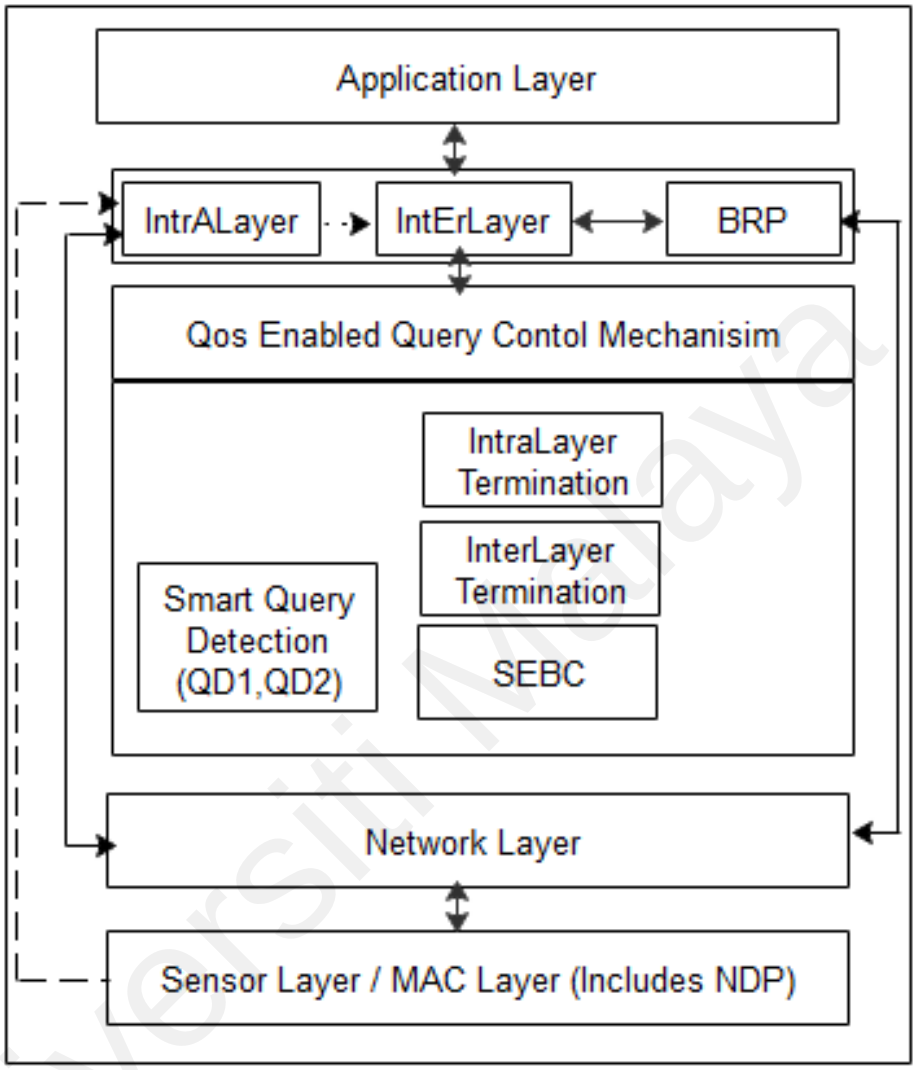


Figure:3.3: The CBF Architecture

Formal definition, MDP algorithm, IALC, IELC, QCM, and network assumptions are explained in the following subsection.

3.3.2 Neighbor Mote Discovery Phase (MDP)

In this Section a neighbor mote discovery algorithm will look after the maintenance of neighbor and cluster routing tables. All motes have the information tables of its neighbors and cluster. The neighbor-mote table holds accessible QoS parameter values along the link between itself and its neighbor-mote, as well as the

neighbor-mote addresses. The intralayer clustering algorithm requires these parameters for selecting the best available routes in the cluster. During this phase, every mote sends beacons to its neighbors on a regular basis. Every mote refreshes its neighbor table with appropriate values when it receives these query packets from a nearby mote. Every mote exchanges neighbor tables with its appropriate neighbors and build cluster routing tables (Gammarano et al., 2018), (Kharche & Pawar, 2017).

Every mote periodically transmits "hello" beacons to its directly connected motes to ensure their status. When a mote delivers a beacon, it immediately updates the neighboring table and records the source of the beacon. Every mote checks its neighbors' mote tables at periodic sample intervals to verify their neighbors' state. If a neighbor beeps no beacons throughout the Max_previous_list interval samplings, it is considered lost. If the neighbor beeps beacons, it is deemed found. Whether a neighbor is found or lost, a notification of an updated link is forwarded to IALC. Figure 3.4(a, b) shows the protocols.

3.3.3 The IntraLayer Clustering (IALC)

The motes measure the routes of intralayer clustering based on link state information for each extended cluster of motes. An interrupt raised by mote discovery protocol (MDP) or IALC link state query packet may become the sources to generate a mote information, received by the link state updates table. All the related link states information is maintained by the link state table. Moreover, it is necessary to recompute the clustering table with any recent updates related to waiting link state. Additionally, the link state tables get updated when the outlier links have been removed. The most recent released updates of link state (with its sources) are propagated to the mote's neighbor contained by the cluster. Finally, a newly discovered neighbor received the entire intra-

cluster link state information from the mote. The protocol is described in Figure 3.5 (a,b).

3.3.4 The Interlayer Clustering (IELC)

The underlying job of the IELC is to find out routes to the hosts (that lie outside the mote's cluster). If IALC local clustering table does not contain the destination mote's information, then a link query request is initiated by the IELC at the network layer. Each route query request allocate with a query-id (unique to the source mote id). A combination of query-id and source-mote-id might uniquely identify the route query request in the network. Figure 3.6 despite the protocol.

Once the request packet records the query-id and source-mote-id, then the query packet may forward to all the border or peripheral motes of the cluster. The detected-queries-table then records the route query request from the mote including other information i.e., the query-id, source-mote-id, broadcasting-mote, and last-hop. The mote then investigates the routing table to find the desired destination mote (if any) residing inside the cluster. The mote responds to the query source, in case the destination mote is found, and replies with a route query reply (carrying a path identified by the last hop information stored in the spotted queries table. In case, the destination mote does not exist in the mote's cluster, the mote, subsequently broadcasts the rout query request to all the outer motes to find the destination mote outside the cluster. The broadcasting mote may generate more control traffic (compared to flooding) if a suitable query control scheme is not available.

An alloy (query detection and early termination) of advanced query control mechanism can exploit to investigate the strength of cluster-based querying. A mote will stop transmitting route query requests (on the departing link based on the

information stored in the discovered queries table), in case the broadcast receiver mote intends to accept the route query request packet.

MDP Query Packet Format

0	4	8	12	16	20	24	28	32
Version			Traffic class		Flow label			
Payload					Next hop		Limit of hops	
Source mote address (mote_id)								
Destination mote address (mote_id)								

Neighbor_mote_table

Neighbor_mote (mote_id)	Arrival (Boolean)	Previous_list (int)

Initial Setting:

Timer-xmit-beacon = random-uniform ($t_{beacons.}$ 2);

Timer-mote-table-update = random-uniform ($t_{beacons.}$ 2);

Transmission of Mote Beacon:

Figure 3.4(a) MDP Query Packet format and Neighbor mote table.

// sporadically transmission of "Hello" beacons to neighbor-motes

Mote-source = mote-id;

Payload(query-packet);

Announce(query-packet);

Timer-xmit-beacon = $t_{beacons}$ ++;

Delivery of Mote Beacon:

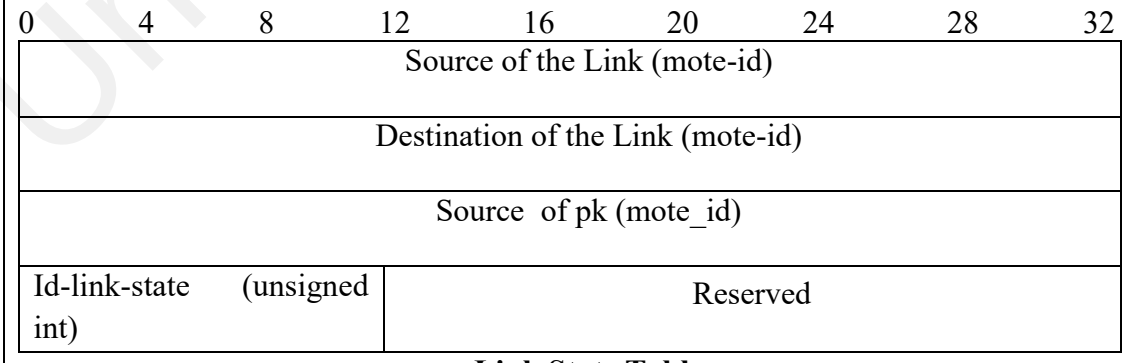
//From delivered mote beacons, noted down detected neighbor-motes in Neighbor

Algorithm 1. MDP neighbor table update.

```
//Identify and eliminate the missing and lost neighbors-mote. Every note has neighbor-mote-table.
Begin (Mote-Table )
 Input : Mote-Table[neighbor-mote].previous_list, Mote-Table [neighbor-mote]. Arrive.
Output: Update-Timer -Mote-Table,
1: If (Mote- Table[neighbor- mote]. Arrive ==Not True ) Then
      2: If(Mote- Table[neighbor- mote]. previous_list ≥ Max_previous_list )Then
            i: Eliminate (Mote-Table[neighbor- mote]);
             //If previous_list has not received the neighbor's-mote beacon then
             eliminate the neighbor-mote from the mote-table.
            ii: Interrupt-load- params (neighbor- mote);
            iii: Set-interrupt (IALC, "neighbor-mote-lost", "Update IALC Routing
                Table");
       Endif
3:Else Mote-Table[neighbor-mote]. previous_list ++;
//Increase number of cycles that neighbor-mote's beacon has not been received.
Endif
4: If (Mote-Table[neighbor-mote]. previous_list == −1) Then
//Immediately, alert and update the IALC, if a new neighbor-mote found.
   i: Interrupt-load-params (neighbor-mote);
   ii:Set-interrupt (IALC, "neighbor-mote-found", "Update IALC Routing Table");
   iii: Mote-Table [neighbor-mote]. previous_list = 0;
Endif
6: Mote-Table [neighbor-mote]. Arrive = Not True;
7: Update-Timer -Mote-Table = T ++;
8:End
```

Figure 3.4 (b). The Neighbor-Mote Discovery Protocol (MDP).

IALC Query Packet Format



Link State Table

Source of the Link (mote_id)	Destination of the Link (mote_id)	Id-link-state (unsigned int)

Pending Link State Table

Source of pk (mote_id)	Source of the Link (mote-id)	Destination of the Link (mote-id)	Id-link-state (unsigned int)	Status of Link (Boolean)

Cluster Routing Table

Destination (mote_id)	Is_cluster_member$(Boolean)	Source routes (mote-id-list)

Figure 3.5 (a): Intralayer Cluster Protocol (IALC).

Algorithm 2. Intralayer cluster (IALC) .

//IALC might be triggered by either from an interrupt made by the mote discovery protocol (MDP) or link state query packet updates.

Begin

Input : query-packet arrived, Interrupt-extract-params, destination-link , Found Neighbor-mote

Output: My-id-link-state, Status-link.

1: **If** (query-packet arrived) **Then**
   i: Extract(query-packet);
   ii: My-changed-link = Not True;
2: **Elseif** Interrupt-extract-params (&destination-link); **Then**
   i: Source-link = my-mote-id;
   ii: Source-pk = my-mote-id;
   iii: Id-link-state = my-id-link-state;
3: **Elseif** (Interrupt-type = "Found Neighbor-mote) **Then**
   i: Status-link = Up;
   // Share all intra cluster link states information to the new discovered neighbor-mote.
4: **Elseif** (neighbot-mote! **(**Link State Table, my-mote-id, link-destination**)) Then**

```
    i:  Forward-link-state-table (Link State Table, link-destination);
    ii:  My-id-link-state ++;
5:  Else
    i:  Status-link =  Down;
6:  Endif
    7:  If (neighbot-mote is (Link State Table, my-mote-id,  link-destination))
        Then
            i:  My-id-link-state ++;
8:  Endif
9:  End.
```

Figure 3.5 (b): The intralayer cluster (IALC) algorithm.

IELC Query Packet Format

0 4 8 12 16 20 24 28 32

Source of the Query (mote-id)			
Destination of the Query (mote-id)			
Last_hop (mote_id)			
Broadcasting-from-mote (mote_id)			
Query-id (unsigned int)	Type-pk (char)	Marker-link (unsigned int)	Hops-maximum (unsigned int)
Route-source [mote 0] (mote_id)			
Route-source [mote 1] (mote_id)			
To			
Route-source [mote N] (mote_id)			

Table for Detected Queries

Source (mote_id)	Query-id (unsigned int)	Last-hop (mote-id-list)	Net-query-coverage

Figure 3.6: Interlayer cluster protocol (IELC).

3.4 Query Control Mechanism

As stated earlier, the important aspect of the flooder mote is required to generate redundant or unwanted routing queries so that the link remains busy, and the authentic mote should be stopped for sending wanted or important query packets. There is a need of mechanism to quantify the amount of total time spent in waiting for the link to get free, also to ensure the signal strength related to query packet along with location consistency of mote. Then, these metrics are compared with the regular time of traffic and estimate the link for redundant queries. Thus, query control mechanism has been introduced to execute and implement such tasks. As described in equation (2), the QCM method employs a change in QueryLimitThreshold (QLT) for detection and termination of unwanted or redundant query packets. It is noticed that the QCM is significant to boost the performance of IoT network in context of the signal strength of query packets, and thus can improve the location consistency for ensuring the connected motes. This helps in protecting the network for reactive flooding attacks. Let calculate the QLT as described in Equation (2):

$$\sum_{i=1}^{n} Md_i = Pf \tag{2}$$

In Equation (2), *M* represents the maximum query request packets for neighbor motes, *n* depicts the net quantity of motes, d_i shows the distance between each mote and QLT (demonstrating the distance between the communicating motes, with *i* as the number of relevant motes, ranging 1 to *n*) and *Pf (Pf* investigates any possible flooding attack on the network), respectively.

The QCM method employs signal strength of link in investigating the consistency of query packet looking for, if the QLT is counter to normal pack transformation value. The transmitting note, sending QLT, is significantly tested and

compared with the current value, to find the position of flooding in the network. A mote is a flooder if it's QLT crosses the highest query packet of a mote. In addition, a mote is not considered to be a flooder if the QLT value is less than the highest query packet.

Algorithm 3. Query control mechanism (QCM)

Begin
Parameters: n, MaxQuery(M): M ∈ Neighbor motes
Input: query limit value, x_0, y_0, x_n, y_n :
Output: Dist, Query limit value (Δ) ;
1: **If** (MaxQuery(M) < QueryLimitThreshold(QLT) **Then**
    i: Check (SignalStrengthConsistancy(SSC));
    ii: Check (Sending QueryLimitThreshold(QLT));
    iii: Check-link = ((SignalStrengthConsistancy(SSC), MaxQuery(M));
2: Elseif (Check-link =Not True) **Then**
    i:Flooding occurred;
3: **Endif**
4: **If**(MaxQuery(M) > QueryLimitThreshold(QLT) **Then**
    i: $Z_0 = (x_0, y_0)$ = mote_location;
    ii: $Z_n = (x_n, y_n)$ = find_mote_location;
    iii: Check-link = Query Packet Sent;
5: **Elseif** (Check-link= Not True) **Then**
    i: Flooding occurred;
Endif
End

When the QCM detects a flooder mote, an alert is generated and passed to all connected neighboring motes so that to switch the motes' routing paths leading to the flooder mote. By this means, QCM eventually removes the flooder mote from the network. The QCM algorithm is stated in Algorithm 3.

n presents the quantity of motes within the network, Z_0 refers to the initial mote position (x_0, y_0), Z_n describes the current position of the mote (x_n, y_n), and M portrays neighboring motes.

The QCM algorithm contains two levels. The QueryLimitThreshold (QLT) information is achieved at initial level, followed by verifying the SignalStrengthConsistancy (SSC) of each mote. Since all motes can disseminate information reaching a particular duration, queries' volume passed by each mote can be recorded. Because of sending QueryLimitThreshold (QLT), the SignalStrengthConsistancy (SSC) need to be validate.

It is supposed that a particular mote can be in one of three states/conditions in the network: typical (normal state of the mote), fishy (the suspicious condition of the mote), and flooder (mote behaves as a flooder). Initially, the motes in the network are in the typical state and query packets could be exchanged by them using a single- or a multi-hop communication pattern. The fishy state carries the path analysis relying on the type of communication (either single- or multi-hop). This can be further exploited by the motes to query exchange of data. In case, the fishy sender mote relies on a single hop communication pattern. On the other hand, the analysis of mote can be achieved based on multi hop if the fishy mode relies on multi hop communication pattern, this can help to query the packets that were transmitted by all motes. Finally, a mote is considered to be a flooder if a pattern contradicting the normal pattern of queries arise, and the way the mote transmits the query packets. A mote can be considered to be in a typical state if the quantity of MaxPacket(M) is analogous to QueryLimitThreshold (QLT). Eventually, the QCM removes all the flooder motes, because of updating the connected neighbor motes by modifying the communication channels and links (approaching from the flooder mote). Figure 3.7 elaborate the whole process related to cluster-based flooding.

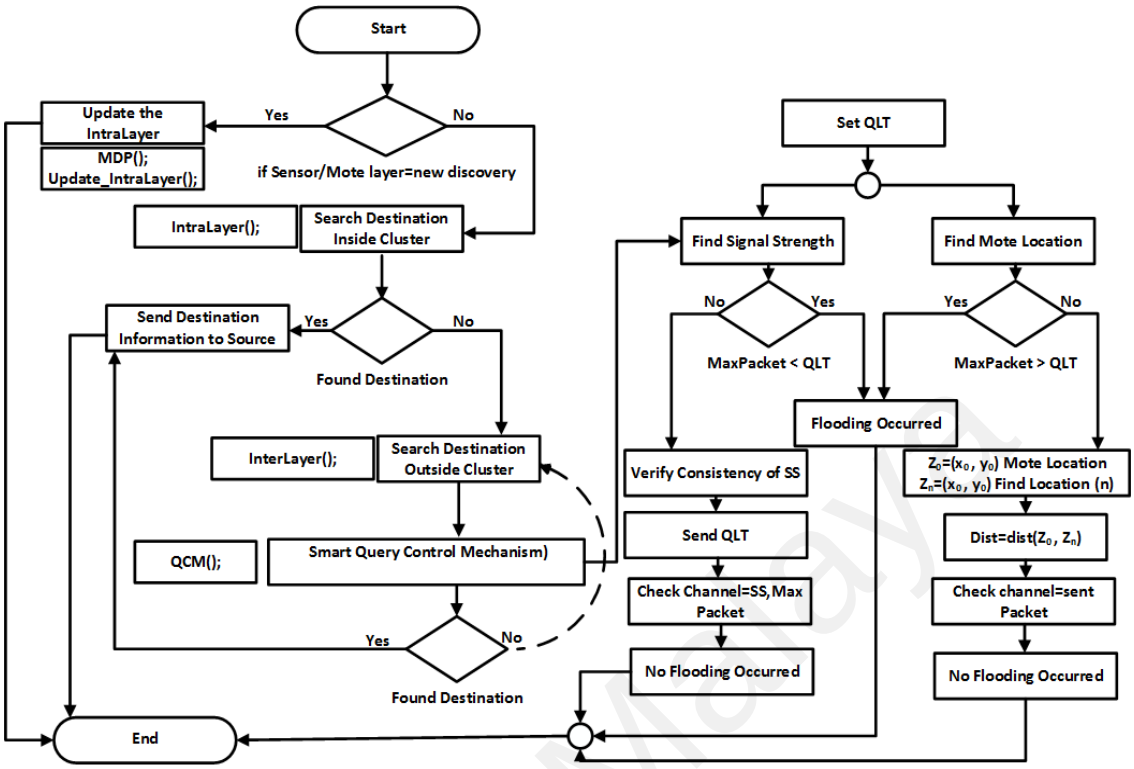


Figure 3.7: Flowchart of the CBF process.

3.5 Model Formation for Cluster Based Flooding (CBF)

This study designed the cluster-based flooding model as either one of two kinds of games: proactively (intralayer) advantageous for the sensor layer and reactively (interlayer) beneficial for the network layer. The intralayer clustering aims at employing more network resources in achieving the recent information related to the motes in handling priority packet queries, and to reduce the delays in the IoT network employing the table-driven mechanism. A static topology of the network is assumed, in addition to assuming that intralayer clustering (IALC) is aware of variation in topology. Intralayer clustering (IALC) eventually improves CBF by repairing the query route and it caches inside the cluster. This method owns a significant feature of non-dependent on any assumptions relating to the size of individual mote's cluster. Finally, the proof of

correctness implies to the networks owning the same cluster radius for every mote, and it also refers to the networks handling the individual mote's own cluster.

As depicted in Figure 3.8, suppose m(t) be the collection of queried motes residing within the cluster (IALC), being an interior or a peripheral mote at time interval t. The queried motes, for instance, motes having been visited already and directly reachable to the source mote. Likely, the remaining collection of unqueried motes are described by m^c (t), unqueried motes are termed as the motes not been visited yet, residing in the outer cluster, and not being directly approachable to the source mote. Suppose B(t) be the subset of m(t), denoted as peripheral or border motes. Every peripheral mote B(t) carries about one neighbor in case of unqueried motes m^c (t) (B(t)→ m^C(t)), peripheral motes are the covered motes, constructing a border between queried and unqueried clusters of the network. It is already lists in Equation (3).

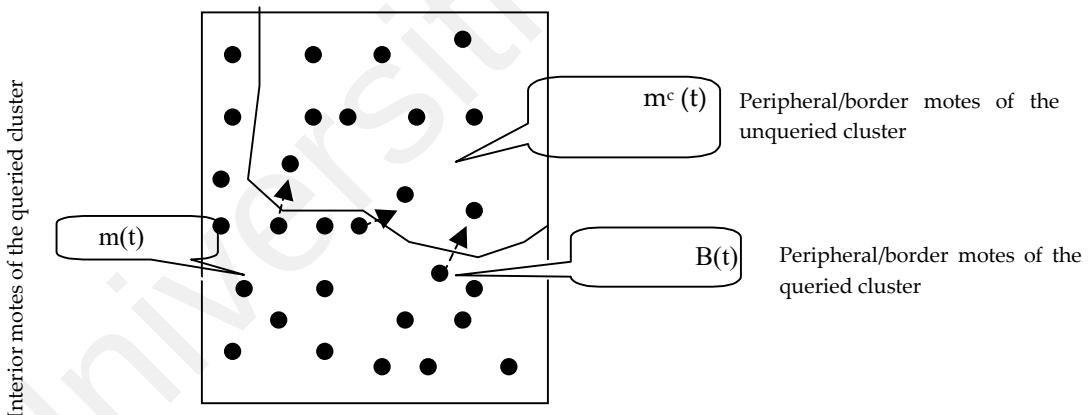


Figure 3.8: Cluster-Based Flooding (CBF) Model Formulation

$$m(t1) \subset m(t2)\, and\, mC(t2) \subset mC(t1), for\ t\,1 \leq t2 \qquad (3)$$

Equation (3) depicts that a mote once queried, it cannot be unqueried or uncovered. Therefore, $m(t_1) \subset m(t_2)$.

The fundamental set principle refers that $m^C(t_2) \subset m^C(t_1)$, as presented in Equation (4):

$$|mC(t)| > 0, |B(t)| > 0 \quad (4)$$

Equation (4) describes all motes belonging to uncoverd mote $m \in m^C(t)$ are reachable by the other motes, having a covered peripheral neighbor $b \in B(t)$ (b denote the collection of queried motes, residing on the cluster boundary). Thus, peripheral mote b was visited by a query, Besides, b can be an interior or a peripheral mote (relating to the source mote). If b is an interior mote, then every b's neighbor means to be queried as well. Though, even some of b's neighbors are unqueried, stated as uncovered motes $m \in m^C(t)$. Consequently, b requires to be the border, or a peripheral mote of a source mote as presented in Equation (5):

$$b \in B(t1): b \notin B(t2); |Mc(t2)| < |Mc(t1)| \quad (5)$$

Referring to Equation (5), suppose that N_b is denoted as the collection of motes that are neighbors of mote b. Then, p will be occurred by the following two conditions:

1. If $(b \in B(t_1)$, then $m^C(t_1) \cap N_b \neq \emptyset$
2. If $(b \notin B(t_2)$, then $m^C(t_2) \cap N_b \neq \emptyset$

The above conditions indicate that $m^C(t_1) \neq m^C(t_2)$, here also presented earlier that $m^C(t_2) \subset m^C(t_1)$, so $|m^C(t_2)| < |m^C(t_1)|$.

In case, a mote b is a broadcast receiver (relating to the source mote) and carries the updated route query for interval t_2 then $b \notin B(t_2)$. Referring to, when mote b achieves the updated route query for interval t_2, it initiates looking for queried destination mote, so revising all its associated motes of their cluster. Hence, it should cover all mote b's neighbors and $b \notin B(t_2)$.

The second kind of interlayer/reactive depicts a two-player game. Here, one player "1" has a role of a “maximizer” and second player "2" roles as a “minimizer”. The maximizer (player 1) aims at achieving the maximum possible volume of energy gains, whereas the minimizer (player 2) attempts to preserve the least volume of energy gains. This mechanism is adopted since every mote (player) of IoT network can possibly exploit the resources of network gains during transmission of the route query packets. These players hold a role of observer mote and are responsible in detection of unwanted flooding in the network. Equation 6 denotes the players as X1 and X2, where X1 refers to inspecting mote and X2 describes the flooder mote.

$$X = \{X1, X2\} \tag{6}$$

(Ic) and *(Ip)* denote the constant inspecting and periodic inspecting respectively in back and forth of the cluster-based flooding (CBF). This allows the communicating motes to examine the link either constantly or with a predetermined interval of time. “ReFa” refers as the assumed strategy of reactive flooding. The mote uses the *(Ic*, *Ip)* strategies to check the communication link, represented as:

$$X = X1 * X2 \tag{7}$$

$$X1 = \{Ic, Ip\} \tag{7a}$$

$$X2 = \{ReFa\} \tag{7b}$$

Here, *X*1 and *X*2 denote player 1 and 2, respectively.

The efficacy of inspecting mote can be presented by assuming the player utility function, to authenticate whether the flooding attacks could be detected significantly or not. Consider the false positive and rate of detection as to primary utility functions to inspect the mote. The flooder utility aims at stopping the successful dissimilation of

query packets to reduce the network throughput, Quality-of-Service (QoS) , by achieving the strikes of redundant query. The utility function (F_u) are set as follows, F_u 1 and F_u 2 represent the detection rate and flooding attack gains, respectively:

$$\{F_u\} = \{F_u 1, F_u 2\} \tag{8}$$

Equations (2), (6), and (7) formulate the strategy of reactive flooding attack based on *Ic* and *Ip* presented below. Here, F_d denotes the duration of the flooding attack, F_{dg} depicts for the gain of detection of flooding, *t* describes the I_p time interval, F_{ag} denotes the gain of a remarkably flung flooding attack. The payoff of the reactive flooding is denoted with P_{refa}, and both (P_c, P_p) are the payoffs when (*Ic*, *Ip*) are used to sense the flooding attack. Payoff describes the cost of starting or detecting an attack:

$$Ic = F_d\left(F_{ag} - P_{refa}\right), \left(F_{dg} - P_c\right) \tag{9}$$

$$Ip = F_d\left(tF_{ag} - P_{refa}\right), t\left(F_d F_{dg} - P_p\right) \tag{10}$$

The strategy of the reactive flooding attack can be expressed by Equation (9) and (10) for both (*Ic*, *Ip*). It is noticed that the flooder mote can instantly trigger an attack if it identifies any activity in the communication link.

3.6 Conclusion

This Chapter presents a QoS enabled cross-layered clustering technique for flooding queries in IoT networks. The QCM is based on an understandable solution that applies to both physical and network layer devices. Cross-layered CBF segments the entire network into numerous clusters, maintaining proactively the local query information using IALC, whereas IELC is held responsible for reactive achievement of routing queries to their destinations (outside the boundary of cluster). CBF carries the potential as an efficient solution compared to conventional schemes in context of query

traffic generation. However, when query detection is absent or terminated, the CBF may generate more control traffic as compared with the standard flooding techniques. The cluster-based flooding model in this study was built as one of two types of games: proactive (intralayer) for the sensor layer and reactively (interlayer) for the network layer. Intralayer clustering seeks to use more network resources to obtain the most up-to-date information about motes to handle priority packet queries and reduce delays in the IoT network using a table-driven approach.

CHAPTER 4: FORMAL VERIFICATION AND ANALYSIS

This Chapter discuss a formal verification and analysis for Cluster Based Flooding using Event-B method and as a case study use to examine the CBF in IoT.

This Chapter is structured into six sections. Section 4.1 and 4.2 briefly presents the motivation for Formal Verification and Modeling in Event-B along with model refinement. Whereas Section 4.3 and 4.4 describes CBF Framework, system requirements and environment assumptions deemed in our development. Section 4.5 presents the entire formal development process, containing the formalization with Event-B and the validation with the ProB. Finally, Section 4.6 concludes the Chapter.

4.1 Motivation for Formal Verification and Modeling in Event-B

Formal methods, mainly formal verification, can improve the quality of the verification system. Formal verification technology can use rigorous mathematical proofs to determine if a system has a particular property. So far, a lot of work has been done to verify route protocols utilizing the model and theorem verification (Bourke et al., 2014; Elsayed et al., 2013; Yang et al., 2006)

Few studies pay attention to the formal validation of cross-layered routing protocols. This chapter proposes a formal specification of cross-layered cluster-based flooding CBF at event B and proves the correctness of the route discovery mechanism. It is a refinement-based method, an improved way to add system details to the corresponding model gradually. It makes modeling and authentication easier for the user by allowing later versions to keep all the proven attributes in the previous model. The basic Rodin (Abrial et al., 2010) auxiliary tool automatically divides proof tasks into sections based on the model structure. This approach has also been extended to systems in a variety of disciplines. Each node in the CBF cluster broadcasts link-state queries

regularly. To model periodic broadcast/flooding activity, some time constraints must be included in the formalization. Also, formalization should limit the spread of node link-state query information within the cluster. Instead of using reactive protocols for broadcasting, nodes typically use a flooding mechanism called border casting services to handle routing requests.

Furthermore, if there are changes inside the cluster or it receives route queries replies carrying discovered routes, a node then updates its routing table accordingly. As a result, formalizing the CBF is more complicated than formalizing a sole proactive or reactive approach protocol. This is an important issue that needs to be discussed and formally specifying this challenge in a significant way. By improving the method, it allows us to design a system from abstract to tangible. The proof obligations ensure the correctness for refinements. A Rodin plugin known as ProB is employed as an animation tool. It is used to authenticate the model and ensure that it has formalized the device specifications.

4.1.1 Event-B Method

Event-B is a state-of-the-art systematic approach for modeling and analysis at the device level (Elsayed et al., 2013) . The B-Method(Yang et al., 2006) has been simplified and expanded to Event B. Set theory, and first-order predicate logic is used to construct Event-B.(Hoang et al., 2013) includes the syntax specification for the Event-B language. This procedure has already been used to validate a variety of complex structures (Abrial et al., 2010; Cansell & Méry, 2006; Elsayed et al., 2013). The context and system construct the Event-B model. The system defines the dynamic part of the model, while the context describes the static role. The following are the apparent meanings of context and machine. The background structure is composed of a tuple, as follows.

Definition 1: S denotes a set of user-defined sets, C represents a set of constants, A denotes a set of axioms that S and C must obey, and T indicates a set of theorems.

Definition 2: Machine structure is composed of a tuple denoted as (V, I, T, VA, E), in which V and I represent as the states of the system and set of predicates that determine the characteristics of variables, respectively. The set of predicates in I should be maintained by each reachable state of the system. The set of theorems that need to be proved for the machine is denoted by T. Each machine has some convergent events that need to be defined and denoted by VA. The behavior of the system is model by events denoted by E. The state set or before-after predicate is associated with an event and consists of two major parts: guards that define the preconditions for each event compilation and behavior, which allocates variables values.(Robinson, 2010) elaborates the summary of the Event-B notation:

The following is the most basic type of an event:

$any\ P\ where\ Gu(P, Vr)\ then\ Ac(P, Vr)$

Where the collection of parameters represented by P, variables set is defined as Vr, aggregation of some guards and actions is denoted as $Gu(P, Vr)$ and $Ac(P, Vr)$ respectively. The guard and parameters are non-compulsory and can be omitted to simplify the event. Actions are made up of multiple tasks that are meant to happen at the same time. A before-after predicate can be used to define the change of variable for each assignment.

The Rodin (Abrial et al., 2010; Jastram, Michael, 2014) is used to create and verify the Event-BModels; verification is done either automatically or manually called proof obligation. Further, it provides plugin architecture which makes it more configurable and extensible (Romanovsky & Thomas, 2013). The ProB (Clark et al.,

2016; Leuschel & Butler, 2008) plugin, a tool used for checking and animation of a model

4.2 Model refinement

Refinement is a magnificent modeling method that helps to construct a step-by-step model (El Mimouni & Bouhdadi, 2018). By using the refinement process, both machine and context can be expanded to design a concrete model. Context holds the static information and can extend to more than one context. A machine is a dynamic part of refinement. It can only refine one prevailing machine, but it can see numerous contexts. The abstract variables can be replaced by concrete ones and refine abstract events.

An event can be maintained by alienating numerous concrete events. In another way, an event can be refined in another event by adding new actions and guards. To guarantee the refinement are made correctly, there is a need for some proof obligation that ensure (a) When the concrete event, known as guard strengthening, is activated, the abstract event is enabled as well, (b) for the variables of the concrete and abstract machines the gluing invariants are preserved. (c) Every activity in the abstract event stimulates the concrete event that corresponds to it. The following description shows the created rules for the proof obligations.

Definition 3: let consider ε_1 , ε_2 are the events and ma_1, ma_2 are the machines $\ni\ \varepsilon_1 \in ma_1,\ \varepsilon_2 \in ma_2$. if ma_2 refines ma_1 then ε_2 also refines ε_1. Furthermore, set of axioms seen to ma_2 is denoted by A_x , set of theorems and invariants are defined in ma_1 and ma_2 is represented by T and I respectively. Guards of ε_2 is represented by Gu, whereas the gluing invariant defined in ε_2 and ma_2 is denoted by *J.* An action of ε_1 and ε_2 know as before-after predicate is represented by BA_p1 and BA_p2 which is used by abstract variable and concrete behavior.

Then it is necessary to prove the conjunction of ε_1 guards is denoted by $A_x, T, I, Gu \vdash grd$; whereas J_m is the modified J

$$A_x, T, I, Gu \vdash J_m;$$

$$A_x, T, I, Gu, BA_p2 \vdash BA_p1;$$

4.3 Overview of Cluster Based Flooding

4.3.1 Informal description

The fundamental idea of CBF is to segment the entire network into different routing clusters. Intralayer clustering (IALC) is employed for proactive maintenance on the sensor layer, with an assistance of route query exchange and update query packets. The MDP-level MAC introduces route updates to IALC for broken or established links among the directly connected neighbor mote. Neighbor-motes are defined to be directly connected and sharing a communication link (called one mote away). The MDP-level (MAC) media access control protocol identifies mote's neighbor. At the same time, interlayer clustering (IELC) reactively transfers route query packets to motes on the network layer. These packets exist outside of the mote's cluster through query-reply packets. IELC employs a broadcast delivery service in transmitting the routing queries to its border or peripheral motes. IELC keeps updated route clustering information of peripheral motes using IALC tables. Finally, this information can determine whether the query for destination mote signifies to their cluster.

IoT devices are more vulnerable to redundant and unwanted queries, which may disrupt data transmission, causing them to delay, require more bandwidth and energy to transmit the query to the destination, which may reduce the QoS of IoT Network in terms of energy consumption, cost, delays, and network throughput.

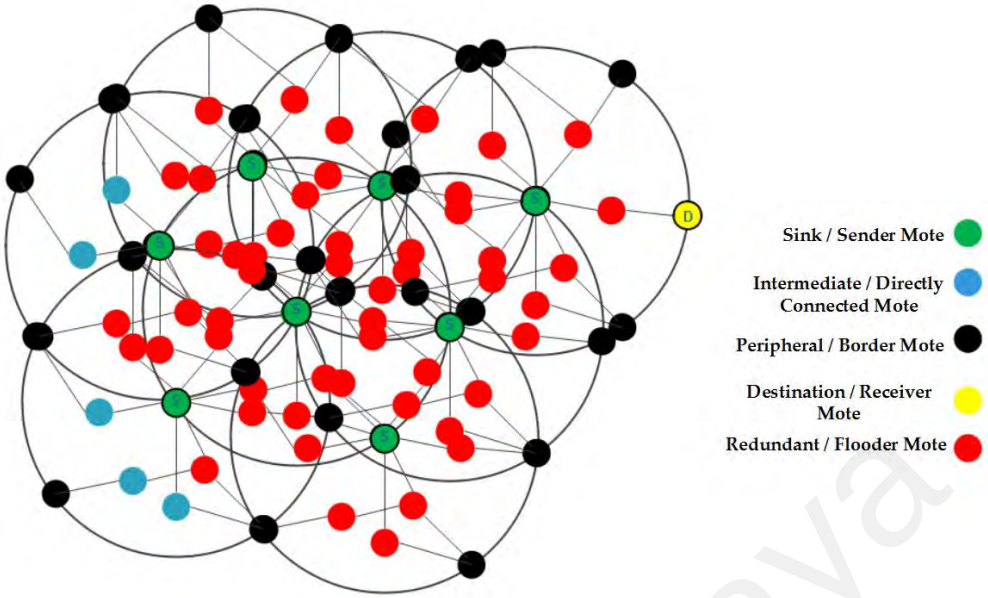


Figure 4.1: Flooded IoT Network During Exchange of Queries

In Figure 4.1 sink / Sender mote searching for target / destination mote and target send data back to sender node in disordered way. Intermediate motes rebroadcast the query automatically which leads to flood the whole network. Generate heavy network traffic and redundant queries by utilizing excessive energy and bandwidth which may leads to more delay and overall degradation of QoS.

4.3.2 System Requirements

During the development process, the sequence number is used to monitor a node's link state history and avoid outdated link-state information. The reactive component ensures the loop freedom of newly discovered query routes. Furthermore, the whole network has a uniform cluster radius. CBF immediately becomes a reactive element if the cluster width is one hop. To concentrate on the unique features of the CBF, the special interest was on identifying the general situation in which the cluster radius is larger than one hop.

The main system requirements are as follows:

REQ-1: Cluster-Based Flooding mechanism is a network routing mechanism that uses a hybrid technique. Its goal is to analyze a new route network on request in a dynamic network environment using local cluster information.

REQ-2: If a valid path exists in the whole network, a route query between two distinct nodes will eventually be discovered.

The REQ-1 explains the protocol's goal. Since a communication network can vary dramatically, a linked connection can go down after a couple of seconds. As a result, when the query source gets the discovered path, it can be invalid. However, if there is a path between the source and the destination in the network topology, the route between them can be discovered eventually.

REQ-2 indicates this requirement. Eventually, this Chapter demonstrate the definition of the correct route.

Definition 4: (Accurate Route) let consider $\vartheta_1, \ldots\ldots, \vartheta_{k\prime}$, where k' states as a positive natural number and k is greater than one and network of random distinctive nodes. The query route that has been discovered is denoted by $route_{query} = \{\vartheta_1 \longmapsto \vartheta_2, \ldots., \vartheta_{k\prime-1} \longmapsto \vartheta_{k\prime}\}$ if it satisfies the criteria for the source s and destination d, it is appropriate.

- s and t represented as motes in the network
- For each $\vartheta_i \longmapsto \vartheta_{i+1} \in route$ with respect to $i \in 1 \ldots k' - 1, \vartheta_i, \vartheta_{i+1}$ are motes which belongs to the network.
- The route does not have a loop.
- Complete path from s to t is known as route_query.

REQ-3: Each node can acquire the connection status of its nearest neighbors on a regular basis and then exchange such knowledge with the other nodes.

REQ-4: Clusters restrict the range of connection state updates.

REQ-5: Each mote in the network cluster proactively maintains the updated topological map. As a result, all nodes have routes to the destinations within the cluster.

The mote collaborates with the nearest neighbor information given by the MDP on a periodical basis. Then it broadcasts the state of its links to the rest of the cluster. According to REQ-3 to REQ-5, every mote sees the local cluster from a new perspective. Since the nodes' perspective can differ from the actual network topology, these requirements ensure that nodes' views in a real-time manner.

REQ-6: The discovery of route query depends on the cluster. If the existence of the route is not present in the route towards the destination in its routing table, the source mote then begins the discovery of the route to the destination. To broadcast the service, the source mote sends the path query request towards the subset of neighbors. It is not necessary that all neighbors receive the same request.

REQ-7: The routing query request will only obtain by the designated destinations decided by the last forwarded motes

REQ-8: Depending on whether the destination is inside the cluster radius, the expected recipient's reaction may be one of the following: (a) It sends a route query response to the source within the cluster, (b) Alternatively, it adds the address to an aggregate route and send flood broadcasts the query request.

REQ-9: With the QoS-enabled cluster-based query control scheme, a route query request can be driven away from the source of query and the regions included by request. As per the query control scheme, a node marks its cluster as reached if it entertains the

request or the reply. After that, it is insignificant to reprocess the request. The next requirement relates the routing update.

REQ-10: Each mote rebroadcasts information or relays query route reply to packets if there are any changes within its cluster. Once the source mote found an authentic route query message, it notifies the directly connected motes about the newly found route in real-time scenarios.

4.4 Environment assumptions

Before proceeding towards the formal development, consider the following CBF network assumptions:

Assumptions

ENV-1: n number of sensor motes are randomly deployed. All motes carry the same capacity in terms of functionality. every mote equipped with an IP address for the sake to work as sensor gateway to exchange query messages. For all connected nodes, mode of communication was adopted as single and multi-hop. Every mote can instigate flooding. In case of some activity detection, the flooder begins to interrupt through the link. There is similarity between the capabilities of the flooder and normal motes. In addition, the flooder mote can also generate redundant query messages (i.e., random flooding queries).

ENV-2: Intermediate status for pairs of different motes might be up or down by using bidirectional links.

ENV-3: Every mote m is aware of broken or uplinks in the network.

ENV-4: In case of null activity for a long time in the network, every mote must know the updated topology map of its surroundings.

According to Env-2, it is assumed that sender and receiver motes can communicate simultaneously, and the links are bidirectional in the network. So, it can broadcast route query replies of discovered routes to the source mote. If mote m and n are directly connected, it might indicate directed links from m → n and n → m and considered up in the network environment.

According to ENV-3, every mote can sense its external links. It is not required that a mote detects the network changes immediately.

Based on ENV-4, if the network is static for a long time, the system will consider being in a stable state, and every mote is aware of their clusters topology.

4.5 Formal Development

This Section describes the formal development of Cluster Based Flooding (CBF). Initially, the refinement strategy is elaborated for analysis; with the help of a step-by-step refinement process, a model formalization is derived from achieving the correctness for REQ1 and REQ2. Model validation is presented at the end of the chapter.

Model initialization: The dynamic network architecture is built using this model.

Refinement 1: In this stage, abstract update events for the routing and link state table are introduced. Furthermore, the stability of the system is considered in a quiescent way.

Refinement 2: This refinement briefly formalizes links state updates and periodically links state broadcasting of each mote.

Refinement 3: The model for the cluster radius is constructed based on refinement 2 using a uniform cluster radius. sequence numbers are utilized to keep track of each query packets and to avoid the processing of old link information.

Refinement 4: Variables are used to keep track of all connected links' distributed behavior and record their transmitted information.

Refinement 5: This refinement analyzes the route query request phase in detail, initially, formalization for border-casting is carried out without any query control mechanism.

Refinement 6: This refinement develops concrete events that indicate that a route request has been received and handled by the intended recipient.

Refinement 7: In this refinement, a cluster-based query control mechanism is considered during the route query process.

Refinement 8: This refinement modeled the route reply process. A mote updates its routing table in response to an IALC notification or a received route reply. Routing table updates for the cluster are also refined in this step. Modeling the interactions between the system and its environment is possible with Event-B formalism. The initial model formalizes the dynamic network environment.

Such as refinement 2 considers environment 3 and Req 3, and req6 and req7 are ensured in refinement 5. According to definition 4, points 1 and 2 were proved in refinement 1, discovered routes validity and freedom of loop are ensured in refinement 5 and 8, respectively. The final model ensured the correctness of discovered route queries. The Req-1 is impartially general and include in each step of refinement. Finally, Req-1 and 2 completes in final refinement.

4.5.1 Environmental Modeling

In the initial context, a carrier set motes were introduced, which represent the collection of all motes in the networks. Based on Environment 1, Motes are axiomatized as a finite set.

The following theorem follows a relation established over finite sets (thm1). It will help to prove the tasks simpler.

$axm1$: $finite(Motes)$

$thm1$: $\forall a, b, f. a \in \mathbb{P}(Motes) \wedge b \in \mathbb{P}(Motes) \wedge f \in a \leftrightarrow b$

$\Longrightarrow (finite(a) \wedge finite(b) \Longrightarrow finite(f))$

A variable *Neighbor_Link* is defined to record the uplink information. The D_Neighbor_Link variable represents the set of links that are presently down and previously up. It can be seen from thm1 and Neighbor_Link that the network topology has a finite number of connected links. As a result, a route query request can be propagated through the entire network. Three events have been created for the purpose to formalize the changes in network topology: Link_New_Add, Link_Brocken_Add, Link_Remove. According to the first two events, initially, the two different motes are connected, and subsequently, the associated links are added to Neighbor_Link. Link_Remove clears the invalid links from Neighbor_Link, which were up links at some stage in the past.

$Link\_New\_Add$
$\textbf{any}\ Mote\ 1\ Mote\ 2\ \textbf{where}$
$Mote\ 1 \in Motes$
$Mote\ 2 \in Motes$
$Mote\ 1 \neq Mote2$
$Mote\ 1 \mapsto Mote2 \notin Neighbor\_link$
$\wedge Mote\ 2 \mapsto Mote1 \notin Neighbor\_link$
$Mote\ 1 \mapsto Mote2 \notin D\_Neighbor\_link$
$Mote\ 1 \mapsto Mote2 \in Neighbor\_link$
$\wedge Mote\ 2 \mapsto Mote1 \in D\_Neighbor\_link$
then
$Neighbor\_link \coloneqq Neighbor\_linkU$
$\{Mote\ 1 \mapsto Mote2, Mote\ 2 \mapsto Mote1\}$

$Link\_Broken\_Add$
$\boldsymbol{any}\ Mote\ 1\ Mote\ 2\ \boldsymbol{where}$
$Mote\ 1 \in Motes$
$Mote\ 2 \in Motes$
$Mote\ 1 \neq Mote2$
$Mote\ 1 \mapsto Mote2 \notin Neighbor\_link$
$\wedge Mote\ 2 \mapsto Mote1 \notin Neighbor\_link$
$Mote\ 1 \mapsto Mote2 \notin D\_Neighbor\_link$
$Mote\ 1 \mapsto Mote2 \in Neighbor\_link$
$\wedge Mote\ 2 \mapsto Mote1 \in D\_Neighbor\_link$
then
$Neighbor\_link := Neighbor\_linkU$
$\{Mote\ 1 \mapsto Mote2, Mote\ 2 \mapsto Mote1\}$
$D\_Neighbor\_link := D\_Neighbor\_link \setminus$
$\{Mote\ 1 \mapsto Mote2, Mote\ 2 \mapsto Mote1\}$

Link_Remove requires the deleted links to be located in the Neighbor_Link, and then add these links to the set D_Neighbor_Link.

$Link\_Remove$
$\boldsymbol{any}\ Mote\ 1\ Mote\ 2\ \boldsymbol{where}$
$Mote\ 1 \in Motes$
$Mote\ 2 \in Motes$
$Mote\ 1 \neq Mote2$
$Mote\ 1 \mapsto Mote2 \in Neighbor\_link$
$\wedge Mote\ 2 \mapsto Mote1 \in Neighbor\_link$
then
$Neighbor\_link := Neighbor\_link \setminus$
$\{Mote\ 1 \mapsto Mote2, Mote\ 2 \mapsto Mote1\}$
$D\_Neighbor\_link := D\_Neighbor\_linkU$
$\{Mote\ 1 \mapsto Mote2, Mote\ 2 \mapsto Mote1\}$

Some Invariants has added to establish Env 2. Neighbor_Link is made up of unique motes for each link (inv3). Furthermore, in a case Neighbor_*Link* hold a dedicated

link i.e mote1 → mote2, then the opposite of this link mote1 → mote2 must be resides in the *Neighbor_Link* (inv4). *Neighbor_Link* and *D_Neighbor_Link* are separate (inv5).

$inv1$:	$Neighbor\_link \in Motes \leftrightarrow Motes$
$inv2$:	$D\_Neighbor\_link \in Motes \leftrightarrow Motes$
$inv3$:	$\forall mote\,1, mote2.\, mote\,1 \in Motes \wedge mote2 \in Motes \wedge$
	$mote\,1 \mapsto mote2 \in Neighbor\_link \implies mote\,1 \neq mote2$
$inv4$:	$\forall mote\,1, mote2.\, mote\,1 \in Motes \wedge mote2 \in Motes \wedge$
	$mote\,1 \mapsto mote2 \in Neighbor\_link$
	$\implies mote\,2 \mapsto mote1 \in Neighbor\_link$
$inv5$:	$Neighbor\_link \cap D\_Neighbor\_link = \emptyset$

4.5.2 Formation of Cluster

This development aims to formalize a model according to the assumptions and requirements of the system environment.

First Refinement. In the first refinement, *Cessation* (*closure)* is defined to denote the transitive cessation of link.

$axm1$:	$Cessation \in (Motes \leftrightarrow Motes) \rightarrow (Motes \leftrightarrow Motes)$
$axm2$:	$\forall r.\, r \subseteq Cessation(r)$
$axm3$:	$\forall r.\, Cessation(r); r \subseteq Cessation(r)$
$axm4$:	$\forall r.s.r \subseteq s \wedge s\,;r \subseteq s \implies Cessation(r) \subseteq s$
$axm5$:	$\forall r.\, Cessation(r)\,;\, Cessation(r) \subseteq Cessation(r)$

A variable *Table_Link_State* is used to specify the information of the link-state stored in each mote. The *add_links* and *remove_links* are two disjoint sets used to update

mote's link state table. To model the above case initially, old neighbor information needs to be removed and then add the updated received information of the neighbor.

Links_Update
$\boldsymbol{any}$ $mote$ add_links $remove\_links$ $\boldsymbol{where}$
$mote \in Motes$
$add\_links \in Motes \leftrightarrow Motes$
$remove\_links \in Motes \leftrightarrow Motes$
Then
$Table\_Link\_State(mote) := (Table\_Link\_State(mote) \backslash remove\_links) \cup add\_links$

The mote's routing table of the cluster is the collection of links, and it is specified by the variable *Table_Routing_Cluster*.

$inv1$:	$Table\_Link\_State \in Motes \rightarrow (Motes \leftrightarrow Motes)$
$inv2$:	$Table\_Routing\_Cluster \in Motes \rightarrow (Motes \leftrightarrow Motes)$
$inv3$:	$\forall n, n \in Motes \Rightarrow Table\_Routing\_Cluster(n) \subseteq Motes \times Motes$

Update_Routing_TableAn event *Update_Table_Routing* is defined to model routing update for each mote. If no links are to add or remove, then the mote's routing table remains the same. Some links are intersecting at *Routes_Add* and *Routes_Remove*. So, it is desired to initially eliminate the outdated link before adding the recent routes.

Update_Routing_Table
$\boldsymbol{any}$ $mote$ $Routes\_Add$ $Routes\_Remove$
Where
$mote \in Motes$
$Routes\_Add \in Motes \leftrightarrow Motes$
$Routes\_Remove \in Motes \leftrightarrow Motes$
$\neg(Routes\_Add = \emptyset \wedge Routes\_Remove = \emptyset)$
Then
$Table\_Routing\_Cluster(mote) := (Table\_Routing\_Cluster(mote) \backslash Routes\_Remove) \cup Routes\_Add$

An event named stabilize is defined to denote the system stability state. Once the stabilize event becomes enabled, the state f the system is considered stable. This step formalizes **ENV-4**.

Stabilize
any *motes* ***where***
$motes \in Motes \longrightarrow \mathbb{P}(Motes)$
$\forall m, n.\, m \mapsto n \in Neighbor\_Link \Leftrightarrow m \mapsto n \in Table\_Link\_State(m)$
$\forall m, n.\, m \mapsto n \in Cessation(Neighbor\_Link) \wedge n \in motes(m)$
$\Longrightarrow (\forall x.\, m \mapsto x \in Table\_Link\_State(n) \Leftrightarrow m \mapsto x \in Table\_Link\_State(m)$

Second Refinement: Every node in the IALC shares neighbor information with other nodes and maintains an updated view of its neighbors. This refinement model exchange procedure by defining some constant that classifies the stationary part of the model.

$inv1$:	$InsertionTime \in Motes \longrightarrow ((Motes \times Motes) \nrightarrow \mathbb{N})$
$inv2$:	$Time \in \mathbb{N}_1)$
$inv3$:	$BroadcastTime \in Motes \longrightarrow \mathbb{N}_1)$
$inv4$:	$\forall n.\, n \in Motes \Longrightarrow (\forall l.\, l \in Table\_Link\_State(n))$
	$\Leftrightarrow l \in dom(InsertionTime(n)))$
$inv5$:	$\forall n.\, n \in Motes \Longrightarrow BroadcastTime(n) \leq Time$
$inv6$:	$\forall m, n.\, m \mapsto n \in Table\_Link\_State(m) \Longrightarrow (m \mapsto n \in NeighborLink \vee m \mapsto n$
	$\in DNeighborLink)$
$inv7$:	$\exists m, n.\, m \mapsto n \in Table\_Link\_State(m) \wedge m \mapsto n \notin NeighborLink$
	$\wedge\, m \mapsto n \in DNeighborLink$
	$\Longrightarrow \neg(\forall q, p.\, q \mapsto p \in NeighborLink \Leftrightarrow q \mapsto p \in Table\_Link\_State(q)))$
$inv8$:	$\exists m, n.\, m \mapsto n \notin Table\_Link\_State(m) \wedge m \mapsto n \in NeighborLink$
	$\Longrightarrow \neg(\forall q, p.\, q \mapsto p \in NeighborLink \Leftrightarrow q \mapsto p \in Table\_Link\_State(q)))$

The *Links_Update* abstract is divided into three sections: *Links_Obtain*, *Links_Transfer*, and *Links_Refresh*. *Links_Obtain* is responsible for the link information of the mote's neighbor. Every mote must know the link information of its neighbors. *Add_links* in this event is the set of links between motes and their newly discovered neighbor. *Remove_links* is responsible for the collection of links among motes and their invalid connected neighbors. Rest is defined to keep track of residual link insertion timing to the state table of motes.

Links_Obtain refines Links_Update
$any\ mote\ add\_links\ remove\_links\ rest$
$where$
$\oplus\ add\_links = \{p \mapsto m \mid p = mote \wedge p \mapsto m \in Neighbor\_Link$
$\wedge\ p \mapsto m \notin Table\_Link\_State(p)\}$
$\oplus\ remove\_links = \{p \mapsto m \mid p = mote \wedge p \mapsto m \notin Neighbor\_Link$
$\wedge\ p \mapsto m \in Table\_Link\_State(p)\}$
$\oplus\ Time - BroadcastTime(mote) \geq period$
$\oplus\ rest = remove\_links \lhd InsertionTime(mote)$
then
$\oplus\ InsertionTime(mote) \coloneqq rest$ $\leftarrow (\left(\left(\{mote\} \lhd dom(rest)\right) \cup add\_links\right) \times \{Time\})$
$\oplus\ BroadcastTime(mote) \coloneqq Time$

Links_Transfer refines Links_Update
$\boldsymbol{any}\ sender\ reciever\ transfer\ mote\ recieved\ links$
$add\_links\ remove\_links$
where
$\ominus\ mote \in Motes$
$\oplus\ sender \in Motes\ \wedge reciever\ \in Motes \wedge transfer\_Mote \in Motes$
$\oplus\ sender \neq reciever\ \wedge\ reciever\ \neq\ transferMote$
$\oplus\ sender \mapsto reciever \in Table\_Link\_State(sender)$
$\oplus\ recieve\_links \in Motes \leftrightarrow Motes \wedge dom(recieve_{links}) = \{transfer\_Mote\}$
$\oplus\ add\_links = recieve\_links$
$\oplus\ remove\_links = \{transfer\_Mote\} \lhd Table\_Link\_State(reciever)$

With
$mote = reciever$
Then
$\ominus\ Table\_Link\_State(mote)$ $:= Table\_Link\_State(mote) \backslash remove\_links$ $\cup\ add\_links$
$\oplus\ Table\_Link\_State(reciever)$ $:= Table\_Link\_State(reciever)$ $\backslash remove\_links \cup add\_links$
$\oplus\ InsertionTime(reciever)$ $:= (remove\_links \lhd InsertionTime(reciever))$ $\cup\ (add\_links \times \{Time\})$

The old link routing information of *Links_Transfer* recorded by the *remove_link* needs to be removed before taking the record of the updated received link recorded by the *add_links.*

Links_Refresh refines Links_Update
$\boldsymbol{any}\ mote\ old\_links$
Where
$\oplus\ oldlinks = \{x \mapsto y \mid x \mapsto y \in Table\_Link\_State(mote) \wedge x \neq mote \wedge$
$(Time - InsertionTime(mote)(x \mapsto y) \geq lifetime)\}$
With
$add\_links = \emptyset$
$remove\_links = old\_links$
Then
$\ominus\ Table\_Link\_State(mote)$ $:= (Table\_Link\_State(mote) \backslash remove\_links) \cup add\_links$
$\oplus\ Table\_Link\_State(mote) := Table\_Link\_State(mote) \backslash old\_links$
$\oplus\ InsertionTime(mote) := old\_links \lhd InsertionTime(mote)$

Links_Refresh is the event responsible for updating the link state table without considering the link sources that have quit from the mote's cluster radius. A new event, Clock_time, is defined to model the time growth process. The **REQ-3** and **ENV-3** are implemented in this step.

Third Refinement: The emphasis of this refinement is on the design of the cluster radius. Instead of the entire network, each mote in CBF keeps cluster information within its

cluster radius. A constant *cluster radius* is used to represent cluster radius, and its value is > 1. A *TTL* variable is defined to store the value of *TTL*. While a mote broadcasts its connected neighbor information, the *TTL* value is set as cluster radius – 1. This broadcast terminates once the *TTL* value reaches 0. A variable *Seq_Num* is defined to track the history of the link-state packet for every mote.

$inv1$:	$TTL \in Motes \rightarrow (Motes \rightarrow \mathbb{N})$
$inv2$:	$Seq\_Num \in Motes \rightarrow (Motes \rightarrow \mathbb{N})$
$inv3$:	$\forall m, p. p \in dom(Table\_Link\_State_m) \Rightarrow Seq\_Num(m)(p) > 0$
$inv4$:	$\forall m, p. p \in dom(Table\_Link\_State) \wedge m \mapsto n \in Cessation$
	$\big(Table\_Link\_State(m)\big) \wedge (\exists x. m \mapsto n \in Table\_Link\_State(m) \wedge$
	$\neg m \mapsto n \in Table\_Link\_State(n))$
	$\Rightarrow \neg(\forall x. m \mapsto n \in Table\_Link\_State(m) \Leftrightarrow m \mapsto n \in Table\_Link\_State(n))$

An event *discard_Links* is used to discards the received mote link-state information having a smaller sequence number. The mote uses transfer_Links to process this packet and record the sequence number to reduce the TTL value.

Links_Transfer refines Links_transfer
$any\ sender\ reciever\ transfer\_Mote\ recieve\_links$
$add\_links\ remove\_links\ recieved\_TTL\ recieved\_Seq\_Num$
$where$
$\bigoplus\ recieved\_TTL > 0 \wedge recieved\_Seq\_Num > 0$
$\bigoplus Seq\_Num(reciever)(transfere\_Mote) \leq recieved\_Seq\_Num$
then
$\bigoplus TTL(reciever) \coloneqq TTL(reciever) \leftarrow$
$\{transfer\_Mote \mapsto recieved\_TTL - 1\}$
$\bigoplus Seq\_Num(reciever) \coloneqq Seq\_Num(reciever) \leftarrow$
$\{transfer\_Mote \mapsto recieved\_Seq\_Num\}$

For the Stabilize, Let add two Conditions:

(1) $Motes = \lambda x.x \in Motes | dom\big(Table\_Link\_State(x)\big)$,
(2) $\forall x, y.x \in Motes \ \wedge y \ \in motes(x)$
    i. $\Rightarrow x \mapsto y \in Cessation\big(Table\_Link\_State(x)\big)$,

To describe the scope of the link-state propagation. The stabilization property of the system is derived as a theorem.

Theorem 1 If the system is stable, and there exists a path in the network topology between node m and node n, and n is in the routing cluster of m, then there exists a route from m to n in m's link state table. Let Guards be the conjunction of all guards of stabilizing event. Then, the formalization of the statement is

$$Guards \Rightarrow (\forall m, n.m \mapsto n \in Cessation(Neighbor\_Link)$$

$$\leftrightarrows \wedge n \ \in ran(Table\_Link\_State(m))$$

$$\Rightarrow m \mapsto n \ \in Cessation\big(Table\_Link\_State(m)\big).$$

As a result of this refinement, each mote has the updated cluster radius information. This refinement ensures the Req 4 and Req 5.

Fourth Refinement: *Transmitted_Link* is defined as stopping a mote from retrieving other mote's private information. *Transmitted _Seq_Num* and *Transmitted _TTL is defined to stipulate the sequence number and TTL values being transmitted.* A variable flag indicates whether a mote can receive information of link state from another mote.

$inv1$:	$Transmitted\_Link \in (Motes \times Motes) \rightarrow (Motes \rightarrow$
	$(Motes \leftrightarrow Motes))$
$inv2$:	$Transmitted\_SeqNum \in (Motes \times Motes) \rightarrow (Motes \rightarrow \mathbb{N})$
$inv3$:	$Transmitted\_TTL \in (Motes \times Motes) \rightarrow (Motes \rightarrow \mathbb{N})$
$inv4$:	$Flag \in Motes \rightarrow (Motes \rightarrow BOOL)$

$inv5$:	$\forall m, n, p. Transmitted\_Link(m \mapsto n)(p) \neq \emptyset \wedge n \neq p$
	$\Rightarrow m \mapsto n \in Table\_Link\_State(m)$
$inv6$:	$\forall m, n, p. Transmitted\_Link(m \mapsto n)(p) \neq \emptyset \wedge n \neq p$
	$\Rightarrow dom\big(Transmitted\_Link(m \mapsto n)(p)\big) = \{p\}$

Further, send_Links, receive_Links, discard_Links, and cancel_SendingLinks are adopted to model the dispersed behavior of packet transmission. The Send_links is enabled when the transfer mote link information is empty, and its TTL is > 0. Or else the mote is just required to refresh the flag by *Cancel_Sending_Links.*

Send_Links
$any\ sender\ transfer\_MoteLinks$
$where$
$sender \in Motes$
$transferMote \in Motes$
$transferMote \in dom(Table\_Link\_State(sender))$
$links = \{x \mapsto y \mid x = sender \wedge x \mapsto y \in Table\_Link\_State(x)\}$
$\forall l.\, l \in links \Rightarrow Transmitted\_links(l)(transfer\_Mote) = \emptyset$
$TTL(sender)(transfer\_Mote) > 0$
then
$Transmitted\_Link := Transmitted\_Link \leftarrow (\lambda l.\, l \in links\|$
$TransmittedLink \leftarrow \{transferMote \mapsto$
$\{transfer\_Mote\} \lhd Table\_Link\_State(sender)\})$
$Transmitted\_SeqNum := Transmitted\_SeqNum \leftarrow (\lambda l.\, l \in links\|$
$Transmitted\_SeqNum(l) \leftarrow \{transfer\_Mote \mapsto Seq\_Num(sender)(transfer\_Mote)\})$
$Transmitted\_TTL := Transmitted\ TTL \leftarrow (\lambda l.\, l \in links\|$
$Transmitted\_TTL(l) \leftarrow \{transfer\_Mote \mapsto TTL(sender) := Flag(sender) \leftarrow$
$\{transfer\_Mote \mapsto TRUE\}$

Receive_Links and refining *Transfer_Links* is used to stop the receiver mote from accessing the sender's private information and direct the receiver to deal with the required link-state information. This link information can be obtained through connected

links *sender_mote → receiver_mote.* The point that is mentioned in with section denotes the assignments of this received information. Receive_Links and Discard_Links must reset the link information that is currently being transmitted *sender_mote → receiver_mote*.

Receive_Links refines Transfer_Links
$any\ sender\ reciever\ transfer\_Mote\ add\_links\ remove\_links$
$where$
$\ominus\ sender \mapsto reciever\ \in Table\_Link\_State(sender)$
$\ominus\ recieve\_links\ \in Motes\ \leftrightarrow Motes\ \wedge dom(recieve_{links}) = \{transfer\_Mote\}$
$\ominus\ add\_links = recieve\_links$
$\ominus\ Seq\_Num(reciever)(transfer_{Mote}) \leq recieved\_SeqNum$
$\ominus\ recieved\_TTL > 0 \wedge recieved\_SeqNum > 0$
$\oplus\ add\_links \coloneqq Transmitted\_Link(sender \mapsto reciever)(transfer\_Mote)$
$\oplus\ Transmitted\_Link(sender \mapsto reciever)\ (transfer\_Mote) \neq \emptyset$
$\oplus\ Transmitted\_Link(sender \mapsto reciever)(transfer\_Mote) > 0$
$\wedge Transmitted\_SeqNum(sender \mapsto reciever)(transfer\_Mote) > 0$
$\oplus\ Flag(reciever)(transferMote) = TRUE$
$\oplus\ seq\_Num(reciever)(transfer\_Mote) \leq$
$Transmitted\_SeqNum(sender \mapsto reciever)(transfer\_Mote)$
with
$recieve\_links = Transmitted\_Link(sender \mapsto reciever)(transfer\_Mote)$
$recieved\_TTL = Transmitted\_TTL(sender \mapsto reciever)(transfer\_Mote)$
$recieved\_SeqNum = Transmitted\_SeqNum(sender \mapsto reciever)(transfer\_Mote)$
then
$\oplus\ Transmitted\_Link(sender \mapsto reciever) \coloneqq Transmitted\_Link(sender \mapsto reciever)$
$\leftarrow \{transfer\_Mote \mapsto \emptyset\}$
$\oplus\ Transmitted\_SeqNum(sender \mapsto reciever) \coloneqq$
$Transmitted\_SeqNum(sender \mapsto reciever) \leftarrow \{transfer\_Mote \mapsto \emptyset\}$
$\oplus\ Transmitted\_TTL(sender \mapsto reciever) \coloneqq$
$Transmitted\_TTL(sender \mapsto reciever) \leftarrow \{transfer\_Mote \mapsto \emptyset\}$
$\oplus\ Flag(reciever) \coloneqq Flag(reciever)\ \leftarrow \{transfer\_Mote \mapsto FALSE\}$

$inv7$:	$\forall m, n, p. Transmitted\_Link(m \mapsto n)(p) \neq \emptyset \wedge n \neq p$
	$\Rightarrow Transmitted\_TTL(m \mapsto n)(p) > 0$
$inv8$:	$\forall l, p. l \in Motes \times Motes \wedge Transmitted\_Link(l)(p) = \emptyset$
	$\Rightarrow Transmitted\_TTL(l)(p) = 0$
$inv9$:	$\forall m, n, p. Transmitted\_Link(m \mapsto n)(p) \neq \emptyset \wedge n \neq p$
	$\Rightarrow Transmitted\_SeqNum(m \mapsto n)(p) > 0$
$inv10$:	$\forall l, p. l \in Motes \times Motes \wedge Transmitted\_Link(l)(p) = \emptyset$
	$\Rightarrow Transmitted\_SeqNum(m \mapsto n)(p) = 0$

4.5.3 Route Query Discovery Process

In the previous refinements, cluster radius has been formalized. The below steps elaborate on the designing of route query request and route query reply to phases in the dynamic environment. By utilizing the bordercast service for route query requests following points is vital to consider.

- The design and development of bordercast tree in order to determine the targeted forwarding motes.
- The records of the routes that have been accumulated.
- Selecting routes in the cluster radius of a motes with its destination.
- The cluster-based query control mechanism which is used to stay away from the route query request from the already covered cluster

In the next refinement, the first and second points are defined. In subsequent refinements, other points are eventually formalized.

Fifth Refinement: Five new variables are defined in this refinement. Variable *Route_Request* is defined to represent the set of route query requests. *Intended_Neighbor* is defined to specify that the bordercast mote must know the set of forwarding neighbors for sending or receiving route query request.

The *Accumulated_Path* variable tracks the accumulated paths with query requests. Two variables are defined to describe the corresponding being transmitted data to understand the system's distributed behavior. *Transmitted_Tag* is used to decide if a neighbor is an intended recipient, and *Transmitted_Path* describes the cumulative routes being transmitted.

$inv1$:	$Route\_Request \in Motes \leftrightarrow Motes$
$inv2$:	$Intended\_Neighbor \in (Motes \times (Motes \times Motes))$
	$\rightarrow \mathbb{P}(Motes)$
$inv3$:	$Accumulated\_Path \in (Motes \times (Motes \times Motes)$
	$\nrightarrow (Motes \leftrightarrow Motes))$
$inv4$:	$Transmitted\_Tag \in (Motes \times Motes) \longrightarrow$
	$((Motes \times Motes) \longrightarrow \mathbb{N})$
$inv5$:	$Transmitted\_Path \in (Motes \times Motes) \longrightarrow$
	$((Motes \times Motes) \longrightarrow (Motes \times Motes))$
$inv6$:	$\forall s, d. s \in Motes \wedge d \in Motes \wedge s \mapsto d \in Route\_Request$
	$\Longrightarrow s \neq d$
$inv7$:	$\forall s, n. s \in Motes \wedge n \in Motes \wedge s \neq n \Longrightarrow (\forall r. s = prj1(r) \wedge$
	$r \in dom(Accumulated\_Path(n))$
	$\Longrightarrow r \in dom(Accumulated\_Path(s)))$

Though the query control mechanism is not analyzed in this phase, it is to keep in mind that a mote must not deal with a processed route query request. The route query request process is initiated. When there is no path to the destination in the cluster routing table of the source mote, *the Source_Forward_Request* event is defined to model this behavior. A variable mote represents the set of forwarding neighbors in the sender's bordercast tree.

Each *mote* has at least one route within the source cluster radius to one of the peripheral motes and must be a nonempty set. The source then places the request on the

determined links with an empty path and specifies the recipients with update *Transmitted_Tag*.

Source_Forward_Request
$any\ source\ destination\ request\ motes\ links$
$where$
$source \in Motes$
$destination \in Motes$
$request = source \mapsto destination$
$request \notin RouteRequest$
$source \neq destination$
$source \mapsto destination \notin closure(RoutingTable(source))$
$motes = \{q \mid source \mapsto q \in Table\_Link\_State(source \wedge$
$(\exists p, c.c \subseteq Table\_Link\_State(source) \wedge q \mapsto p \in closure(c) \wedge$
$card(c) = cluster\_Radius - 1 \wedge$
$(\forall s.s \subseteq Table\_Link\_State(source) \wedge q \mapsto p \in closure(s)$
$\wedge card(s) \geq card(c)))\}$
$motes \neq \emptyset$
$links = \{p \mapsto q \mid p = source \wedge q \in motes\}$
$\forall l.l \in links \Rightarrow Transmitted\_Tag(l)(request) = 0$
$request \notin dom(Accumulated\_path(source))$
Then
$Route\_Request := Route\_Request \cup \{request\}$
$Accumulated\_path := Accumulated\_path \leftarrow \{source \mapsto$
$(Accumulated\_path(source) \leftarrow \{request \mapsto \emptyset\})\}$
$Intended\_Neighbor := Intended\_Neighbor \leftarrow$
$\{(source \mapsto request) \mapsto motes\}$
$Transmitted\_Tag := Transmitted\_Tag \leftarrow$
$\lambda l.l \in links \mid Transmitted\_Tag \leftarrow \{request \mapsto 1\})$
$Transmitted\_Path := Transmitted\_Path \leftarrow$
$\lambda l.l \in links \mid Transmitted\_Path \leftarrow \{request \mapsto \emptyset\})$

If the *intended_forwarding_neighbours* aren't empty, the mote will use the *bordercast_Request* event to bordercast the request. They are both abstract.

$inv8$:	$\forall n, m, r.n \neq m \wedge Transmitted\_Path(n \mapsto m)(r) \neq \emptyset$
	$\Rightarrow r \in dom(Accumulated\_Path(n))$
$inv9$:	$\forall n, m, r, path.n \neq m \wedge m \in Intended\_Neighbor(n \mapsto r)$
	$\Rightarrow (r \mapsto path \in Accumulated\_Path(n)$
	$\wedge Transmitted\_Tag(n \mapsto m)(r) \neq 0$
	$\Rightarrow TransmittedPath(n \mapsto m)(r) = path)$

$inv10$:	$\forall n, m, r. n \neq m \wedge Transmitted\_Tag(n \mapsto m)(r) \neq \emptyset$
	$\Rightarrow m \in Intended\_Neighbor(n \mapsto r)$
$inv11$:	$\forall l, r, l \in Motes \times Motes \wedge Transmitted\_Tag(l)(r) = 0$
	$\Rightarrow Transmitted\_Path(l)(r) = \emptyset$
$inv12$:	$\forall n, r. n \in Mote \wedge r \in Route\_Request \wedge r \in$
	$dom(Accumulated\_Path(n)) \Rightarrow Accumulated\_Path(n)(r)$
	$\in Motes \leftrightarrow Motes$
$inv13$:	$\forall n, s, t. n \in Motes \wedge s \mapsto t \in Motes \times Motes \wedge$
	$s \mapsto t \in dom(Accumulated\_Path(n))$
	$\Rightarrow (\forall m, path. path = Accumulated\_path(n)(s \mapsto t)$
	$\wedge m \in dom(path) \Rightarrow \neg(m \mapsto m \in Cessation(path))))$

Receive_Request need to be refined to more concrete events. This refinement ensures the **Req 6 and Req 7** of system requirements.

Receive_Request
$any\ sender\ reciever\ request\ motes\ path$
$where$
$sender \in Motes \wedge reciever \in Motes \wedge sender \neq reciever$
$request \in Route\_Request$
$Transmitted\_Tag(sender \mapsto reciever)(request) = 1$
$request \notin dom(Accumulated\_Path(reciever))$
$motes \in \mathbb{P}(Motes)$
$path \in Motes \leftrightarrow Motes \wedge sender \mapsto reciever \in path$
$\forall n. n \in dom(path) \Rightarrow \neg n \mapsto n \in Cessation\{path\}$
$then$
$Accumulated\_Path(reciever) :=$
$Accumulated\_Path(reciever) \leftarrow \{request \mapsto path\}$
$Intended\_Neighbor := Intended\_Neighbor$
$\leftarrow \{(reciever \mapsto request) \mapsto motes\}$
$Transmitted\_Tag(sender \mapsto reciever) :=$
$Transmitted\_Tag(sender \mapsto reciever) \leftarrow \{request \mapsto 0\}$
$Transmitted\_Path(sender \mapsto reciever) :=$
$Transmitted\_Tag(sender \mapsto reciever) \leftarrow \{request \mapsto \emptyset\}$

Sixth Refinement: Receive Request has been split into two events in this refinement, depending on if the recipient has legitimate routes to the destinations within its cluster

radius. One is *Receive_Request_NoRoute*, which indicates that the destination is not within the cluster radius of the receiver. The mote must first create the bordercast tree with the calculated recipients denoted by *motes* before forwarding this request with the updated *path*.

Receive_Request_No_Route refine Receive_Request ∃ ∀≠ ∈ ∉∧⊆≥ ↦⇒↔∪ ℙ ⨁ ⊖⊖ ∀∅
$any\ sender\ reciever\ request\ motes\ path\ destination$
$where$
$\ominus\ motes \in\ \mathbb{P}(Motes)$
$\ominus\ path \in Motes \leftrightarrow Motes \wedge sender \mapsto reciever\ \in path$
$\oplus\ destination = prj2(request)$
$\oplus\ reciever \mapsto\ destination \notin$
$closure(Table\_Link\_State(reciever))$
$\oplus\ motes \subseteq \{q \mid reciever\ \mapsto q \in Table\_Link\_State(reciever) \wedge$
$(\exists p, c.\, c \subseteq Table\_Link\_State(reciever)\ \wedge q \mapsto p \in Cessation(c) \wedge$
$card(c) = cluster\_Radius - 1 \wedge$
$(\forall s.\, s \subseteq Table\_Link\_State(reciever)\ \wedge q \mapsto p \in Cessation(s) \wedge$
$card(s) \geq card(c))))\}$
$\oplus\ motes \neq \emptyset$
$\oplus\ path = Transmitted\_Path(sender \mapsto reciever)(request)$
$\cup \{sender \mapsto reciever\}$

Receive_Request_Has_Route refine Receive_Request
$any\ sender\ reciever\ request\ motes\ path\ destination$
$Collection\ routes$
$where$
$\ominus\ path \in Motes \leftrightarrow Motes \wedge sender \mapsto reciever\ \in path$
$\oplus\ motes = \emptyset$
$\oplus\ destination = prj2(request)$
$\oplus\ reciever \mapsto\ destination \in$
$closure(Table\_Link\_State(reciever))$
$\oplus\ collection = \{S \mid S \subseteq Table\_Link\_State(reciever)\ \wedge$
$card(S) \leq Cluster\_Radius \wedge$
$reciever \mapsto\ destination \in Cessation(S)\}$
$\oplus\ routes = union(\{R \mid \forall S.\, S) \in collection \wedge$
$R \in collection \wedge card(R) \leq card(S)\})$

$\oplus\ path = Transmitted\_Path(sender \mapsto reciever)(request)$
$\cup \{sender \mapsto reciever\} \cup routes$

Another event, *receive_Request_Has_Route*, is responsible for creating route query reply along with the route that has been discovered. In the receiver's cluster radius, there is at least one route to the targeted mote. As a result, appropriate routes without redundant links should be chosen based on certain metrics, such as range (hops).To measure the union of all shortest paths, two guards *collection* and *routes* have been added.

The variable card represents the cardinality of a finite input set is sets be an empty set because it is unnecessary to border cast this request. As a result, the cumulative path, which a receiver constructs with the destination within its cluster radius, comprises the entire route from the sender mote to the target mote (thm1).

Req 8 is ensured in this refinement.

$inv1$:	$\forall s, n. s \in Motes \wedge n \in Motes \wedge s \neq n$
	$\Rightarrow (\forall r, path. r \in Route\_Request \wedge$
	$r \mapsto path \in Accumulated\_Path(n) \wedge\ s = prj1(r)$
	$\Rightarrow\ s \mapsto n\ \in Cessation\ (path))$
$thm1$:	$\forall s, t, n. s \mapsto t\ \in Route\_Request \wedge n \in Motes\ \wedge s \neq n$
	$\Rightarrow (\forall r, path. r =\ s \mapsto t \wedge$
	$r \mapsto path \in AccumulatedPath(n)$
	$\wedge n \mapsto t \in Cessation(\ path) \Rightarrow s \mapsto t \in Cessation(\ path))$

Seventh Refinement: The cluster-based query control mechanism aims to steer route requests away from the already covered cluster and sender mote. The query control mechanism is considered in this refinement. When constructing a border cast tree, a mote must be aware of the coverage information within its cluster radius.

Avariable *Cluster_Coverage* ∈ Mote*s* → *((Motes* × Mote*s)* → P*(Motes))* is defined to describe this information. A source node launches the route request and forwards it to the neighbors in its border cast tree, so its cluster radius is covered. Source mote initiates the route query request and forwards it to its bordercast tree neighbors, ensuring that its cluster radius is already covered. The cluster radius is marked as covered by refining *source_Forward_Request*. *Cluster* is the set of motes inside the source cluster radius.

Source_Forward_Request refines Source_Forward_Request
$any\ source\ destination\ request\ motes\ links\ zone$
Where
$\oplus\ cluster = \{m \mid source \mapsto m \in$
$Cessation(Table\_Link\_State(source)) \vee m = source\}$
$\oplus\ source \notin Cluster\_Coverage(source)(request)$
then
$\oplus\ Cluster\_Coverage(source) := Cluster\_Coverage(source) \leftarrow$
$\{request \mapsto cluster\}$

In addition, if the received route request has been sent, a bordercaster cluster radius is identified as covered (refine bordercast_Request). A new event *send_Reply* is defined to mark the cluster radius as covered.

Send_Reply
$any\ mote\ request\ source\ destination\ path\ cluster$
Where
$mote \in Motes$
$request \in Route\_Request$
$source = prj1(request) \wedge destination = prj2(request)$
$request \in dom(Accumulated\_Path(mote))$
$path = Accumulated\_Path(mote)(request)$
$source \mapsto destination \in Cessation(path)$
$cluster = \{q \mid mote \mapsto q \in Cessation(Table\_Link\_State(mote))$
$vq = mote\}$
then

$Cluster\_Coverage(mote) \coloneqq Cluster\_Coverage(mote) \leftarrow$
$\{request \mapsto cluster\}$

To prune the branches that lead to the motes that are already covered on the periphery, receive_Request_No_Route is refined with two conditions to calculate the *motes*. This refinement establishes the system requirement **REQ-9**.

Receive_Request_No_Route refines Receive_Request_No_Route
$any\ sender\ reciever\ \ request\ motes\ \ path\ destination$
where
$\ominus\ motes \subseteq \{q \mid reciever\ \mapsto q\ \ \in Table\_Link\_State(reciever)\ \wedge$
$(\exists p, c.c \subseteq\ Table\_Link\_State(reciever)\ \wedge q\ \mapsto p \in Cessation(c)$
$\wedge\ card(c) = cluster\_Radius - 1 \wedge$
$(\forall s.s \subseteq\ Table\_Link\_State(reciever)\ \wedge q\ \mapsto p \in Cessation(s)$
$\wedge\ card(s) \geq\ card(c)))\}$
$\oplus\ motes = \{q \mid reciever\ \mapsto q\ \ \in Table\_Link\_State(reciever)\ \wedge$
$(\exists p, c.c \subseteq\ Table\_Link\_State(reciever)\ \wedge q\ \mapsto p \in Cessation(c)$
$\wedge\ card(c) = cluster\_Radius - 1 \wedge$
$(\forall s.s \subseteq\ Table\_Link\_State(reciever)\ \wedge q\ \mapsto p \in Cessation(s)$
$\wedge\ card(s) \geq\ card(c)) \wedge$
$p \notin Cluster\_Coverage(reciever)(request)) \wedge q \neq sender\}$

4.5.4 Updating Cluster Routes

To formalize the cluster routing updates, four new variables have been defined. *Reply_Sender* is defined to describe the responses received from previous senders. *Reply_Path* sets out the routes obtained with the route query replies. Furthermore, *Transmitted_Sender* and *Transmitted_Reply_Path* reflect the information being transmitted.

$inv1$:	$Reply\_Sender\ \in Motes \longrightarrow (Motes) \leftrightarrow (Motes \times Motes))$
$inv2$:	$ReplyPath\ \in Motes \longrightarrow (Motes) \leftrightarrow (Motes \times Motes) \rightarrow$
	$(Motes \leftrightarrow Motes))$

$inv3$:	$Transmitted\_Sender \in (Motes \times Motes) \rightarrow (Motes \leftrightarrow (Motes \times Motes))$
	$(Motes \times Motes))$
$inv4$:	$Transmitted\_Reply\_Path \in (Motes \times Motes) \rightarrow$
	$((Motes \times Motes) \rightarrow (Motes \leftrightarrow Motes))$
$inv5$:	$\forall n, m, r. n \neq m \wedge Transmitted\_Reply\_Path\ (n \mapsto m)(r) \neq$
	$\emptyset \Rightarrow n \mapsto r \in Transmitted\_Sender\ (n \mapsto m)$
$inv6$:	$\forall n, s, d, rq, path. n \in Motes \wedge s \in Motes \wedge d \in Motes \wedge$
	$rq = s \mapsto d \wedge path = Reply\_Path(n)(s \mapsto rq) \wedge path \neq \emptyset$
	$\Rightarrow s \mapsto d \in Cessation(path)$

Update_Cluster_Table is further refined into two concrete events to demonstrate how motes update their cluster information using Req 10. One is *Update_Cluster_Table_IALC*.

It aims to keep the cluster routing details based on the neighborhood's existing connectivity. A mote's table can include routes to destinations outside the cluster radius. The motes that move out of the cluster radius must be identified, and their associated routes must be excluded. The variables *add_Routes* and *remove_Routes* are defined to gather all the link information of existing and prior cluster radius. The last guard is responsible for any changes in the cluster radius of the mote, which is a vital prerequisite for the update.

Update_Cluster_Table_IALC refines Update_Cluster_Table
$any\ mote\ add\_Routes\ remove\_Routes\ cluster\ motes$
$where$
$\bigoplus cluster = \{m \mid mote \mapsto m \in Cessation(Table\_Link\_State(mote))$
$\vee\ m = mote\}$
$\bigoplus add\_Routes = \{p, q. p \mapsto q \in Motes \times Motes \wedge$
$p \mapsto q \in Table\_Link\_State(mote) \mid p \mapsto q\}$
$\bigoplus motes = \{m \mid mote \mapsto m \in closure(Table\_Link\_State(mote)) \wedge$
$(\exists c. c \subseteq cluster\_Routing\_Table(mote) \wedge mote \mapsto m \in Cessation(c) \wedge$
$Card(c) \leq cluster\_Radius)\}$
$\bigoplus remove\_Routes = (mote \cup cluster) \lhd cluster\_Routing\_Table(mote) \rhd$

$(mote \cup cluster)$
$\oplus\ motes \neq \emptyset \vee \neg(add\_Routes = cluster) \lhd cluster\_Routing\_Table(mote) \rhd cluster)$

Update_Cluster_Table_Reply has defined to model the up-to-date routing information of a mote which obtains a route query reply packet. Links belongs to other routes are meaningless and not necessary to record. It satisfies the fact that the reversed cumulative path returns a route response to the query source. The entire query route is denoted by *the path.*

Every relaying mote must be aware of the routes which lead to the destination. Thus, the route from the source to the current relaying mote is removed from the path. *Add_Routes* is defined to denote the result links which are not included in the cluster table.

Update_Routung_Table_Reply refines Update_Cluster_Table
$any\ sender\ reciever\ add\_Routes\ remove\_Routes\ request\ path$
$where$
$\ominus\ mote \in\ Motes$
$\ominus \neg(add\_Routes = \emptyset \wedge remove\_Routes = \emptyset)$
$\oplus\ sender \in Motes$
$\oplus\ reciever \in Motes$
$\oplus \neg(add\_Routes = \emptyset \wedge remove\_Routes = \emptyset)$
$\oplus\ sender \mapsto request \in$
$Transmitted\_Sender(sender \mapsto reciever)$
$\oplus\ path \neq \emptyset \wedge path =$
$Transmitted\_Reply\_Path(sender \mapsto reciever)(request)$
$\oplus\ request\ \in closure(path)$
$\oplus\ reciever \in (dom(path)\{p \mid sender \mapsto p \in Cessation(path)$
$\vee\ p = sender\})$
$\oplus\ add\_Routes = \{p \mid p \in dom(path)\ \wedge$
$p \mapsto reciever \in Cessation(path) \lhd path$
$\oplus \neg addRoutes \subseteq Cluster\_Routing\_Table(reciever)$
With
$mote = reciever$

then
$\oplus\, Reply\_Sender(reciever) \coloneqq Reply\_Sender(reciever) \cup$
$\{sender \mapsto reciever\}$
$\oplus\, Reply\_Path(reciever) \coloneqq Reply\_Path(reciever) \lhd$
$\{(sender \mapsto reciever) \mapsto path\}$
$\oplus\, Transmitted\_Sender(sender \mapsto reciever) \coloneqq$
$Transmitted\_Sender(sender \mapsto reciever) \setminus$
$\{sender \mapsto reciever\}$
$\oplus\, Transmitted\_Reply\_Path(sender \mapsto reciever) \coloneqq$
$Transmitted\_Reply\_Path(sender \mapsto reciever) \lhd$
$\{request \mapsto \emptyset\}$

Send_Reply is refined to begin the route query reply process, and a new event *Forward_Reply* is introduced to forward the route query reply to the next mote determined by the cumulative distance. Until responding, a mote first ensures that it is connected to the next mote. Inside Send_Reply, *the mote* directs the already discovered paths to the *next* mote, determined by *the path*. The node and path information is then placed next to the link node *mote* → ***next***.

Send_Reply refines Send_Reply
$any\ mote\ request\ source\ destination\ path\ cluster\ next$
Where
$\oplus\, next \mapsto mote \in Accumulated\_Path(mote)(request)$
$\oplus\, mote \mapsto next \in Table\_Link\_State(mote)$
$\oplus\, Transmitted\_Reply\_Path(mote \mapsto next)(request) = \emptyset$
then
$\oplus\, Transmitted\_Sender(mote \mapsto next) \coloneqq$
$Transmitted\_Sender(mote \mapsto next) \cup \{mote \mapsto request\}$
$\oplus\, Transmitted\_Reply\_Path(mote \mapsto next) \coloneqq$
$Transmitted\_Reply\_Path(mote \mapsto next) \leftarrow \{request \mapsto path\}$

The formalization ensure Requirements Req 1 , Req 2 and Req 10. Figure 2 depicts the synchronization of the established events.

4.5.5 Validating the Model

The ProB, an animation and modeling checking tool, is used to validate the model, a plugin provided by Rodin. It enables the animation of Event-B models without requiring them to be translated into specific ProB models. In addition, this animator facilitates step-by-step animation as well as non-deterministic tasks. The ProB perspective in Rodin displays a summary of a machine's current state, a collection of all allowed events, and proper argument configurations. As a result, users can modify the system state by selecting an active event with possible arguments. There may be an infinite number of system states because the motes in the network, the broadcast time, and the cluster radius are not specified in this formalization.

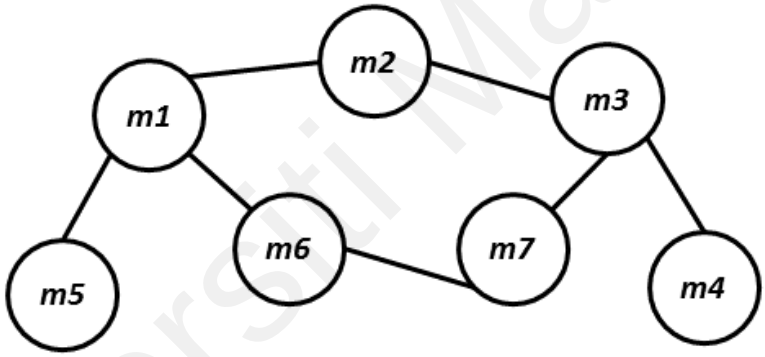


Figure 4.2: Network Topology for Model Validation

Auxiliary context is provided to assign the given set, *Motes*, and a few constants presented below to avoid state explosions and validate the model properly. A predicate, *Divider* (S, s1, . . ., sn), is defined to form partitions of set S s1.....sn, further Time_Clock is set to one indicating that the time is increment by one.

$axm1$:	$Divider(Motes, \{m1\}, \{m2\}, \{m3\}, \{m4\}, \{m5\}, \{m6\}, \{m7\})$
$axm2$:	$Lifetime = 4$
$axm3$:	$period = 2$
$axm4$:	$Cluster\_Radius = 2$

In this model, a Cessation (*closure)* function is defined to calculate the transitive closure. It is necessary to simplify this computation to make the animation smooth. As a result, the network topology in Figure. 4.4 is static. This network's connectivity is defined by the variable *Neighbor_Link*, which is treated as a constant (axm5). After that, a constant variable *closureN L (axm6)* is defined to describe the transitive closure of *Neighbor_Link (axm7)*. It is noticed that the stabilize event needs to be stimulated first for the system's stability state.

$axm5$:	$\text{Divider}(Neighbor\_Link,$
	$\{m1 \mapsto m2, m1 \mapsto m5, m1 \mapsto m6\},$
	$\{m2 \mapsto m1, m2 \mapsto m3\}, \{m3 \mapsto m2, m3 \mapsto m4, m3 \mapsto m7\},$
	$\{m4 \mapsto m3\}, \{m5 \mapsto m1\}, \{m6 \mapsto m1, m6 \mapsto m7\},$
	$\{m7 \mapsto m6, m7 \mapsto m3\})$
$axm6$:	$closureNL \in Motes \leftrightarrow Motes$
$axm7$:	$Partition(closureNL, NeighborLink,$
	$\{m1 \mapsto m3, m1 \mapsto m4, m1 \mapsto m7\},$
	$\{m2 \mapsto m4, m2 \mapsto m5, m2 \mapsto m6, m2 \mapsto m7\},$
	$\{m3 \mapsto m1, m3 \mapsto m5, m3 \mapsto m6\},$
	$\{m4 \mapsto m1, m4 \mapsto m2, m4 \mapsto m5, m4 \mapsto m6, m4 \mapsto m7\},$
	$\{m5 \mapsto m2, m5 \mapsto m3, m5 \mapsto m4, m5 \mapsto m6, m5 \mapsto m7\},$
	$\{m6 \mapsto m2, m6 \mapsto m3, m6 \mapsto m4, m6 \mapsto m5\},$
	$\{m7 \mapsto m1, m7 \mapsto m2, m7 \mapsto m4, m7 \mapsto m5\},$
$axm8$:	$Request \in Motes \leftrightarrow Motes$
$axm9$:	$Request = \{m1 \mapsto m4\}$

According to the above-mentioned network topology, it is intended to discover a path from $m1$ to $m4$. A variable *Request* is defined to record this route query information from sender $m1$ and receiver $m4$. The preliminary model was changed by introducing this auxiliary context and eliminating the current machine that defined variation in the network topology. The corresponding refinements are formalized according to this context.

1.Time_Clock(1) 2. timeClock(1)	$Time = 3$
3. Links_Obtain($m1, \{m1 \mapsto m2, m1 \mapsto m5, m1 \mapsto m6\}, \dots$) 4. Links_Obtain ($m2, \{m2 \mapsto m1, m2 \mapsto m3\}, \dots$) 5. Links_Obtain ($m3, \{m3 \mapsto m2, m3 \mapsto m4, m3 \mapsto m7\}, \dots$) 6. Links_Obtain ($m6, \{m6 \mapsto m1, m6 \mapsto m7\}, \dots$) 7. Links_Obtain ($m7, \{m7 \mapsto m3, m7 \mapsto m6\}, \dots$)	$Table\_Link\_State = \{m1 \mapsto \{m1 \mapsto m2, m1 \mapsto m5, m1 \mapsto m6\}, m2 \mapsto \{m2 \mapsto m1, m2 \mapsto m3\}$ $m3 \mapsto \{m3 \mapsto m2, m3 \mapsto m4, m3 \mapsto m7\}, m4 \mapsto \emptyset, m5 \mapsto \emptyset,$ $m6 \mapsto \{m6 \mapsto m1, m6 \mapsto m7, m1 \mapsto m2, m1 \mapsto m5, m1 \mapsto m6\} \dots)$
8.Send_Links($m1, \{m1 \mapsto m2, m1 \mapsto m5, m1 \mapsto m6\}, \dots$) 9.Links_Recieve ($m2, \{m1 \mapsto m2, m1 \mapsto m5, m1 \mapsto m6\}, \dots$) 10. recieveLinks($m6, \{m1 \mapsto m2, m1 \mapsto m5, m1 \mapsto m6\}, \dots$)	$Table\_Link\_State = \{m2 \mapsto \{m2 \mapsto m1, m2 \mapsto m3, m1 \mapsto m5, m1 \mapsto m6\},$ $\dots, m6 \mapsto \{m6 \mapsto m1, m6 \mapsto m7, m1 \mapsto m5, m1 \mapsto m6\}, \dots\},$
11. Send_Links($m2, \{m2 \mapsto m1, m2 \mapsto m3\}, \dots$) 12. Links_Recieve ($m1, \{m2 \mapsto m1, m2 \mapsto m3\}, \dots$) 13. Links_Recieve ($m3, \{m2 \mapsto m1, m2 \mapsto m3\}, \dots$)	$Table\_Link\_State = \{m1 \mapsto \{m1 \mapsto m2, m1 \mapsto m5, m1 \mapsto m6, m2 \mapsto m1, m2 \mapsto m3\},$ $\dots, m3 \mapsto \{m3 \mapsto m2, m3 \mapsto m4, m3 \mapsto m7, m2 \mapsto m1, m2 \mapsto m3\}, \dots\},$
14. Send_Links($m3, \{m3 \mapsto m2, m3 \mapsto m4, m3 \mapsto m7\}, \dots$) 15.Links_Recieve ($m2, \{m3 \mapsto m2, m3 \mapsto m4, m3 \mapsto m7\}, \dots$) 16.Links_Recieve ($m7, \{m3 \mapsto m2, m3 \mapsto m4, m3 \mapsto m7\}, \dots$)	$Table\_Link\_State = \{\dots, m3 \mapsto \{m2 \mapsto m1, m2 \mapsto m3, m1 \mapsto m2, m1 \mapsto m5, m1 \mapsto m6, m3 \mapsto m2, m3 \mapsto m4, m3 \mapsto m7\}, \dots,$ $m7 \mapsto \{m7 \mapsto m3, m7 \mapsto m6, m3 \mapsto m2, m3 \mapsto m4, m3 \mapsto m7\}, \dots\},$
17.Send_Links($m6, \{m6 \mapsto m1, m6 \mapsto m7\}, \dots$) 18. Links_Recieve ($m1, \{m6 \mapsto m1, m6 \mapsto m7\}, \dots$) 19. Links_Recieve ($m7, \{m6 \mapsto m1, m6 \mapsto m7\}, \dots$)	$Table\_Link\_State = \{m1 \mapsto \{m1 \mapsto m2, m1 \mapsto m5, m1 \mapsto m6, m2 \mapsto m1, m2 \mapsto m3, m6 \mapsto m1, m6 \mapsto m7\}, \dots,$ $m7 \mapsto \{m7 \mapsto m3, m7 \mapsto m6, m3 \mapsto m2, m3 \mapsto m4, m3 \mapsto m7, m6 \mapsto m1, m6 \mapsto m7\}, \dots\},$
20. Send_Links($m7, \{m7 \mapsto m3, m7 \mapsto m6\}, \dots$) 21. Links_Recieve ($m3, \{m7 \mapsto m3, m7 \mapsto m6\}, \dots$) 22. Links_Recieve ($m6, \{m7 \mapsto m3, m7 \mapsto m6\}, \dots$)	$Table\_Link\_State = \{\dots, m3 \mapsto \{m3 \mapsto m2, m3 \mapsto m4, m3 \mapsto m7, m2 \mapsto m1, m2 \mapsto m3, m7 \mapsto m3, m7 \mapsto m6\}, \dots,$ $m6 \mapsto \{m6 \mapsto m1, m6 \mapsto m7, m1 \mapsto m2, m1 \mapsto m5, m1 \mapsto m6, m7 \mapsto m3, m7 \mapsto m6\}, \dots\},$
23.Update_Table_Routing_IALC($m1, \dots$) 24. Update_Table_Routing_IALC ($m2, \dots$) 25. Update_Table_Routing_IALC ($m3, \dots$) 26. Update_Table_Routing_IALC ($m6, \dots$) 27. Update_Table_Routing_IALC ($m7, \dots$)	$Cluster\_Routing\_Table = \{m1 \mapsto \{m1 \mapsto m2, m1 \mapsto m5, m1 \mapsto m6, m2 \mapsto m1, m2 \mapsto m3, m6 \mapsto m1, m6 \mapsto m7\}, m2 \mapsto \{m2 \mapsto m1, m2 \mapsto m3, m1 \mapsto m2, m1 \mapsto m2, m1 \mapsto m5, m1 \mapsto m6, m3 \mapsto m2, m3 \mapsto m4, m3 \mapsto m7\}, m3 \mapsto \{m3 \mapsto m2, m3 \mapsto m4, m3 \mapsto m7, m2 \mapsto m1, m2 \mapsto m3, m7 \mapsto m3, m7 \mapsto n6\}, n4 \mapsto \emptyset, m5 \mapsto \emptyset, n6 \mapsto \{m6 \mapsto m1, m6 \mapsto m7, m1 \mapsto m2, m1 \mapsto m5, m1 \mapsto m6, m7 \mapsto m3, m7 \mapsto m6\}, m7 \mapsto \{m7 \mapsto m3, m7 \mapsto m6, m3 \mapsto m2, m3 \mapsto m4, m3 \mapsto m7, m6 \mapsto m1, m6 \mapsto m7\}\}$

28.Source_Forward_Request($m1, m1 \mapsto m4, \dots$)	$Accumulated\_Path = \{m1 \mapsto \{(m1 \mapsto m4) \mapsto \emptyset, \dots\}, \dots\}$ $Intended\_Neighbor = \{(m1 \mapsto \{(m1 \mapsto m4)) \mapsto \{m2, m6\}, \dots\}$
29.Recieve_Request_Has_Route($m2, m1 \mapsto m4, \dots$) 30. Receive_Request_No_Route($m6, m1 \mapsto m4, \dots$)	$Accumulated\_Path = \{m2 \mapsto \{(m1 \mapsto m4) \mapsto \{m1 \mapsto m2, m2 \mapsto m3, m3 \mapsto m4\}, \dots\}, \dots, m6 \mapsto \{(m1 \mapsto m4)\{m1 \mapsto m6\}, \dots\}, .\}$ $Intended\_Neighbor = \{\dots, \big(m6 \mapsto (m1 \mapsto m4)\big) \mapsto \{m7\}, \dots\}$
31.Bordercast_Request($m6, m1 \mapsto m4, \{m6 \mapsto m7\}, \dots$) 32. Receive_Request_Has_Route($m7, m1 \mapsto m4, \dots$)	$Accumulated\_Path = \{\dots, m7 \mapsto \{(m1 \mapsto m4) \mapsto \{m1 \mapsto m6, m6 \mapsto m7, m7 \mapsto m3, m3 \mapsto m4\}, \dots\}, \dots\}$
33.Send_Reply($m2, m1 \mapsto m4, \dots$) 34.Send_Reply($m7, m1 \mapsto m4, \dots$)	$Transmitted\_ReplyPath = \{(m2 \mapsto m1) \mapsto \{(m1 \mapsto m4) \mapsto \{m1 \mapsto m2, m2 \mapsto m3, m3 \mapsto m4\}, \dots\}, \dots, (m7 \mapsto m6) \mapsto \{(m1 \mapsto m4) \mapsto \{m1 \mapsto m6, m6 \mapsto m7, m7 \mapsto m3, m3 \mapsto m4\}, \dots\}, \dots\}$
35.Update_Routing_Table_Reply($m1, m1 \mapsto m4, \dots$) 36. Update_Routing_Table_Reply($m6, m1 \mapsto m4, \dots$)	$Cluster\_Routing\_Table = \{m1 \mapsto \{m1 \mapsto m2, m1 \mapsto m5, m1 \mapsto m6, m2 \mapsto m1, m2 \mapsto m3, m6 \mapsto m1, m6 \mapsto m7, m3 \mapsto m4\}, \dots, m6 \mapsto \{m6 \mapsto m1, m6 \mapsto m7, m1 \mapsto m2, m1 \mapsto m5, m1 \mapsto m6, m7 \mapsto m6, m7 \mapsto m3, m3 \mapsto m4\}, \dots\}$
37.Forward_Reply($m6, m1 \mapsto m4, m1 \dots$) 38.Update_Cluster_Routing_Table_Reply($m1, m1 \mapsto m4, \dots$)	$Cluster\_Routing\_Table = \{n1 \mapsto \{m1 \mapsto m2, m1 \mapsto m5, m1 \mapsto m6, m2 \mapsto m1, m2 \mapsto m3, m6 \mapsto m1, m6 \mapsto m7, m3 \mapsto m4, m7 \mapsto m3\}, \dots\}$

The Cessation (*closure)* predicates are *replaced* with some other predicates regarding the cluster radius 2 for the following refinements. The radius of the cluster is set to 2, whereas *motes(x)* denotes the set of *x's* neighbors.

To verify the existing routes that reside in the *x's* neighbors, it is required that $x \rightarrow y$ must be in (*x)*. It's worth noting that those changes only apply to models with a specific cluster radius 2.

Few statements, such as *inv1 and thm1* in the sixth refinement, cannot be changed because they contain the *Cessation (closure)* function with an unspecified path. Hence, the *Cessation (closure)* function is neglected, and more attention needs to pay to manually check their correctness while animating the models. To discover routes from $m1$ to $m4$, some operations are enabled manually by this model. The sequence of operations is used to discover the route query in the network topology.

[$m1 \rightarrow m2$, $m2 \rightarrow m3$, $m3 \rightarrow m4$] and [$m1 \rightarrow m6$, $m6 \rightarrow m7$, $m7 \rightarrow m3$, $m3 \rightarrow m4$]

The animation example is elaborated in Figure 4.2. A list of step-by-step operations is carried out to enhance the readability, including some additional parameters to discover the entire route discovery from $m1 \rightarrow m4$. Steps 3-7 are used to obtain the neighbor information for $m1$, $m2$, $m3$, $m6$, and $m7$, and the current value of *Table_Link_State* is displayed on the right side. Static parts of variables are represented by ". . .". During the animation, the axioms and invariants remain true.

4.6 Conclusion

This Chapter used a refinement-based process to design a formal specification of cluster-based flooding. The aim is to use the border casting service to examine the route discovery process. Every mote in the dynamic network environment sends information to its neighbors in the cluster radius. This model is not only limited to formally describes the above issues but also takes account of the system's stabilization property. The system is considered stable if the network environment is idle for a long time. Every mote in the cluster has a route to any other mote within its radius.

Some invariants are also defined to validate the route discovery properties. It is to be noted that the CBF's target is to use a border casting service rather than broadcasting or flooding to find the appropriate routes. It has formed approximately 400 proof obligations, of which half are automatically proven.

The rest, which includes an arithmetic or set operations, are proven manually. Discharging the generated proof obligation ensures that the refinements are correct, and the properties (invariants) are preserved. The development can also be used to explore other cross-layered routing protocols.

CHAPTER 5: QoS ENABLED QCM TESTBED

5.1 Introduction

In the conventional IoT query propagation model, sensors send requests to the access point or central gateway mote of the network. This access point handles the queries first and then transfers queries using the underlying routing mechanism to the right locations of the network.

There are shortcomings to this simplistic approach: sensors may send redundant and duplicate queries i.e one sensor may carry unwanted query of another application or sensor. This is because of the overlapping cluster queries. The total energy usage therefore grows as the query size is enlarged. In such situations, device resources (in terms of bandwidth or energy for the sensor node) are lost due to too many redundant network query transmissions that can result in detectable QoS transmission (Fathallah et al., 2019).

This Chapter discusses the QoS enabled QCM testbed used to detect and terminate the redundant and unwanted queries in IoT networks. The QCM testbed aims to reduce the number of duplicate/overlapping queries in IoT networks to improve QoS. The query control mechanism is aware of all the query information. Therefore, all the overlapping clusters in the whole query space can be easily calculated.

The Chapter is organized into the following sections. Section 5.1 presents the Introduction, Section 5.2 explains the benchmarking testbeds, Section 5.3 elaborates the experimental tools and schematic of the testbed, Section 5.4 presents the QCM in detail and elaborate different query detection and termination scenarios to evaluate the performance of QCM, Section 5.5 presents evaluation, Section 5.6 mentions the performance results and Finally, Section 5.7 concluded the Chapter.

5.2 Existing Testbeds

All around the planet, there are several existing testbeds for related purposes. Different testbeds or laboratories concentrate on multiple facts. Some state-of-the-art testbeds are mentioned below. It is essential to notice that none of the below testbeds solely focused on elimination of redundant and unwanted queries in IoT networks to enhance QoS. The QCM testbed is purely for the purpose of mitigation of multiple and overlapping cluster queries in IoT networks.

5.2.1 FIT IoT-LAB:

FIT IoT is an open source testbed composed of 2728 low-power remote motes and 117 portable robots that are being utilized to explore different research and experiments regarding the huge scope IoT usage (Adjih et al., 2015). It features low-level and advanced Internet-level protocols and is deployed at six sites across France.

Even though all the destinations have distinctive sensor gateway hub and equipment abilities, every one of them are associated and accessible through a similar online interface, normal REST interfaces and reliable CLI devices. It makes a platform for heterogeneous testing and is totally open source.

5.2.2 INDRIYA 2:

This testbed is the modern version of INDRIYA 1, installed at Singapore national university in a 3-dimensional way (Doddavenkatappa et al., 2012). This testbed is specifically intended for testing in programming environments for sensor networks, communication protocols and device architecture, etc. It provides the research platform with public access round the clock. At any given stage, anybody can upload executables, build jobs over the motes, and schedule them to run. Then, via the web portal, the tracking and visualization portion is completed. Deployment-wise Indriya 2nd version is

essentially the same as the previous. This solves some of the problems such as having different types of motes, more scalability, and having only one language base for development to make it easier to manage and update (Appavoo et al., 2019).

5.2.3 MoteLab:

It is a network-based testbed that has a series of nodes permanently installed that are connected to a central server that has a web interface that stores information and schedules activities. By logging the data with the assistance of automation, it smoothes production and debugging, which then ensures that the performance of the sensor network system is tested offline. It also provides access to the test bed for both local and remote users with the web interface. The scheme that takes care of the planning quota means that there is a rational system of sharing. It is completely open source and deployed at the University of Harvard (Werner-Allen et al., 2005).

5.2.4 The TKN WIreless NetworkS Testbed:

The TKN WIreless NetworkS Testbed (TWIST) has been planned and developed at the Technische Universität, Berlin, by the Telecommunication Networks Company (TKN). Another open-source indoor WSN testbed is versatile and supports experimenting with multiple node setups, network-wide scripting, and various debugging methods. It also offers support for a heterogeneous network setup and has self-configuration capability using generic interfaces on the hardware (S.-F. Li et al., 2005).

5.2.5 FlockLab:

The FlockLab is a testbed developed and operated at the Swiss Federal Institute of Technology in Zurich, Switzerland by the Electronic Engineering and Networks Laboratory. This testbed that requires several services to run through all nodes concurrently and synchronously.

This testbed has an additional function, as all the nodes are equipped with GPIO pins to store all logical events. It uses GPIO tracing as a debugging method for time-sensitive code (Lim et al., 2013). It embraces several target architectures, which in essence, when it comes to the same physical topology, enables a comparative study of applications and protocols. Users can apply power profiling and GPIO tracing against all targets to compare power and logical occurrences. They may also dynamically change the target supply voltage to strongly imitate battery consumption.

5.2.6 SensLAB:

It is an open WSN testbed that has been established and implemented on a vast scale to enable scalable testing in the WSN domain. The testbed is made up of 1024 nodes and is scattered around four areas. To provide a broad variety of features and implementations, every position has an equal proportion of 256 sensor nodes with unique characteristics. Two sites have links to mobile nodes, and all 256 nodes can connect with each other using the radio interfaces inside any given location. Each node can also serve as a sink node and can also connect with the entire network's other sink nodes or any external computer on the internet (Burin des Rosiers et al., 2012) .

5.3 Experimental Tools and Schematic

This Section explained the tools used to evaluate testbed for Query Control Mechanism along with IoT controller used to provide centralized control of the entire network. In the QCM testbed, 16 heterogeneous IoT sensors are deployed with Arduino Mega 2650 controller.

5.3.1 Arduino Controller

Arduino is an open-source hardware project that designs and manufactures single-board microcontrollers and microcontrollers to sense and control physical and digital

objectives for digital frame devices and interactive purposes (*Arduino - ArduinoBoardMega2560*, n.d.), (Badamasi, 2014), (Barrett, 2013).

ATmega2560 is a 54 digital input/output pin dependent microcontroller board where 14 are often used as PWM outputs. Sixteen analog inputs, 4 UART serial hardware ports, a 16 MHz crystal oscillator, a USB link, a power jack, an ICSP header, and a reset button. It includes everything and needs to help the microcontroller; plug it into a USB cable device or power it to get started with an AC to DC converter or charger. Figure 5.1 and Table 5.1 illustrates Pinout diagram and technical specification for ATmega2560.

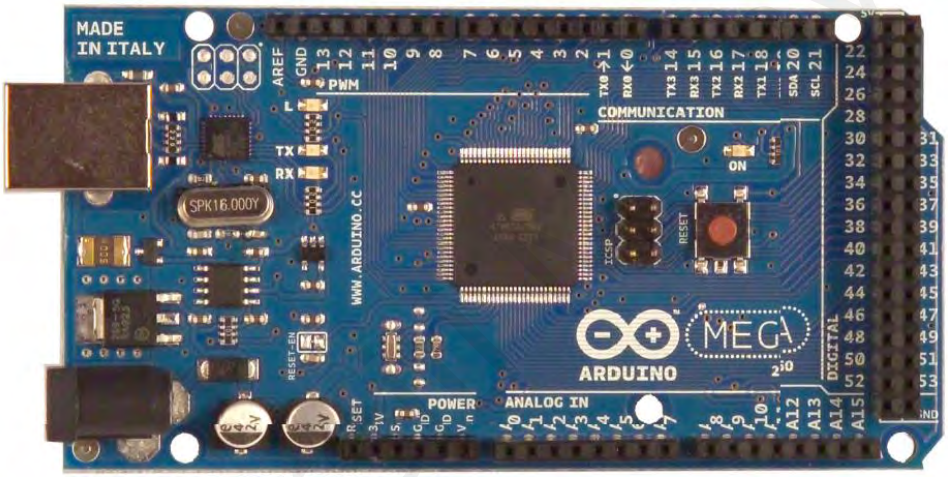


Figure 5.1: Pinout Schematic of ATmega 2650

The Arduino Mega is explicitly designed for projects requiring more memory space and complex circuits. Other boards available on the market that make Arduino Mega unusual for regular projects can do most of the electronic projects well. However, due to its ability to store more instructions in the code memory and several digital and analog I/O pins, some projects are carried out exclusively by Arduino Mega, such as making 3D printers or controlling more than one device.

In this testbed all traffic management is controlled by a central IoT controller. This testbed used Gunino Mega 2650. IoT Controller is responsible to collect data sensed

by IR and RFID sensors and to take appropriate action for each input based on the appropriate algorithm.

Table 5.1: Specification of Mega 2650

Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

5.3.2 IR Sensor:

Infrared sensors are the electronic devices used to sense different characteristics of their surroundings (i.e heat, motion, etc). IR sensors play a key role in this testbed by sensing the arrival of the query on every cluster for calculating the query traffic on each cluster and transfer the counts to the main controller. Furthermore, the IoT controller calculates and analyzes the query traffic by counting the number of queries on a particular cluster and decide in a centralized manner accordingly (Zappi et al., 2010), (*IR Sensor : Circuit Diagram, Types Working with Applications*, n.d.).

5.3.3 RFID Sensor: Radio Frequency Identifier

RFID is a promising technology for automated object identification with high accuracy. RFID sensors are deployed for the detection and identification of special queries on each cluster. In the IoT network, WSNs and RFID are the enabling technologies that form the basic building blocks for sensing and communication among devices and objects connected to the network (Su, Jian and Liu, Alex X and Sheng, Zhengguo and Chen, 2020). The inexpensive nature of RFID and its strong support

towards the business community make it the leading enabling technology in the IoT domain because it can transform ordinary objects into smart objects (Landt, 2005). Wireless sensor networks are composed of heterogeneous technologies which include sensors, wireless, and fixed communications devices. Constructing IoT is significantly based on sensors and RFID. IoT provides a platform for people and different objects to connect and communicate together anytime and anywhere with anything in real-time provided they be online.

5.3.4 Indicators:

The QCM testbed is equipped with RGB indicators to notify the user about the flooding. These indicators are dynamically integrated with the IoT controller and change its state according to the mentioned criteria (imposed by IoT controller).

5.3.5 Cloud / Edge Servers:

Cloud server is a central entity with huge storage and computing capabilities. Whereas the edge server is a small cloud near the end-user. In the QCM testbed cloud and edge servers are used only when the local IoT controller is unable to perform complex computation due to its limited storage and computing capabilities. in such cases, tasks and computation are offloaded to the local edge or centralized cloud servers. Cloud servers are also used to analyze statistical data and intelligent future decision (Xiong et al., 2018), (Peng et al., 2020), (Ren et al., 2019).

5.3.6 Layered Description and Schematic

The QCM testbed schematic is classified based on the architectural design of each component used in developing the QoS enabled QCM testbed. Each component used and their description is provided in Table 5.2.

Table 5.2: Layered Description of Hardware

Network layer	NRF Model 324G (nRF241,01+1) , WIFI, xigbee etc
	Wifi Serial Module (ESP-8266)
Sensor Layer	Arduino Board Mega Rev 3
	IR sensor Low-Cost Ranging Module , PIR sensors , Ultrasonic , Water Moister , RFID
	RFID Tag and Reader

From the IoT layered description mentioned in Table 5.2, a circuit is developed by combining all the components discussed in Section 5.3 and Figure 5.2. which helped to achieve the aims and objectives of QoS enabled QCM testbed**.**

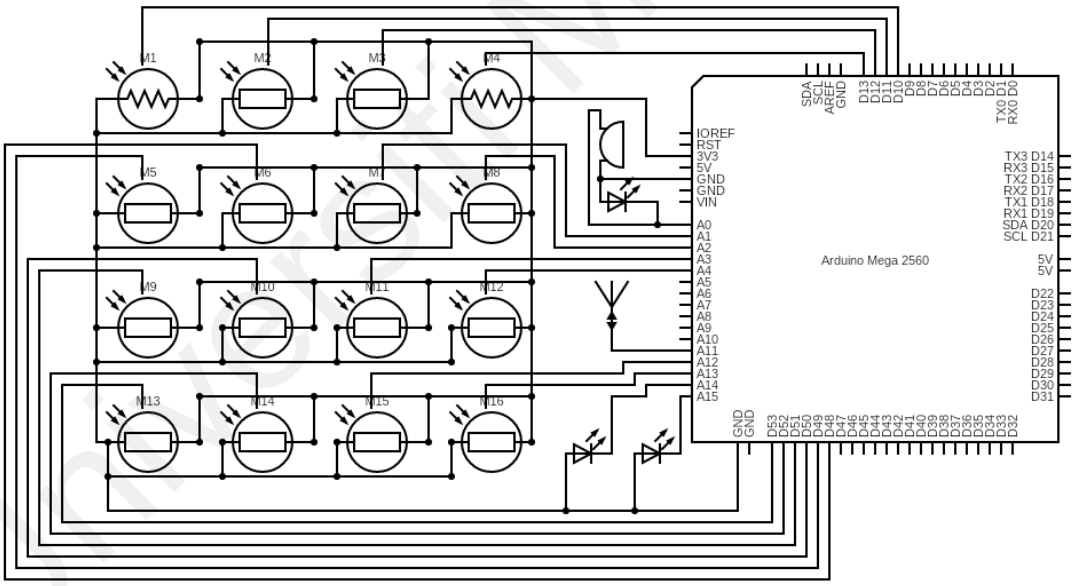


Figure 5.2: Schematic of the QCM Testbed

The testbed is capable of detection and termination of overlapping and redundant network queries that reappear in the same cluster that has been already queried. The QCM testbed also has a unique feature to monitor the mote location and its signal strength for

the purpose to identify the flooding queries and transfer them to the controller for termination and further processing.

The testbed used heterogeneous IoT Sensors, i.e., IR, Ultrasonic, RFID, temperature, sound, LDR, and water sensing for generating queries. It acts as the input unit for the testbed that monitors and sensed each query in their vicinity. The testbed uses embedded devices provided in Table 5.2 to carry out its essential operation. Embedded devices are being used in this system due to their advantages such as user-friendly, easy to program, and simplicity.

The retrieved information was then sent to the Arduino controller for a certain processing amount on the sensors query. Then forward the sent queries to computational servers, i.e., Cloud and local servers via ZigBee or Wi-Fi to be processed further for sharing and storing purposes. The controller is also responsible for the computation of channel signal strength and mote location.

5.4 Query Control Mechanism (Testbed)

In Cluster Based Flooding, clusters are heavily overlap due to which route queries are extended to multiple network motes. In fact, the query network has been effectively streamlined to reach all network motes, effectively flood the network. However, a more disappointing result is that the IELC generates much more traffic than the flooding itself since when the flood is in operation, the query is sent on a path the length of which is equal to the radius of the cluster. In this case, an adequate query termination criterion is needed then the traditional approaches provided. To understand the cause of CBF control traffic, one of the main characteristics of the routing cluster needs to be highlighted: the mote reply to a query request may have provided detailed information about the entire

cluster of the mote. From this perspective, additional path query traffic can be viewed because of query threads overlapping, that is overlapped visited cluster. Therefore, the query control procedures' purpose is to reduce the route query traffic by directing the threads outward from the visited cluster. This issue can be tackled by identification, termination and prevention of visited cluster overlap. See Figure 5.3.

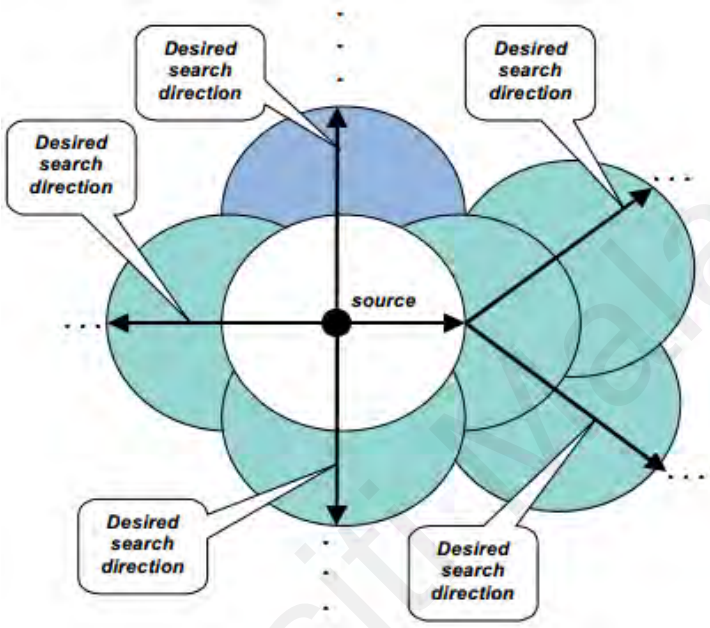


Figure 5.3: Desired Search Direction of Overlapping Clusters

5.4.1 Smart Query Mitigation

The conventional method to eliminate the query thread is to remove once it appears in the last queried mote. Thus, it does not entirely leverage the cluster structure.

A more comprehensive technique is to drop a thread that occurs in a cluster that has already been queried. This criterion poses the first obstacle to designing an efficient termination process: identifying the already visited cluster when only a single mote has been queried (the central mote).

5.4.1.1 Loop-Back Mitigation (LM)

When all motes are configured with the same cluster radius, cluster membership is commutative (i.e., if mote1 belongs to node m2's cluster, then node m2 belongs to m1's cluster). It is identifying a thread which returns to a cluster that it has already queried is reasonably straightforward. To decide whether every hop (excluding the most recent hop) lies within its cluster, a mote merely examines the acquired path in the obtained route query packet. The thread is abandoned when the loop-back event persists. Figure 5.4 demonstrates this scheme's instance, which refers to as Loop-back mitigation (LM). Mote m8 send a query route to m6, which forward it to m10, m10 send the same query message to m12. m12 eliminates the query and will not broadcast the query to m8 because it also lies in the m12's cluster. LM is an effective loopback thread handling mechanism since the cumulative path's knowledge is adequate to classify all loopback events.

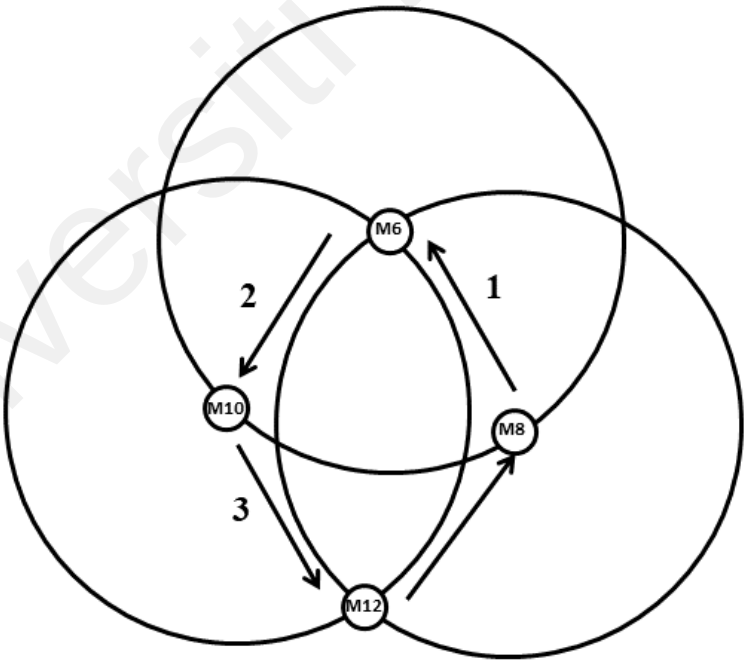


Figure 5.4: Loop-Back Mitigation (LM)

(a) LM termination criteria:

1. mote ∈ *route*

2. i < j, where *route*[i] = = mote and *route*[j] = = prev_periph

3. mote ∈ {*cluster*}

4. mote ∉ {*border motes*}

5.4.1.2 Smart Query Detection (SQD(a) / SQD(b))

The main cause of redundant queries in the cluster is because of it has been visited by another query. In contrast to the loopback scenario as mentioned previously, most of the thread overlapping occurs by a thread appearing in a cluster that was previously queried by another thread. Unlike the loop-back case just described, in this case, the ability to reduce and terminate enormously relies on the motes' ability to identify a cluster they belong to, has already been visited. The source mote in the cluster who processed the query clearly understands that the same query has already visited its cluster. To inform the rest of the cluster motes, some type of 'spying' techniques needs to be implemented without adding unnecessary control traffic. According to the CBF design principle, it is a more suitable way to perform query detection smartly. The initial level of smart query detection SQD(a) is to permit the interior motes responsible for forwarding the queries to cluster edge for detecting and revoking these queries. Any mote can sense queries within the range of a query-transmitting mote on single-channel networks. The SQD(b) can expand query detection functionality by providing route queries using IP broadcasts. Figure 5.5: illustrates both levels of smart query detection.

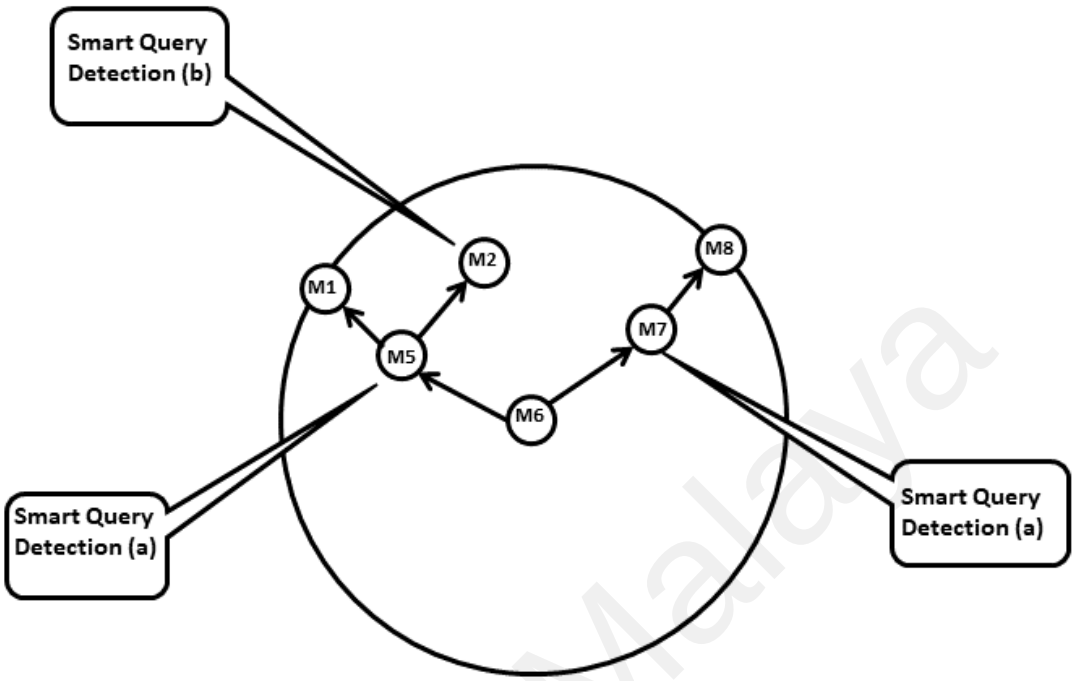


Figure 5.5: Smart Query Detection SQD(a) / SQD(b)

Mote identification is also recorded in terms of ID, which first broadcast the detected query thread. Redundant queries that can transmit by the same mote are not automatically revoked to ensure full network coverage. For instance, two border motes m1 and m8 received a flooding query from m6. Using SQD(a), the interior motes m5 and m7 can detect redundant queries passing through it. In another way, when SQD(b) is applied, mote m2 can "eavesdrop" on the transmissions of m5 and record the query thread.

5.4.1.3 Early Mitigation (EM)

The thread termination can further be improved by a throw away the query thread on entering to already queried cluster. When only the border motes are allowed to terminate the query thread, it is possible that redundant queries may enter the already visited areas, resulting in excessive traffic generation.

The termination capability may extend to the interior motes that forward the query thread to eliminate this excessive traffic generation. This method is referred as Early Mitigation (EM) as mentioned in Figure 5.6. in the Figure 5.6 a route query is broadcasted by m6 to m11, first this query is received by mote 10 and forward it to mote 8 for delivery to the destination mote 11 which is reside on the border of the cluster. Mote m8 terminates the thread instead of delivering the query to mote m11 because another thread of the same query was already detected. It is noted that EM only permits the partial involvement of the interior motes to process the route queries. Interior motes are forbidden to issue new queries, or else the CBF would convert into a flooding protocol.

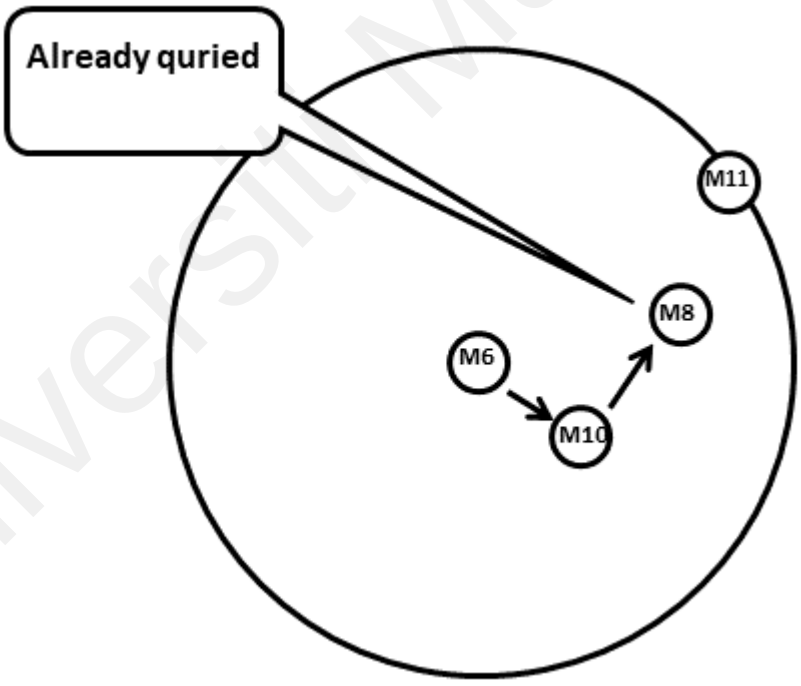


Figure 5.6: Early Mitigation

(a) EM termination criteria:

1. *Query* → {source,id} ∈ *detected queries*

2. *Query* → prev_periph > min {*detected queries*[] → prev_periph}

5.4.1.4 Selective Flooding

Now to solve the more complicated problem of thread overlap prevention. By focusing on eliminating local overlaps, a certain degree of control can be exerted on the direction of thread distribution, thereby reducing the overlap of threads farther in the network. Local thread overlap is caused by the overlap of peripheral motes in the cluster, particularly with cluster radius enlargement.

Instead of flood queries to all the border motes, a method called selective flooding (SF) can be used to provide the same coverage by flood the queries to a subset of selected/chosen border motes. SF must be aware of the network topology of extended cluster information provided by the IALC, which is twice the cluster radius before forwarding, initially, a mote needs first to specify the subset of outer border motes visited by its given inner border motes.

Sender mote needs to flood the query to the subset of given inner border motes, which in turn minimize the partitioning set of outer border motes. An illustrative example of SF operation is provided in Figure 5.7.

The inner border motes of m10 are m5, m6 and m7, whereas the outer border motes of m10 are m0, m1, m2, m3, m4 and m8. As seen because of the overlapping cluster, the inner border motes (m2 and m3) of m6 are also inner border motes of m5 and m7.

Hence, m6 can choose to be eliminated from the forwarding recipient list of m10. Mote5 can cover m0, m1 and m2, while m7 can give coverage to m3, m4 and m8, avoiding overlapping cluster queries over extended clusters with maximum coverage.

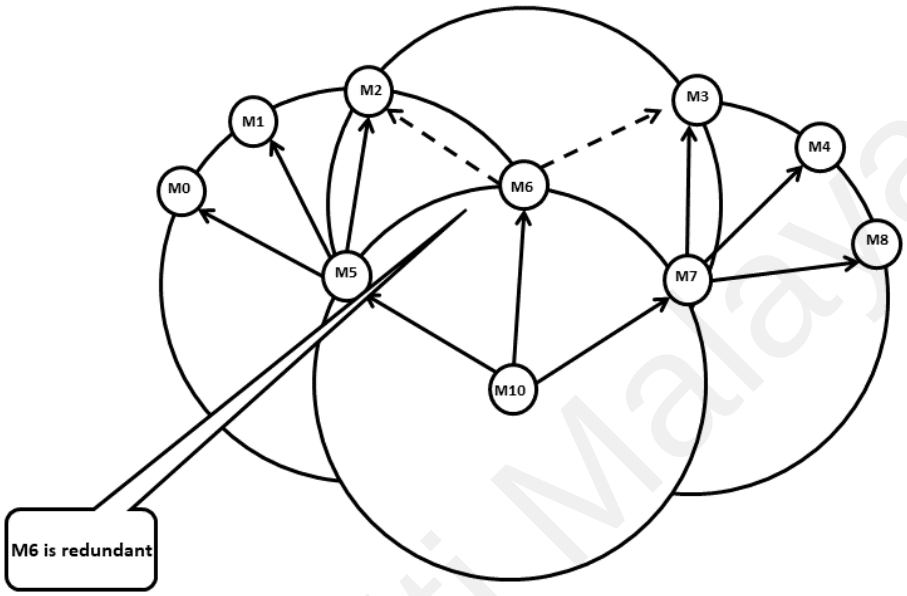


Figure 5.7: Selective Flooding (SF)

5.5 Evaluation of Query Control Mechanism (QCM)

The efficiency of the QCM testbed was assessed by using 16 heterogeneous IoT motes having a cluster radius (r). Effectiveness of the QCM was measured by the generation of control traffic. Instead of measuring the control traffic in term of packets, it measures it in ID fields which are transmitted at the IELC (i.e. network layer).It is because the route accumulation length of IELC control traffic is variable. Total control traffic can be seen as the addition of ID fields in the query packet transmitted by the intracluster update and intercluster reply to queries. Hello or alive beacons used for mote discovery are exempted from the control traffic

To accommodate the computational load of testbed, the IALC and IELC are emulated separately. The Mega 2560 microcontroller from Arduino family, based on the ATmega2560P microcontroller by Atmel with 16 heterogeneous IoT sensors were deployed to gauge the efficacy of IALC in each radius of a cluster. Initially, the IALC algorithm was executed for 300 seconds

The IALC algorithm was run for 300 seconds. None of the queries was recorded for the first ten seconds of the emulation to prevent it from additional measurements and to stabilize the initial Intracluster path query discovery process. The IELC algorithm assumes that the network topology may be static during the route request process.It is often presumed that the total network load is low. The delays in query propagation and mote processing are relatively insignificant.

5.6 Performance Results

The following statistics present the findings of the smart QCM testbed. Figure 5.8 indicates the cluster radius (r) dependency of intracluster control packets on different network reconfiguration. The full flooding and selective flooding methods are different because the selective flooding allows the IALC to retain an extended cluster of radius = (2). It is demonstrated that the rise in IALC traffic resulting from the extended cluster is quite significant. The sum of IALC control traffic for each mote depends on the cluster's radius r* for both selective and full flooding. It is expected because the amount of IALC traffic per mote is directly proportional to the number of queried motes in the cluster. As shown, when the cluster radius r = 1, the control overhead for intracluster is none because all the motes in a radius of r=1 are by default directly connected motes and considered neighbors. Further, MDP is responsible for propagating information required to maintain the mote connectivity inside the cluster.

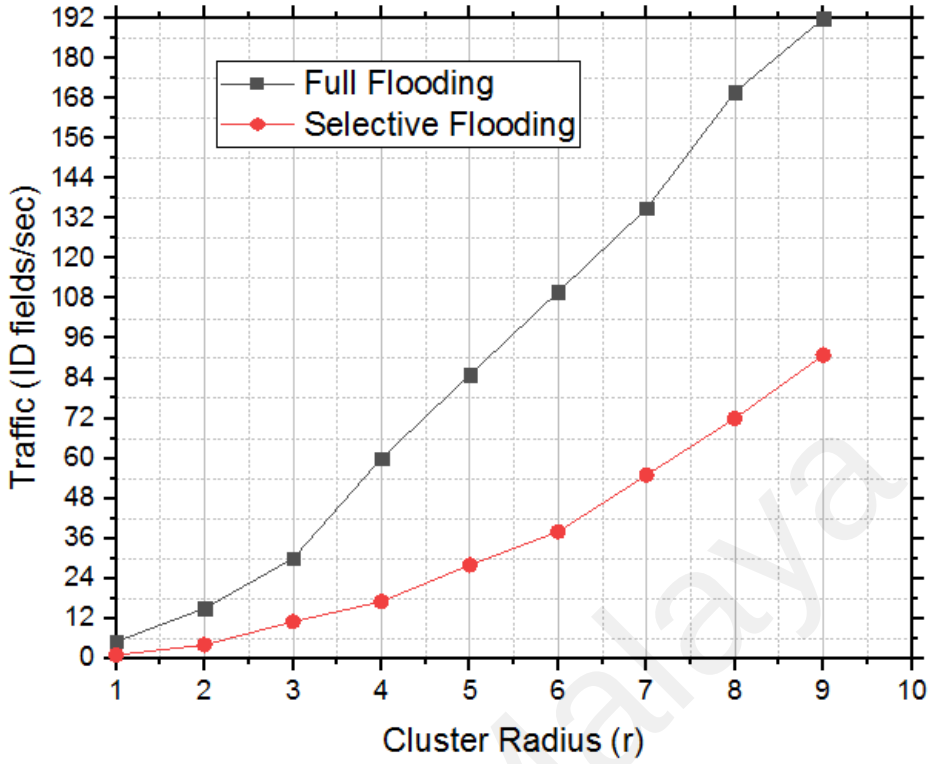


Figure 5.8: IALC Traffic

To analyze the IELC control traffic, initially, the full flooding and selective flooding were examined separately.

The efficiency of the query mitigation strategies, which are useful in managing the distribution of IELC traffic in conjunction with full flooding, is seen in Figure 5.9. The amount of IELC traffic per Route query is expected to decrease the cluster radius to be considered successful. To properly spread the query packet, some form of smart query detection technique (either SQD(a) or SQD(b) is required. Approximately 40 percent less reactive path discovery traffic could be encountered by single-channel networks that can implement SQD(b) than those that only implement SQD (a).

Early mitigation is shown to be not an effective technique by itself, but it delivers a significant performance in term of IELC traffic reduction when combined with other

methods. The amount of traffic by IELC seems to be decreasing by extended the query detection capabilities, and the elimination criteria become stricter.

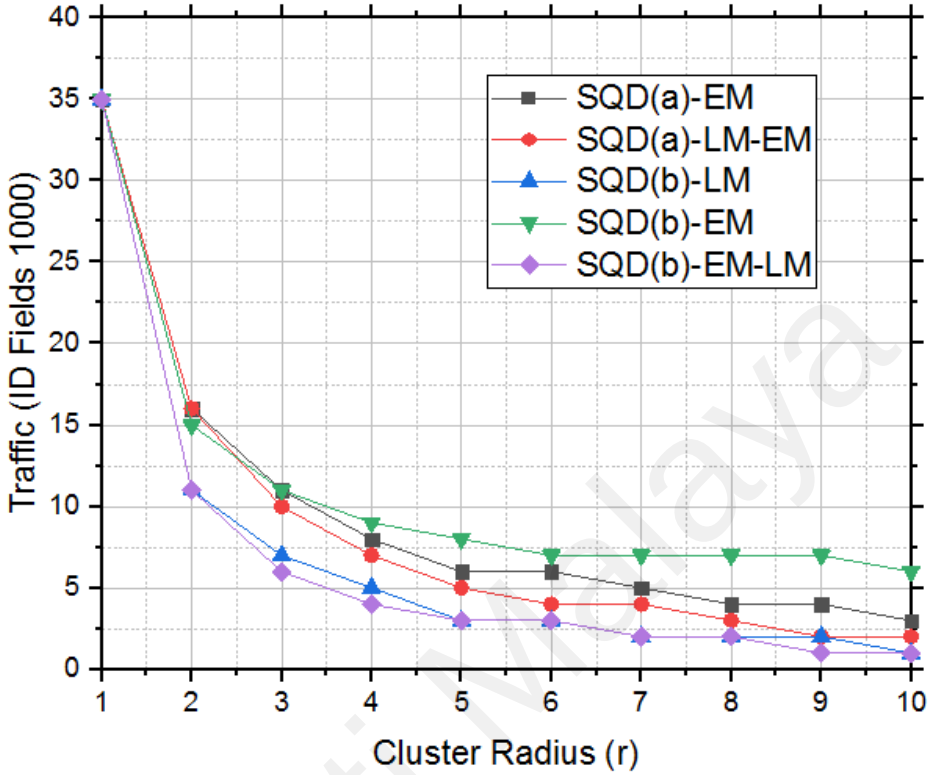


Figure 5.9: IELC Traffic (Full Flooding)

The extent to suppress redundant query traffic is clearly illustrated in Figure 5.9, with the deployment of the QCM. As mentioned earlier, the amount of reactive traffic would increase while increasing the cluster radius without an efficient query management technique. It is noted that the volume of control query traffic rises linearly with the cluster radius while none of the suggested query control schemes are used. For example, when radius r=3, the IELC produces almost twice as much traffic as flooding without proper query control and about 10 - 50 times as much traffic as the most useful query control methods. The best comparative IELC traffic performance by implementing the full flooding and selective flooding is illustrated in Figure 5.10. Under all the same conditions, selective flooding can significantly reduce IELC traffic compared to full

flooding. When (r = 1), selective flooding will generate approximately 20% of the total flooding. As the cluster increases, the impact is even more significant.

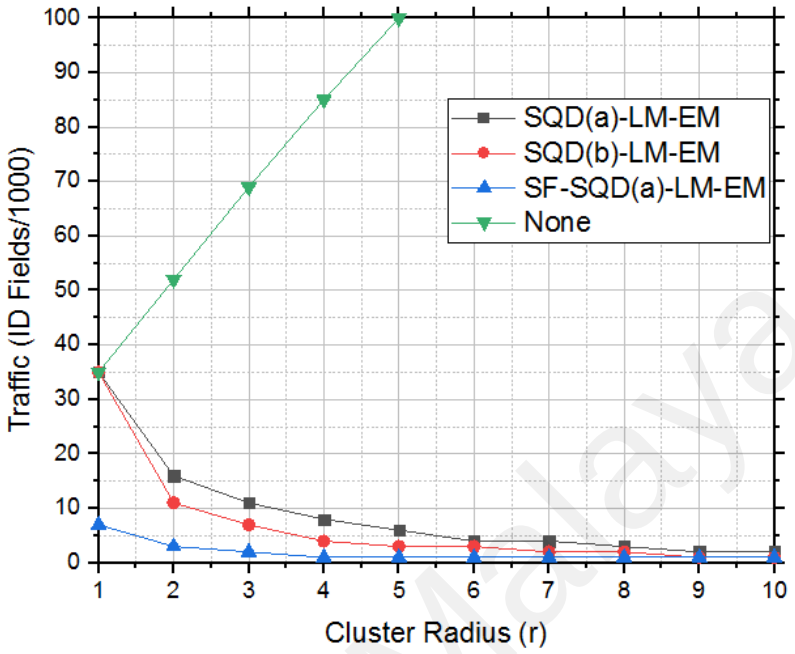


Figure 5.10: IELC Traffic Per Route Query Discovery

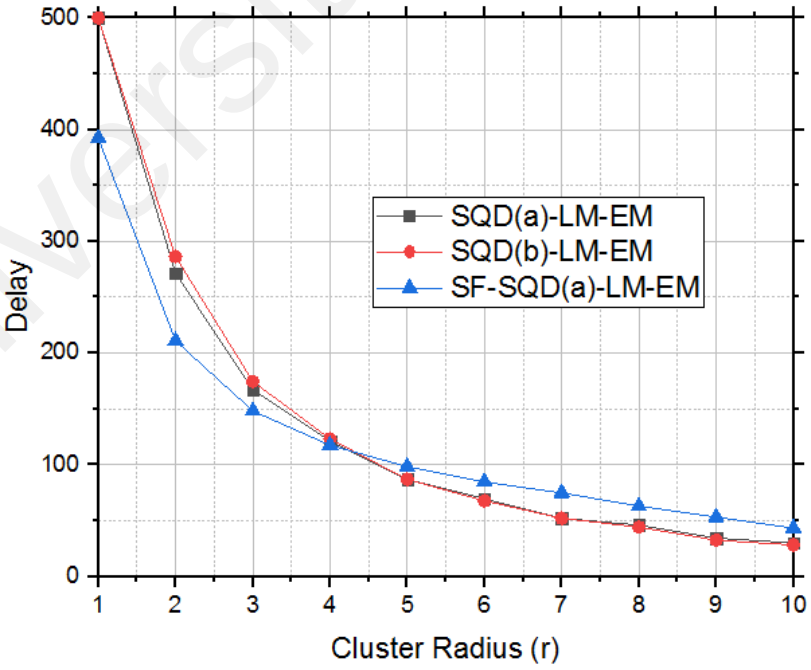


Figure 5.11: Delay of IELC Route Query Discovery

The performance of the CBF was also measured in term of delay while discovering the route queries. As shown in Figure 5.11, the routing query discovery delay is the same as the IELC control traffic, which decreases with the cluster radius. The same factors that oversee the IELC traffic pattern essentially influence this relationship. The transmission time for query propagation is initially minimized because of the aggregate queries' shorter size for the clusters having an extended radius. Second, because of the increased separation gap between border motes, each query encounters least IELC queuing delays. For small cluster size, selective flooding schemes show better delay performance than full flooding schemes but marginally deficient performance in term of delay for enormous cluster radius.

With a small cluster radius, selective flooding results from the decreased queuing latency for every border mote. However, the increased list of assigned internal border mote can be comparatively large within a wider cluster radius, creating additional communication delays, thereby reducing the advantages of minimizing queuing delays, remembering that selective flooding's performance is more favorable when the cluster radius is small. As seen, the full flooding performance in term of rout query responding is little as 1/3 time compared to selective flooding.

5.7 Conclusion and Future work

An undesirable side effect of flooding is the overlapping of query threads. Which may lead to the propagation of redundant and unwanted queries, resulting in excessive resource utilization and may reducing QoS. This chapter introduced and analyzed the smart query detection and mitigation techniques (LM, SQD(a)/SQD(b), EM and SF) which effectively combat the redundant querying, while generating no additional control traffic and requiring negligible computational overhead. Further reduction of the intercluster control traffic can be achieved by preventing thread overlap locally through

selective flooding. When the CBF is configured to minimize total routing control traffic, it is evident that full flooding responds to route queries at least three times faster than a selective flooding implementation. Based on these results, selective flooding may be a suitable platform for the IELC in multiple channel networks where conservation of bandwidth is more important than delay performance. In all other cases, it appears that the simpler full flooding is the preferred query propagation mechanism.

5.7.1 Future Work

The testbed needs to handle additional functionality and more possibilities for research and needs to adapt continually. This section aims to cover how more QoS-related problems can be encouraged and discussed on the IoTs. If the extension of testbeds is an obvious path forward for more diverse IoT-related innovations and protocols, additional attractive aspects are often considered.

5.7.1.1 Making the Testbed Accessible

To build all kinds of test cases, it will be beneficial to make the project open source for the community of IoT developers. This can be done by making the testbed available via the web and monitoring all motes with the correct logging.

5.7.1.2 Intelligent Testbed

One of the next moves will also be to use specialized Machine Learning, and Artificial Intelligence approaches to handle sophisticated IoT network QoS analyses and provide appropriate redundant query recognition systems.

CHAPTER 6: RESULTS AND DISCUSSIONS

6.1 The Implementation Details

This Chapter presents the detailed implementation of our method. The study employs a well-known Contiki Cooja network simulator for formation of redundant flooding queries (Thomson et al., 2016) (Romdhani et al., 2016). Table 6.1 presents a comprehensive specification of the simulation parameters. These parameters are standardized as per IEEE 802.15.4 radio regulation. The study employs a random topology to cater the heterogeneous nature of IoT devices to reflect the capabilities of sensor mote connectivity. The study chooses a network size of 1 to 64 motes as simulation model, deployed randomly under an area of 100-to-300-meter square. The motes are kept as active transmitters and active receivers. Moreover, the study evaluates the simulation model having six different scenarios (i.e., with different interval of traffic, with different number of malicious motes, realistic scenarios, varied mobility speed, varied simulation area, and varied pause time). The study keeps the inner arrival time as exponential. The study explores various levels of network saturation based on traffic intensity that varies from low saturated networks (p_{max} = 0.1) to high saturated networks (p_{max} > 1). Table 6.1 highlights the foundation of suitable assessment parameters in flooding.

Table 6.1: Parameters Values of Simulation Scenario.

Mote Type	Sky Mote
Initial state Energy	100 J
Power (idle state)	31 Mw
Power (receiving state)	35 Mw
Power (sending state)	31 Mw
Power (sleep state)	15 µW
Simulation name	QoS Enabled CBF
Radio medium	UDGM with Distance Loss
Startup Delay	1 ms
Random Seed	123,456 (Default)
Positioning	Random

Topology	Random
Number of Motes	16

The Unit Disk Graph Medium (*UDGM*: distance loss) is the radio medium used for simulation with a transmission range of 50 m and interference range of 100 m. It carries an initial energy of 100 joules. Each device moves randomly with startup delay of one millisecond. Each mote consumes 31 mW energy in an ideal state and 15 μW in the sleeping state. A total of 35 and 31 mW energy is consumed during data receiving and sending states, respectively. Moreover, the transmission delay for high-speed links is insignificant. For example, a 1500-byte packet transmitted over a 155 Mbps STM-1/OC-3 link would take only 0.08 ms. Following formulas are used to calculate the performance metrics and then convert it to percentage accordingly.

$$Throughput = \left(Blog_2(1 + SINR)\right) \times 100$$

$$B = Available\ Bandwidth$$

$$SINR = Signal\ to\ Interference\ Plus\ Noise\ Ratio$$

$$SINR = \frac{P_m g_{n-1}^n}{\sum_{n-1 \neq V}^n P_m g_V^R + \sigma^2}$$

$$P_m = Transmitting\ Power\ From\ node_n\ to\ node_{n-1}$$

$$g_{n-1}^n = channel\ gain\ from\ node_n\ to\ node_{n-1}$$

$$\sigma^2 = Additive\ white\ guassian\ noise$$

Energy Consumption: in our case we have four different powers that is idle power, sleep power, sending power, receiving power.

$$Total\ power = P_{Idle} + P_{sleep} + P_{send} + P_{recieving}$$

$$Energy\ Consumption = \frac{Throughput}{Total\ power}$$

$$E.C = \frac{B.log_2(1 + SINR)}{\sum Total\ power} \times 100$$

$$Delay = \left(\frac{Time\ Recieved - Time\ send}{10^3}\right) \times 100$$

6.2 Results and Discussion

This Section presents the performance evaluation and analysis of the existing techniques i.e., DnC, SLA(Abdelaal et al., 2016), (Alqahtani et al., 2016) and Hy-IoT (Sadek, 2018) with the QCM method. The comparative analysis is based on the tracing and alleviating the redundant (unwanted) reactive flooding. The study considers MDP protocol for Contiki and routing protocol as ad hoc routing protocols. The study performs 60 times simulation to achieve the suitable outcomes of this experimentation considering following six special scenarios:

- Different traffic intervals: this scenario is vital to ensure the suitability of flooding attacks in regulating the defensive strategies under different traffic intervals. The traffic intervals range from 1 to 10 seconds, where 1 second is treated as faster and 10 seconds are considered slower.
- Different number of mischievous nodes: this scenario is suitable in context to analyze the impact of flooding attack over the network. It helps to take suitable action to combat mischievous motes. This study treats motes 2, 6, 10, and 15 as mischievous motes. as described previously the fastest traffic is considered as 1 second.

- Realistic scenario: in this scenario, the study restricts the motes for not transferring the information of routing query simultaneously. Motes are allowed to transfer information of the routing, at different time intervals, randomly set to 1 to 10 seconds.
- Scenario based on a varied mobility speed of motes: The importance of motes mobility motivates us to evaluate the proposed schemes with varied mobility speed. The topology with more speedy motes is more dynamic and vice versa. In this scenario, the mobility speed of motes is kept changing such as 5,10,15,20 and 25 m/seconds as shown in the following table 6.2.

Table 6.2: Configuration Simulation Scenario Parameters with Varied Mobility Speed

Parameter	Value
Number of Nodes	64
Pause Time	1 second
Maximum Speed	5,10,15,20,25 m/sec
Area	200 m^2
Mobility Model	Random Way Point
Positioning	Random
Topology	Random

- Scenario based on a varied simulation area: The area in which motes moves and communicate effects the performance of wireless sensor networks. In this scenario, the simulation area is kept changing such as 100, 150, 200, 250 and 300 m^2 as shown in Table 6.3.

Table 6.3: Configuration Simulation Scenario Parameters with Varied Simulation area

Parameter	Value
Number of Nodes	64
Pause Time	1 second
Maximum Speed	15 m/sec
Area	100,150,200,250, 300 m^2
Mobility Model	Random Way Point
Positioning	Random
Topology	Random

- Scenario based on a varied pause time: Pause time in a scenario also effect performance of wireless sensor networks such as energy consumption, delay and throughput. In this scenario, the varied pause time such as 3, 6, 9, 12 15 seconds is used to find the effect of pause time on proposed scheme as mentioned in table. 6.4.

Table 6.4: Configuration Simulation Scenario Parameters with Varied Pause Time

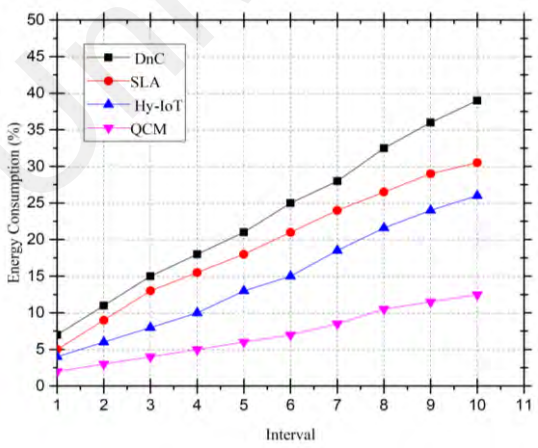
Parameter	Value
Number of Nodes	64
Pause Time	3, 6, 9, 12 15 seconds
Maximum Speed	15 meter/seconds
Area	100 m^2
Mobility Model	Random Way Point
Positioning	Random
Topology	Random

Here, it's worth to mentioned that the sensor motes fallow the random mobility model and have random topology that are positioned randomly in most of the realistic simulation scenarios. This study focused on to find the changing behaviors (values) of six simulations parameters (that is traffic intervals, number of mischievous motes, mobility speed, simulation area and pause time) that have an impact on the performance of the QoS enabled cross-layered clustering (Cluster Based Flooding) protocol.

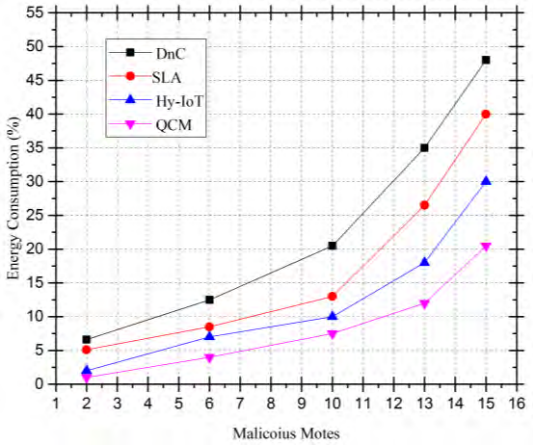
6.2.1 Average Energy Consumption

Figure 6.1 presents the average energy consumption and number of malicious motes having different intervals of time. Figure 6.1(a) depicts the comparison of the proposed QCM approach with DnC, SLA, and Hy-IoT methods in context of reduction in the average energy consumption. The proposed method can detect flooder motes and can detach them from the network, thus reducing the energy consumption levels (arising under redundant (unwanted) flooding attacks. While we notice the average energy consumption of prevailing DnC, and SLA as about 21 and 18%, respectively, under 1–5 s intervals, resulting in rise of this ratio continuously as the interval increases. Nevertheless, the QCM consumes energy under 6% compared to the 13% of the prevailing Hy-IoT method.

Clearly, the average consumption of energy stays directly associated to the time interval for all methods, the consumption of energy increases with the increase in time interval. The study noticed that the flooder mote consumed highest energy since it directly transmits unwanted queries when it detects the communication activity over the network. Figure 6.1(b) presents the average energy consumption having different number of mischievous motes stated as 2, 6, 10, 15.



(a)



(b)

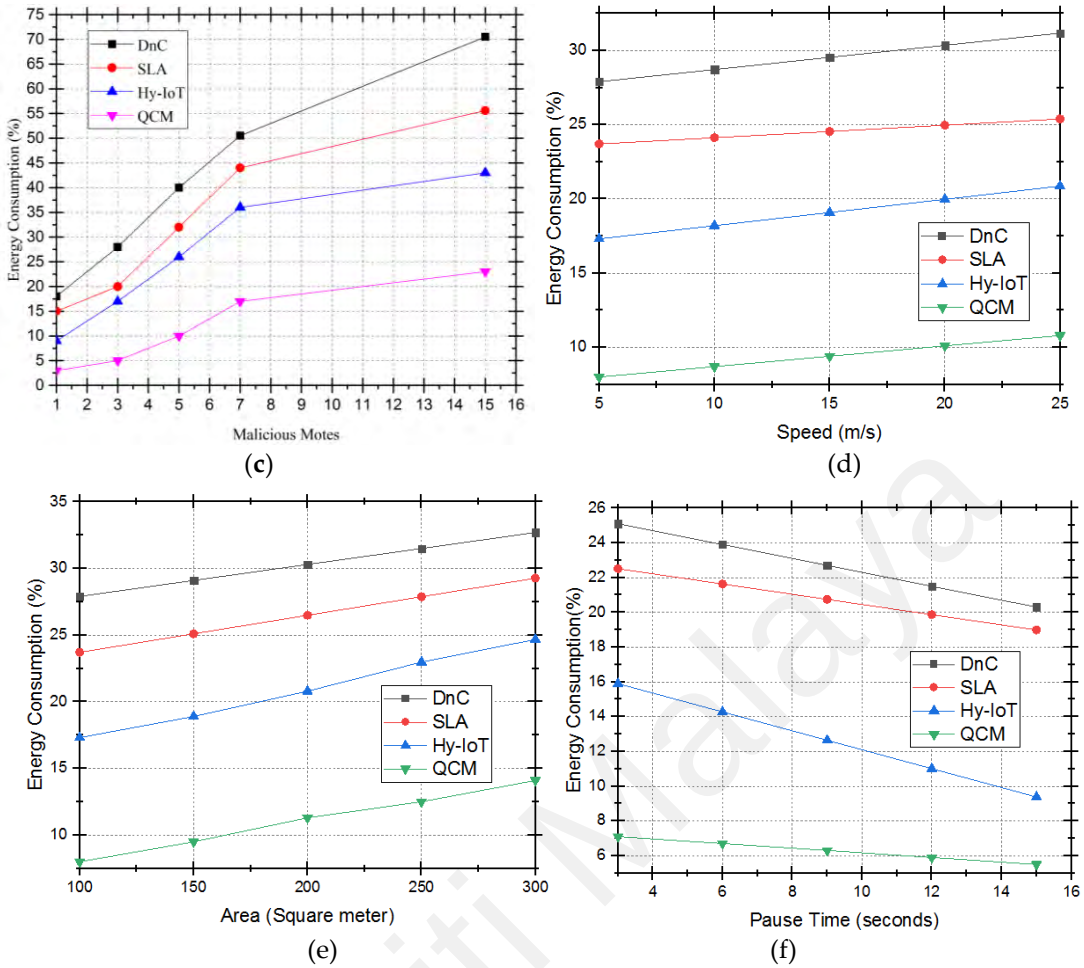


Figure 6.1: Average Energy Consumption in Context of Different Traffic Intervals with (a) Malicious Motes (b), Malicious Motes with Realistic Condition (c), Different Mobility Speed (d) Different Simulation Area (e) Different Pause Time (f)

This study conducted a realistic analysis of QCM to investigate the level of mischievous motes in context of network expansion under flooding. It can be notice from the results that energy consumption is increased with the presence of malicious motes. For instance, the mote 2 consumes energy approximately 5% and 8% for SLA and DnC methods respectively. The malicious mote 15 consumes energy approximately 48% and 40% for both methods. Thus, the presentation of QCM helps us to decrease energy to about 2%, 4%, 20% at malicious modes 2, 6, 15 respectively. it can be notice that the

proposed QCM outperformed the existing methods approximately 3% and 8% with malicious motes 2 and 6 respectively and gradually increase to 32% at mote 15.

The QCM can reduce the energy level consumption by halting the flooder mote (under unwanted transmission to its active state), by significant treating the flooding attacks.

Figure 6.1(c) presents average energy consumption addressed with realistic condition and malicious modes. The random dissipation of packets related to route query are referred under the realistic condition at different intervals of time. During the time of unwanted flooding attack, the simulation of realistic condition is very important to know about efficiency of simulated methods. The results describe that average energy consumption of DnC & SLA, under malicious mote 1 is about 18% and 15% respectively. This consumption elevates to approximately 70% and 56% under malicious mote 15, whereas the QCM addresses the consumption reducing to 3% and 24% approximately under malicious motes 1 and 15 respectively. Moreover, the energy consumption is about 6% lower than the conventional Hy-IoT at malicious mote 1 and 20% at malicious mote 15 having the same scenario.

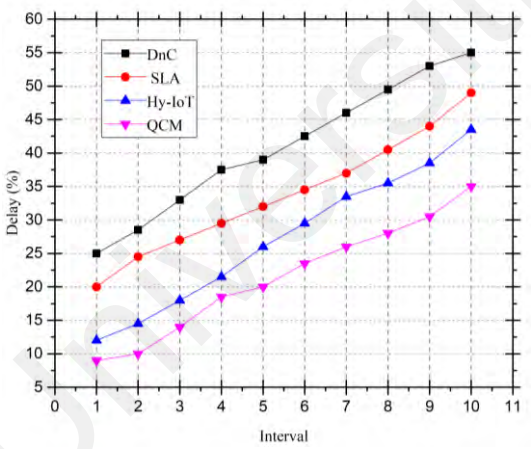
The performance of the QCM is analyzed with varied mobility speeds, area and pause time of motes as mentioned in Figure 6.1(d, e and f). Mote's moments increase with increase in speed causes more energy consumption. Connection breaking ratio also increase with increase in speed due to quickly change in the position and intermediate nodes. These phenomena also increase in delay and decrease in throughput of QCM and other protocols with increase in mobility speed of motes. In Figure 6.1(d), QCM has outperformed the DNC, SLA and Hy-IoT by having minimum increase in energy consumption of 2%. QCM has lowest 10.8% average energy consumption. In case of increasing simulation area, QCM has comparatively consume less energy of 11.08% as

mentioned in Figure 6.1(e). The increase in pause time motivates toward lower energy consumption in wireless sensor networks. The QCM has lowest possible energy consumption with varied pause time scenario as shown in Figure 6.1(f). The increase in pause time causes less connection breaking ratio and the topology is more static.

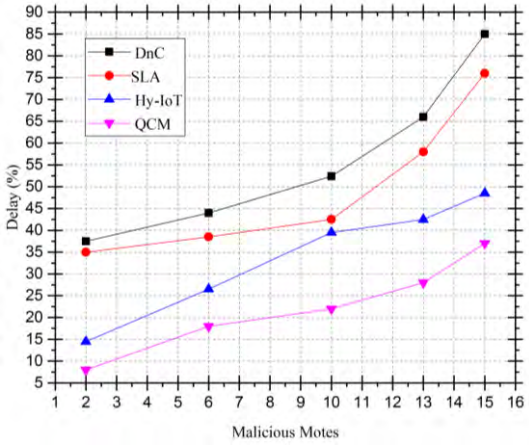
These results of energy consumption with various simulations scenarios proved that the proposed QCM significantly improved the performance of the network.

6.2.2 Traffic Delay

The proximity of traffic delay to different cases including malicious motes, time interval, malicious motes (realistic condition), varied mobility speed, varied simulation area and varied pause time Figures 6.2 (a–f), respectively. In Figure 6.2(a), QCM performed significant as against the existing methods DnC, SLA, and Hy-IoT demonstrating minimum traffic delays. The study investigated the QCM as a good detector, pause manager, and detacher for flooding motes from the network.



(a)



(b)

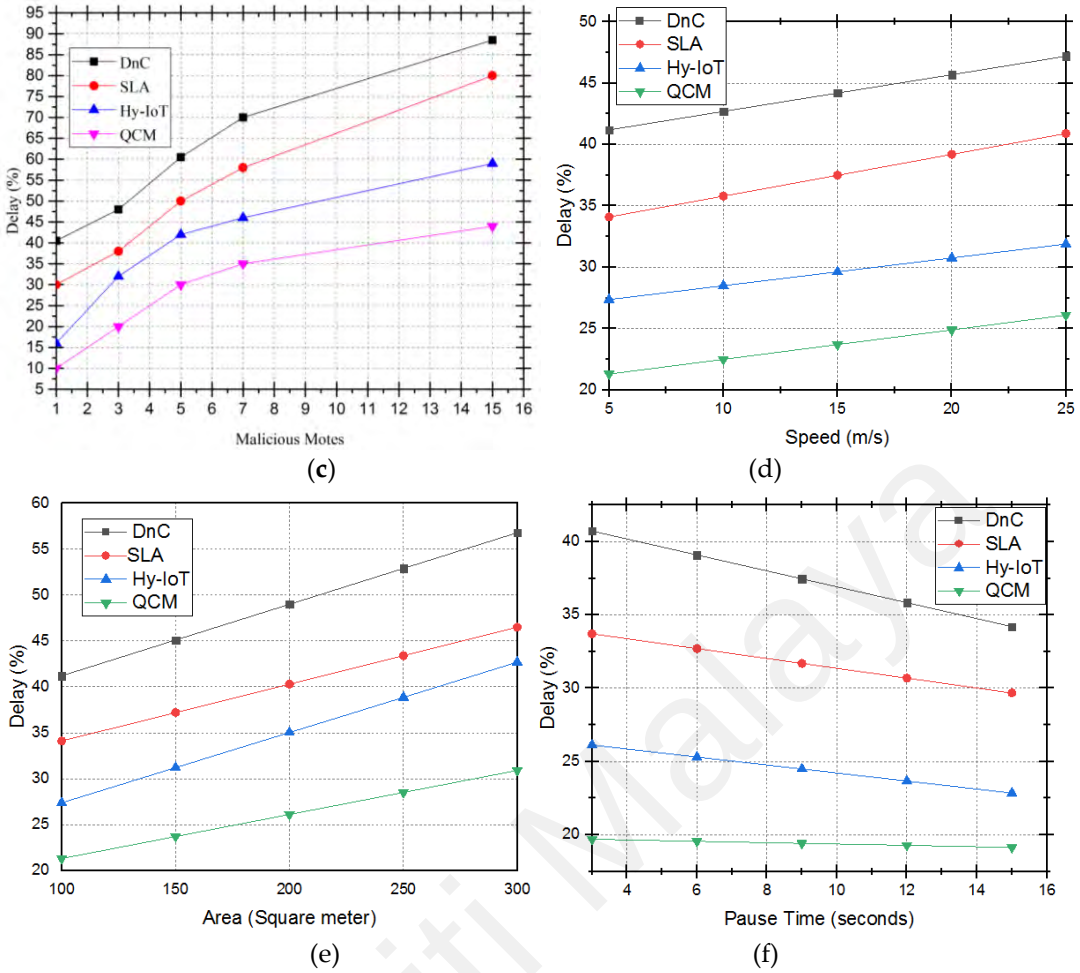


Figure 6.2: Delay in Context of Different Traffic Intervals with (a) Malicious Motes (b), Malicious Motes with Realistic Condition (c), Different Mobility Speed (d) Different Simulation Area (e) Different Pause Time (f).

These measures greatly improved the performance. Further, this could also remove the redundant (unwanted) queries by disengaging the flooding motes. DnC and SLA exhibited a traffic delay of 26% and 20%, respectively for interval 1, as presented in Figure 6.2(a). The proposed algorithm reduced the traffic delay to about 10% in the network at the same interval. QCM could achieve a drop in traffic delay about 4% lower than the conventional Hy-IoT. Thus, the percentage increases with the increase in interval, since the percentage is directly associated with the interval.

Figure 6.2(b) presents the traffic delay scenario in context of increase in number of malicious motes. Under normal reactive flooding scenario, the delay in traffic for algorithms DnC and SLA are approximately 37% and 35% respectively. Any increase of number of malicious motes, we can notice that delay in traffic also increases. For malicious mote 15, the largest delay in traffic for DnC and SLA are observed as about 85% and 75% respectively. The QCM algorithm performed significant as compared with existing Hy-IoT for malicious motes 2 and 15, and it lowers the delay to approximately 7% and 37% gradually. With the same malicious motes 2 and 15, the QCM performs comparatively better with respect to delay in traffic having about 15% and 49% delay respectively. The study observed that the link waiting time was largely decreased by the QCM algorithm with respect to decrease in traffic delay.

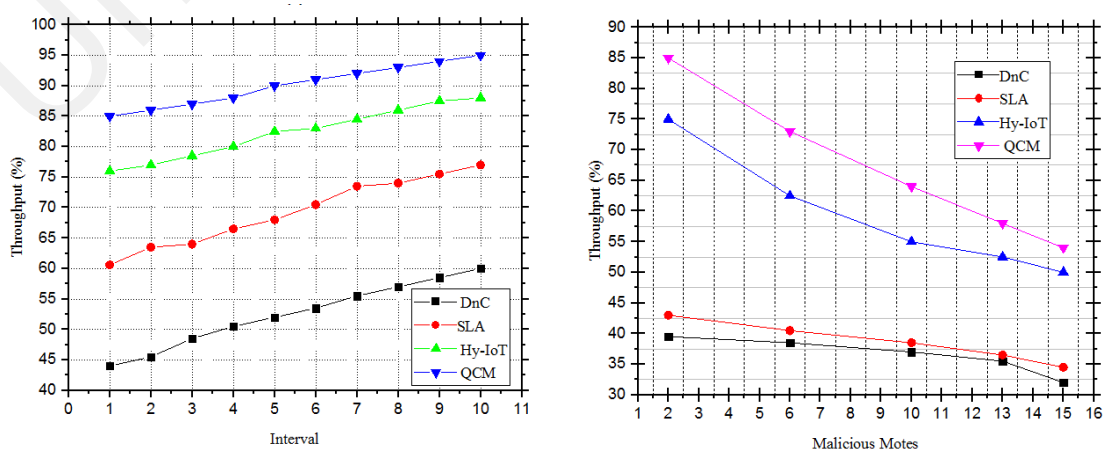
Figure 6.2(c) presents the realistic network scenario in context of evaluating the performance of existing and flooding techniques under increasing number of malicious motes. It is observed that DnC and SLA depicted a traffic delay of about 45% and 30%, at malicious mote, respectively. Whereas the delay was noticed to be increased to approximately 89% and 81%, for DnC and SLA, under malicious mote 15, respectively. we noticed the outperformance of the QCM attaining approximately 11% and 45% of traffic delays under malicious motes 1 and 15, respectively. This pointed out a profound decrease in traffic delay achieved by QCM by approximately 6% and 15% compared with prevailing Hy-IoT under motes 1 and 15, respectively.

The QCM and other protocols such as DnC, SLA, and Hy-IoT are thoroughly analyzed using varied mobility speed, varied simulation area and varied pause time to find the effect on their traffic delay. Although the traffic delay of QCM and other variants (DnC, SLA, and Hy-IoT) increases with increase in mobility speed as shown in Figure 6.2(d) due to increase in connection breakup ratio and topology changes between the

sender and destination. QCM outperform the DnC (44.2%), SLA (37.5%), and Hy-IoT (29.63%) by minimum average delay of 23.7%. In case of varied simulation area, the QCM shows lowest average delay of 26.1% as compared to DnC (49%), SLA (40.3%), and Hy-IoT (35.03%) as shown in Figure 6.2(e). Traffic delays increase with increase in simulation area as the distance between communicating motes increases. The varied pause time scenario shows that QCM has lower average traffic delay of 19.14% as compared to DnC (34.21%), SLA (29.68%), and Hy-IoT (22.85%) as mentioned in Figure 6.2(f). QCM has lowest average traffic delay to its mechanism that effectively control the redundant and unwanted flooding.

6.2.3 Throughput

The study also evaluated the performance of algorithms to assess the network throughput as shown in Figure 6.3. The throughput is assessed in context of six scenarios, such as interval of time, increasing in the number of malicious motes, malicious motes with realistic network conditions, varied mobility speed, varied simulation area and varied pause time. In Figure 6.3(a), network throughput of DnC and SLA are decreases up to approximately (44% and 61%), respectively at an interval of 1 s. While, with the QCM the throughput boosts to approximately 85%, this result is 9% better as compared to existing Hy-IoT at the same time interval 1 s.



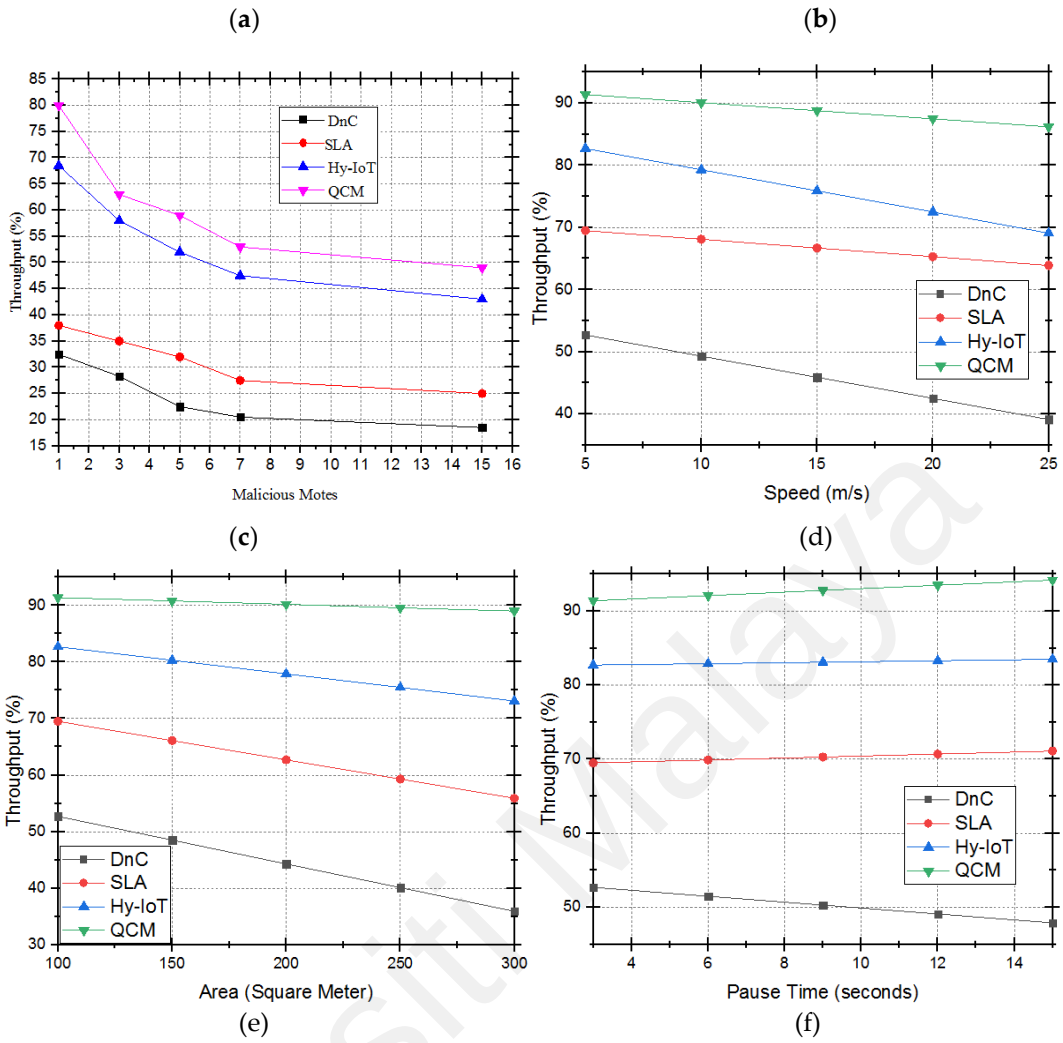


Figure 6.3: Throughput in Context of Different Traffic Intervals with (a) Malicious Motes (b), Malicious Motes with Realistic Condition (c), Different Mobility Speed (d) Different Simulation Area (e) Different Pause Time (f).

According to the result as shown, the performance of network throughput is inversely proportional to the time interval, as throughput decreases with the increase in the interval. Flooder motes keep engaged the transmission link, which generates unwanted and redundant routing request queries and directly affects the QoS of the IoT networks.

Figure 6.3(b) presents the throughput of QCM and existing algorithms DnC, SLA, and Hy-IoT , assessed at increasing the number of malicious motes. At first, under malicious mote 2, the throughput of DnC and SLA is about 39% and 44% that gradually lowers to 30% and 35% respectively for malicious mote 15. The QCM achieved 86% throughput at malicious mote 2, that gradually decreased to about 53% at malicious mote 15. These findings are about 9% and 6% significant than the existing Hy-IoT at malicious motes 2 and 15, respectively. The significance in results depicted the prompt link access to different motes during the unwanted query attacks.

The study also figured out the throughput of QCM and other algorithms DnC, SLA, and Hy-IoT under realistic network conditions by observing the increments of number of malicious motes, demonstrated in Figure 6.3(c). The findings showed that the network throughput remained 34% and 37% for DnC and SLA, respectively under malicious mote 1. This performance dropped to about 17% and 25%, respectively for malicious mote 15. From the results it is noticed that 80% and 49% throughput achieved by the QCM for malicious motes 1 and 15, respectively. QCM could achieve 12% and 5% throughput significant than the existing Hy-IoT at malicious motes 1 and 15, respectively under realistic conditions.

The Figure 6.3(d) shows that throughput of QCM exceeds by having 88.8 % as compared to other variants such as DnC (45.9%), SLA (66.7%), and Hy-IoT (75.9%) with increase in mobility speed. In case of varied simulation area, the increase in throughput for QCM is very higher throughput of 90.2%. Other variants have comparatively low throughput such as DnC (44.3%), SLA (62.7%), and Hy-IoT (77.9%) as mentioned in Figure 6.3(e). The throughput of these variants increases with increase in pause time as shown in Figure 6.3(f) since the topology is more static with higher pause time. Here, the QCM outperformed by having higher average throughput of 92.8% as compared to

DnC (50.3%), SLA (70.7%), and Hy-IoT (83.1%). The efficient underlying mechanism of QCM by controlling unwanted queries and providing fast link for communication motivates to have higher throughput as compared to other variants.

6.3 Performance Evaluation and Validation of QCM

This Section demonstrates the statistical evaluation (performance) of various query control methods in context of reducing energy consumption, cost and network flooding. Particularly, this section measure the performance of QCM algorithm. Using statistical measurements, we draw the significant conclusions achieved by the QCM algorithm against the existing algorithms i.e., DnC, SLA, and Hy-IoT. This performance was evaluated by eliminating redundant flooding queries. We evaluated the inferential statistics of different algorithms under six different scenarios i.e at different traffic intervals, malicious mote, malicious mote with realistic condition, simulation areas, pause time, and mobility speed. The results demonstrated that the QCM algorithm performed significant as compared to existing algorithms and the significant probability remained less than 0.05 indicating the excellent performance of QCM algorithm. We ensured that QCM achieves 95% confidence interval, demonstrating that QCM algorithm perform robust as compared to other existing algorithms.

6.3.1 Evaluation Methodology

This study is relying on the hypothesis that the proposed **QCM (Query Control Mechanism)** algorithm (Khan, F. A., Noor, R. M., Mat Kiah, M. L., Noor, N. M., Altowaijri, S. M., & Rahman, A. U., 2019) outperforms the other existing algorithms, i.e., DnC, SLA, and Hy-IoT in context of QoS-enabled layered clustering, under reactive flooding for IoT devices.

This study incorporates two research hypotheses for inferential analysis,

1. Null hypothesis H_0 ($\mu_{2-}\mu_1 = 0$): The null hypothesis states that the study could not achieve a significant difference in the performance evaluation of existing and the proposed method (DnC, SLA, Hy-IoT, and QCM).
2. Alternative hypothesis H_1 ($\mu_{2-}\mu_1 > 0$): The alternative hypothesis states that the study could achieve a significant difference in the performance evaluation of existing and the proposed method (DnC, SLA, Hy-IoT, and QCM).

Next, the study analyzed the T-test and ANOVA statistics for hypothesis testing. Suppose the sample mean difference is

$$\bar{d} = \mu_{2-}\mu_1 \tag{4}$$

where μ_1 and μ_2 are the sample means of dataset of first and second algorithms respectively. Sample standard deviation

$$S_D = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{d})^2} \tag{5}$$

Here, data points are $x_1, x_2, x_3, \ , \ , x_N$ refer to the data points of results in two comparable algorithms. Paired Sample T-test:

$$T = \frac{\overline{d} - 0}{S_D/\sqrt{n}} \tag{3}$$

Here, *n* depicts the observations. The study evaluates that the probability value has a significant number for the two hypotheses. The probability value is ensured to achieve the 95% confidence interval. The statistical test measures the probability value based on the data points. As a benchmark value, we kept the confidence interval has 0.05 for statistical significance. Any algorithm that achieves the probability value larger than 0.05 is considered insignificant in terms of inferential analysis of its data points.

In addition, the study also performed an "ANOVA test" (Ahmad et al., 2016, 2017) to investigate performance measure of algorithms. The ANOVA test contains the following features,

Mean square for samples,

$$MSR = \frac{SSR}{k-1} \tag{6}$$

Similarly, the mean square for error,

$$MSE = \frac{SSE}{n-k} \tag{7}$$

Now the F statistics becomes

$$F = \frac{MSR}{MSE} \tag{8}$$

The study investigates the performance of different QoS-enabled layered-based clustering algorithms for reactive flooding in the Internet of Things with the following measures.

1. **Inferential Analysis in Terms of Energy Consumption**
    - **a.** Having different intervals of traffic
    - **b.** With malicious mote
    - **c.** Having malicious mote with a realistic condition
    - **d.** With different simulation area
    - **e.** With respect to pause time
    - **f.** With respect to mobility speed
2. **Inferential Analysis in Terms of Delay**
    - **a.** With different intervals of traffic
    - **b.** With malicious mote
    - **c.** With malicious mote under a realistic condition
    - **d.** Having different simulation area
    - **e.** With respect to pause time

f. With respect to mobility speed

3. Inferential Analysis in Terms of Throughput

a. Having different intervals of traffic

b. With malicious mote

c. With malicious mote under a realistic condition

d. With different simulation area

e. With respect to pause time

f. With respect to mobility speed

The significance value, represented by probability value "P" is a statistical measure for the performance evaluation of QCM and existing algorithms in terms of accepting or rejecting the null and alternative hypothesis.

6.4 Results

This section describes the inferential analysis of experimental results related to the performance evaluation and validation of the **QCM (Query Control Mechanism)** algorithm (Khan, F. A., Noor, R. M., Mat Kiah, M. L., Noor, N. M., Altowaijri, S. M., & Rahman, A. U., 2019). which present the rejection of the Null hypothesis and acceptance of the alternative hypothesis since the **QCM** algorithm outperforms (95% confidence interval) the existing algorithms, i.e., DnC, SLA, and Hy-IoT for QoS-enabled layered-based clustering for reactive flooding in the Internet of Things.

Case 1: Inferential Analysis in Terms of Energy Consumption

Figure 6.4 presents the average energy consumption and number of malicious motes having different intervals of time depicting the comparison of the proposed QCM approach with DnC, SLA, and Hy-IoT methods in context of reduction in the average

energy consumption. The QCM can detect flooder motes and can detach them from the network, thus reducing the energy consumption levels (arising under redundant (unwanted) flooding attacks. While, it is noticed that the average energy consumption of prevailing DnC, and SLA as about 21 and 18%, respectively, under 1–5 s intervals, resulting in rise of this ratio continuously as the interval increases. Nevertheless, the QCM consumes energy under 6% compared to the 13% of the prevailing Hy-IoT method.

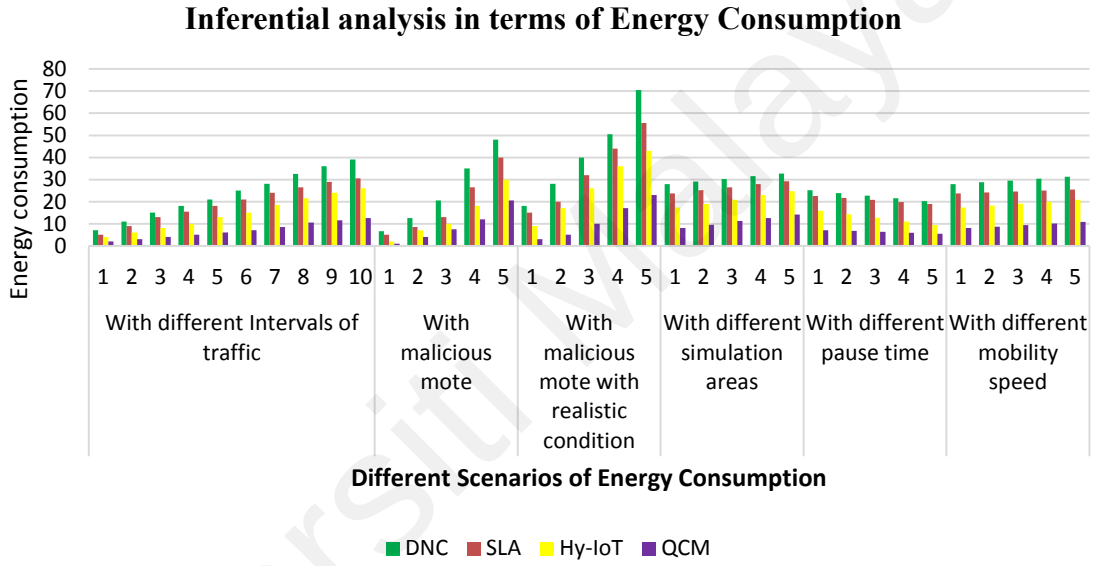


Figure 6.4: Energy Consumption with Respect to Different Scenarios.

Clearly, the average consumption of energy stays directly associated to the time interval for all methods, the consumption of energy increases with the increase in time interval. The study noticed that the flooder mote consumed highest energy since it directly transmits unwanted curries when it detects the communication activity over the network.

Table 6.6 demonstrates the inferential analysis of data related to **QCM** and other existing algorithms. It can be seen that the statistically significant value **P** is less than our chosen confidence interval of 0.05 which is evidence that **QCM** algorithm

outperforms the existing algorithms. Hence, Null hypothesis is rejected and **QCM** achieves the significant prediction value in the desired confidence interval.

Table 6.6: Inferential Analysis of the QCM Algorithm in Terms of Energy Consumption Scenarios.

“Energy consumption” with "different intervals of traffic"			
Method	***"QCM", "DNC"***	***"QCM", "SLA"***	***"QCM", "Hy-IoT"***
P. Correlation	0.995	0.985	0.998
t Stat	−7.234	−7.658	−5.902
Prob.(1-tail)	**0.000**	**0.000**	**0.000**
t Critic. (1-tail)	1.8331	1.833	1.833
Prob.(2-tail)	**0.000**	**0.000**	**0.000**
t Critic. (2-tail)	2.262	2.262	2.262
“Energy consumption” with "malicious mote"			
Method	***"QCM", "DNC"***	***"QCM", "SLA"***	***"QCM", "Hy-IoT"***
P. Correlation	0.991	0.990	0.997
t Stat	−3.691	−3.080	−2.910
Prob.(1-tail)	**0.010**	**0.018**	**0.021**
t Critic. (1-tail)	2.131	2.131	2.131
Prob.(2-tail)	**0.020**	**0.0369**	**0.043**
t Critic. (2-tail)	2.776	2.776	2.776
“Energy consumption” with "malicious mote with realistic conditions"			
Method	***"QCM", "DNC"***	***"QCM", "SLA"***	***"QCM", "Hy-IoT"***
P. Correlation	0.988	0.997	0.985
t Stat	−5.475	−5.744	−5.700
Prob.(1-tail)	**0.002**	**0.002**	**0.002**
t Critic. (1-tail)	2.131	2.131	2.131
Prob.(2-tail)	**0.005**	**0.004**	**0.004**
t Critic. (2-tail)	2.776	2.776	2.776
“Energy consumption” with "different simulation areas"			
Method	***"QCM", "DNC"***	***"QCM", "SLA"***	***"QCM", "Hy-IoT"***
P. Correlation	0.998	0.998	0.995
t Stat	82.405	142.770	35.598
Prob.(1-tail)	**0.000**	**0.000**	**0.000**
t Critic. (1-tail)	2.131	2.131	2.131
Prob.(2-tail)	**0.000**	**0.000**	**0.000**
t Critic. (2-tail)	2.776	2.776	2.776
“Energy consumption” with "pause time"			
Method	***"QCM", "DNC"***	***"QCM", "SLA"***	***"QCM", "Hy-IoT"***
P. Correlation	1	1	1
t Stat	28.991	42.573	7.289
Prob.(1-tail)	**0.000**	**0.000**	**0.000**
t Critic. (1-tail)	2.131	2.131	2.131
Prob.(2-tail)	**0.000**	**0.000**	**0.001**
t Critic. (2-tail)	2.776	2.776	2.776
“Energy consumption” with "mobility speed"			
Method	***"QCM", "DNC"***	***"QCM", "SLA"***	***"QCM", "Hy-IoT"***
P. Correlation	1	1	1

t Stat	237.352	76.519	72.124
Prob.(1-tail)	**0.000**	**0.000**	**0.000**
t Critic. (1-tail)	2.131	2.131	2.131
Prob.(2-tail)	**0.000**	**0.000**	**0.000**
t Critic. (2-tail)	2.776	2.776	2.776

This study conducted a realistic analysis of QCM to investigate the level of mischievous motes in context of network expansion under flooding. It is evident from the results that energy consumption is increased with the presence of malicious motes. For instance, the mote 2 consumes energy approximately 5% and 8% for SLA and DnC methods respectively. The malicious mote 15 consumes energy approximately 48% and 40% for both methods. Thus, the presentation of QCM helps us to decrease energy to about 2%, 4%, 20% at malicious modes 2, 6, 15 respectively. In addition, the *QCM* achieved excellent probability "P-value" at different simulation areas, pause time, and mobility speed. In all the above cases, the algorithm outperforms the other existing algorithms at 95% confidence interval.

Table 6.7 presents the ANOVA test statistics of the QCM algorithm compared with other algorithms. It can be found here that "F statistics" values are sufficiently larger than "F critical values". In addition, the "P values" are less than 0.05, which achieves our 95% confidence interval, showing that the QCM algorithm outperforms the existing algorithms evaluated through inferential analysis.

Table 6.7: ANOVA statistics in terms of "energy consumption" scenarios

"Energy consumption" with "different intervals of traffic"						
Variation	*SS*	*Df*	*MS*	*F*	*P-value*	*F crit*

Between Groups	1454.170	3	484.723	7.408	0.000	2.866
Within Groups	2355.439	36	65.428			
Total	3809.609	39				
“Energy consumption” with "malicious mote"						
Variation	*SS*	*Df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	673.109	3	224.369	4.334	0.029	3.238
Within Groups	2690.596	16	168.162			
Total	3363.705	19				
“Energy consumption” with "malicious mote with realistic conditions"						
Variation	*SS*	*Df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	2399.974	3	799.991	3.349	0.045	3.238
Within Groups	3821.948	16	238.871			
Total	6221.922	19				
“Energy consumption” with "different simulation areas"						
Variation	*SS*	*Df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	1046.381	3	348.793	60.448	0.000	3.238
Within Groups	92.321	16	5.770			
Total	1138.702	19				
“Energy consumption” with "pause time"						
Variation	*SS*	*Df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	860.920	3	286.973	91.260	0.000	3.238
Within Groups	50.313	16	3.144			
Total	911.233	19				
“Energy consumption” with "mobility speed"						
Variation	*SS*	*Df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	1116.191	3	372.063	279.366	0.000	3.238
Within Groups	21.309	16	1.331			
Total	1137.5	19				

Case 2: Inferential Analysis in Terms of Delay

The proximity of traffic delay to different cases including malicious motes, time interval, malicious motes (realistic condition), varied mobility speed, varied simulation area and varied pause time respectively. QCM performed significant as against the existing methods DnC, SLA, and Hy-IoT demonstrating minimum traffic delays. The study investigated the QCM as a good detector, pause manager, and detacher for flooding motes from the network. These measures greatly improved the performance. Further, we could also remove the redundant (unwanted) queries by disengaging the flooding motes. DnC and SLA exhibited a traffic delay of 26% and 20%, respectively for interval 1, as presented in Figure 6.2(a). The proposed algorithm reduced the traffic delay to about 10% in the network at the same interval. QCM could achieve a drop in traffic delay about 4% lower than the conventional Hy-IoT. Thus, the percentage increases with the increase in interval, since the percentage is directly associated with the interval.

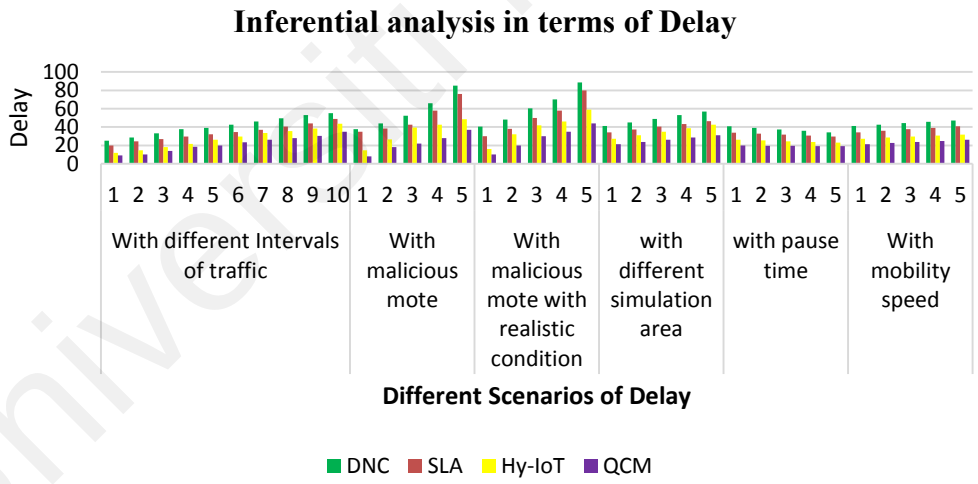


Figure 6.5: Delay with Different Intervals of Traffic

Figure 6.5 depicts the "delay" in context to different scenarios, i.e., various traffic intervals, malicious mote, with realistic conditions, different simulation areas, pause time, and mobility speed. The significant performance can be notice achieved by **QCM** having lowest delay as against other existing algorithms.

Table 6.8 portrays the inferential analysis of data points of **QCM** and other existing algorithms. A statistically significant **P** value can be noticed, lower than chosen confidence interval of 0.05, depicting that the **QCM** algorithm performed excellent as against the existing algorithms. Thus, we can reject the Null hypothesis.

Table 6.8: Inferential analysis in terms of "delay" scenarios.

"Delay" with "different intervals of traffic"			
Method	***"QCM", "DNC"***	***"QCM", "SLA"***	***"QCM", "Hy-IoT"***
P. Correlation	0.994	0.988	0.996
t Stat	−35.043	−28.821	−8.856
Prob.(1-tail)	**0.000**	**0.000**	**0.000**
t Critic. (1-tail)	1.833	1.833	1.833
Prob.(2-tail)	**0.000**	**0.000**	**0.000**
t Critic. (2-tail)	2.262	2.262	2.262
"Delay" with "malicious mote"			
Method	***"QCM", "DNC"***	***"QCM", "SLA"***	***"QCM", "Hy-IoT"***
P. Correlation	0.971	0.941	0.960
t Stat	−8.753	−7.964	−5.894
Prob.(1-tail)	**0.000**	**0.000**	**0.002**
t Critic. (1-tail)	2.131	2.131	2.131
Prob.(2-tail)	**0.000**	**0.001**	**0.004**
t Critic. (2-tail)	2.776	2.776	2.776
"Delay" with "malicious mote with realistic conditions"			
Method	***"QCM", "DNC"***	***"QCM", "SLA"***	***"QCM", "Hy-IoT"***
P. Correlation	0.978	0.971	0.994
t Stat	−11.515	−7.200	−7.656
Prob.(1-tail)	**0.000**	**0.000**	**0.000**
t Critic. (1-tail)	2.131	2.131	2.131
Prob.(2-tail)	**0.000**	**0.001**	**0.001**
t Critic. (2-tail)	2.776	2.776	2.776
"Delay" with "different simulation areas"			
Method	***"QCM", "DNC"***	***"QCM", "SLA"***	***"QCM", "Hy-IoT"***
P. Correlation	1	1	1
t Stat	21.593	28.688	8.831
Prob.(1-tail)	**0.000**	**0.000**	**0.000**
t Critic. (1-tail)	2.131	2.1318	2.131
Prob.(2-tail)	**0.000**	**0.000**	**0.000**
t Critic. (2-tail)	2.776	2.776	2.776
"Delay" with "different pause time"			
Method	***"QCM", "DNC"***	***"QCM", "SLA"***	***"QCM", "Hy-IoT"***
P. Correlation	1	1	1
t Stat	17.131	19.964	10.544
Prob.(1-tail)	**0.000**	**0.000**	**0.000**
t Critic. (1-tail)	2.131	2.131	2.131
Prob.(2-tail)	**0.000**	**0.000**	**0.000**
t Critic. (2-tail)	2.776	2.776	2.776
"Delay" with "different mobility speed"			

Method	***"QCM", "DNC"***	***"QCM", "SLA"***	***"QCM", "Hy-IoT"***
P. Correlation	1	1	1
t Stat	96.637	39.032	119.804
Prob.(1-tail)	**0.000**	**0.000**	**0.000**
t Critic. (1-tail)	2.131	2.131	2.131
Prob.(2-tail)	**0.000**	**0.000**	**0.000**
t Critic. (2-tail)	2.776	2.776	2.776

Table 6.9 presents the ANOVA test statistics of the QCM algorithm compared with other algorithms. It is viable that "F statistics" values are sufficiently larger than "F critical values". In addition, the "P values" are less than 0.05, which achieves our 95% confidence interval, showing that the QCM algorithm outperforms the existing algorithms evaluated through inferential analysis.

Table 6.9: ANOVA statistics in terms of "Delay" scenarios

"Delay" with "different intervals of traffic"						
Variation	*SS*	*Df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	1454.171	3	484.723	7.408	0.000	2.866
Within Groups	2355.439	36	65.42886			
Total	3809.61	39				
"Delay" with "malicious mote"						
Variation	*SS*	*Df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	673.1095	3	224.369	4.334	0.029	3.238
Within Groups	2690.596	16	168.162			
Total	3363.706	19				
"Delay" with "malicious mote with realistic conditions"						
Variation	*SS*	*Df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	2399.974	3	799.991	3.349	0.045	3.238
Within Groups	3821.948	16	238.871			
Total	6221.922	19				
"Delay" with "different simulation area"						

Variation	*SS*	*Df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	1380.523	3	460.174	16.271	0.000	3.238
Within Groups	452.489	16	28.280			
Total	1833.012	19				
"Delay" with "different pause time"						
Variation	*SS*	*Df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	945.079	3	315.026	115.367	0.000	3.238872
Within Groups	43.69	16	2.730			
Total	988.769	19				
"Delay" with "mobility speed"						
Variation	*SS*	*Df*	*MS*	*F*	***P-value***	*F crit*
Between Groups	1206.208	3	402.069	81.87849	0.000	3.238
Within Groups	78.569	16	4.910			
Total	1284.777	19				

Case 3: Inferential Analysis in Terms of Throughput

Network throughput of DnC and SLA are decreases up to approximately (44% and 61%), respectively at an interval of 1 s. While with the QCM the throughput boosts to approximately 85%, this result is 9% better as compared to existing Hy-IoT at the same time interval 1 s. According to the result as shown in figure 6.6, the performance of network throughput is inversely proportional to the time interval, as QoS decreases with the increase in the interval. Flooder motes keep engaged the transmission link, which generates unwanted and redundant routing request queries and directly affects the QoS of the IoT networks.

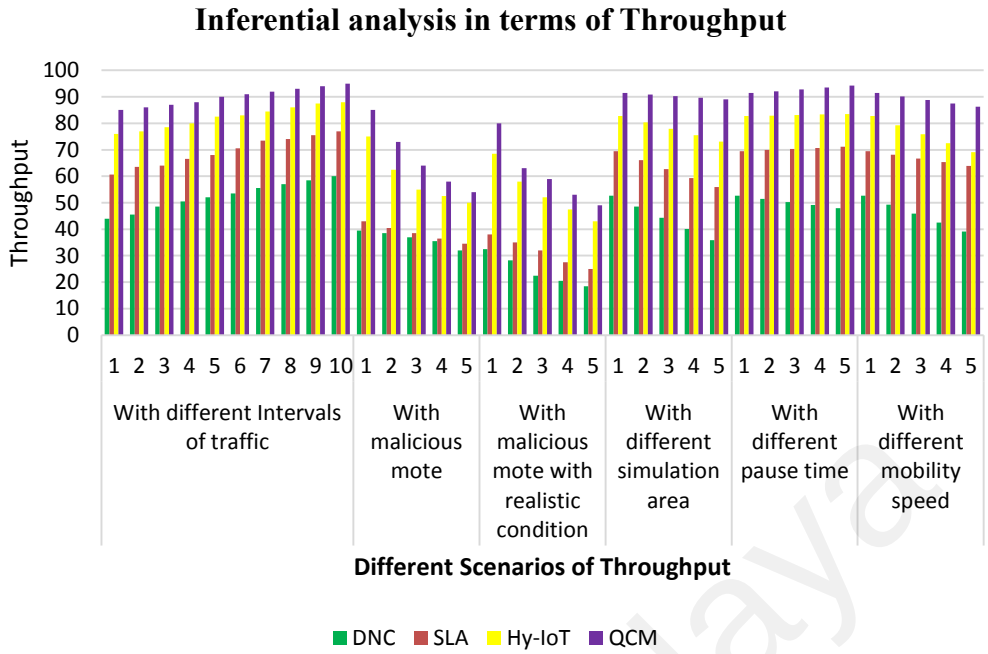


Figure 6.6. Throughput with Different Intervals of Traffic.

Table 6.10 portrays the inferential analysis of data points of **QCM** and other existing algorithms. A statistically significant **P** value can be noticed, lower than chosen confidence interval of 0.05, depicting that the **QCM** algorithm performed excellent as against the existing algorithms. Thus, we can reject the Null hypothesis.

Table 6.10. Inferential Analysis in Terms of "Throughput" Scenarios

"Throughput" with "different intervals of traffic"			
Method	***"QCM", "DNC"***	***"QCM", "SLA"***	***"QCM", "Hy-IoT"***
P. Correlation	0.993	0.992	0.997823573
t Stat	59.533	29.609	28.80340889
Prob.(1-tail)	**0.000**	**0.000**	**0.000000**
t Critic. (1-tail)	1.833	1.833	1.833112933
Prob.(2-tail)	**0.000**	**0.000**	**0.000000**
t Critic. (2-tail)	2.262	2.262	2.262157163
"Throughput" with "malicious mote"			
Method	***"QCM", "DNC"***	***"QCM", "SLA"***	***"QCM", "Hy-IoT"***
P. Correlation	0.903	0.986	0.988
t Stat	6.867	6.871	6.044
Prob.(1-tail)	**0.001**	**0.001**	**0.001**
t Critic. (1-tail)	2.131	2.131	2.131
Prob.(2-tail)	**0.002**	**0.002**	**0.003**
t Critic. (2-tail)	2.776	2.776	2.776
"Throughput" with "malicious mote with realistic conditions"			
Method	***"QCM", "DNC"***	***"QCM", "SLA"***	***"QCM", "Hy-IoT"***
P. Correlation	0.960	0.938	0.989

t Stat	12.246	9.025	5.969
Prob.(1-tail)	**0.000**	**0.000**	**0.001**
t Critic. (1-tail)	2.131	2.131	2.131
Prob.(2-tail)	**0.000**	**0.000**	**0.003**
t Critic. (2-tail)	2.776	2.776	2.776
"Throughput" with "different simulation area"			
Method	***QCM and DNC***	***QCM and SLA***	***QCM and Hy-IoT***
P. Correlation	1	1	1
t Stat	-18.031	-13.889	-9.663
Prob.(1-tail)	**0.000**	**0.000**	**0.000**
t Critic. (1-tail)	2.131	2.131	2.131
Prob.(2-tail)	**0.000**	**0.000**	**0.000**
t Critic. (2-tail)	2.776	2.776	2.776
"Throughput" with "different pause time"			
Method	***"QCM", "DNC"***	***"QCM", "SLA"***	***"QCM", "Hy-IoT"***
P. Correlation	-1	1	1
t Stat	-31.6337	-106.066	-27.435
Prob.(1-tail)	**0.000**	**0.000**	**0.000**
t Critic. (1-tail)	2.131847	2.131	2.131
Prob.(2-tail)	**0.000**	**0.000**	**0.000**
t Critic. (2-tail)	2.776	2.776	2.776
"Throughput" with "different mobility speed"			
Method	***"QCM", "DNC"***	***"QCM", "SLA"***	***"QCM", "Hy-IoT"***
P. Correlation	1	1	1
t Stat	-28.890	-312.541	-8.687
Prob.(1-tail)	**0.000**	**0.000**	**0.000**
t Critic. (1-tail)	2.131	2.131	2.131
Prob.(2-tail)	**0.000**	**0.000**	**0.000**
t Critic. (2-tail)	2.776	2.776	2.776

Table 6.11 demonstrates the ANOVA test statistics of the QCM algorithm compared with other algorithms. It can be noticed that “F statistics” values are sufficiently larger than “F critical values”. In addition, the “P values” are less than 0.05, which achieves our 95% confidence interval, showing that the QCM algorithm outperforms the existing algorithms evaluated through inferential analysis.

Table 6.11: ANOVA Statistics in Terms of “Throughput” Scenarios

"Throughput" with "different intervals of traffic"						
Variation	*SS*	*Df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	1454.170	3	484.723	7.408	0.000	2.866

Within Groups	2355.439	36	65.428			
Total	3809.609	39				
"Throughput" with "malicious mote"						
Variation	*SS*	*Df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	673.109	3	224.369	4.334246	0.029	3.238
Within Groups	2690.596	16	168.162			
Total	3363.705	19				
"Throughput" with "malicious mote with realistic conditions"						
Variation	*SS*	*Df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	2399.974	3	799.991	3.349	0.045	3.238
Within Groups	3821.948	16	238.871			
Total	6221.922	19				
"Throughput" with "different simulation area"						
Variation	*SS*	*Df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	5891.138	3	1963.713	88.956	0.000	3.238
Within Groups	353.2	16	22.075			
Total	6244.338	19				
"Throughput" with "different pause time"						
Variation	*SS*	*Df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	5057.838	3	1685.946	1266.438	0.000	3.238
Within Groups	21.3	16	1.33125			
Total	5079.138	19				
"Throughput" with "different mobility speed"						
Variation	*SS*	*Df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	4890.638	3	1630.213	97.43519	0.000	3.238
Within Groups	267.7	16	16.731			
Total	5158.338	19				

6.5 Discussion (Hypothesis Testing)

The study centered the hypothesis that the **QCM (Query Control Mechanism)** algorithm (Khan, F. A., Noor, R. M., Mat Kiah, M. L., Noor, N. M., Altowaijri, S. M., & Rahman, A. U., 2019) outperforms the other existing algorithms, i.e., DnC, SLA, and Hy-IoT for QoS-enabled layered-based clustering for reactive flooding on the IoTs. The study expounded several defensive techniques counter to redundant/unwanted routing queries, that cause massive network traffic, and thus results in flooding under IoT networks. In this research, the authors instigated the Interlayer clustering (IELC) of CBF by proposing a query control mechanism (QCM) for detection and termination of redundant queries. The proposal relied on strength of link signal, query packet consistency, and limit threshold.

The scientific findings of this study clearly demonstrated that the proposed QCM algorithm performed significant against other state of the art defensing algorithms in context of average energy consumption, delay in traffic, and throughput. The study observed that the QCM significantly reduced the average energy consumption under different traffic intervals. The performance of QCM was also realized better in context of average energy consumption with malicious mods as compared to conventional algorithms at different motes. Besides, the QCM also demonstrated very profound performance in context of network delay as compared to the state-of-the-art algorithms by decreasing the delay considerably.

In addition, under malicious motes scenario, the QCM algorithm decreased the delay in network traffic to a significant extent. Finally, the QCM greatly enhanced the throughput as compared to Hy-IoT.

The comparative analysis of results revealed the difference between QCM algorithm and the existing algorithms. This study plans to enhance the work in future by considering a discreet component circuit implementation model employing Bouali's system for detection of other attacks. This will help in enhancing the quantity and types of motes to validate the reliability of QCM under the scenario of different motes. In addition, the study plans to consider the proactive part Interlayer clustering of CBF, that is appealing in high priority IoT networks, requiring smaller delays, i.e., in context of smart transportation, health, security and other types of physical prototype models.

The statistical evaluation of the proposed and the existing methods reveal that the significant "P value", based on the data points of different algorithms, demonstrated that the proposed algorithm outperformed the existing algorithms by achieving the "P value" lower than 0.05, and ensuring the achievement of 95% confidence interval. The inferential analysis demonstrated that the study could reject the null hypothesis and can accept the alternative hypothesis since the QCM algorithm performed significant as compared to other existing algorithms.

This research utilized statistical measures in performance evaluation of different QoS-enabled layered-based clustering algorithms in context of reactive flooding in IoT. The study performed the inferential analysis in terms of **Energy Consumption** under different cases including malicious motes, time interval, malicious motes (realistic condition), varied mobility speed, varied simulation area and varied pause time. Similarly, Inferential analysis was also taken in terms of **Delay** under different cases including malicious motes, time interval, malicious motes (realistic condition), varied mobility speed, varied simulation area and varied pause time. Further, the research estimated the inferential measures in the context of **Throughput** having different

cases including malicious motes, time interval, malicious motes (realistic condition), varied mobility speed, varied simulation area and varied pause time

6.6 Conclusions and Future Work

This Chapter has presented different defensive techniques in context of redundant routing queries, that lead to heavy network traffic, causing flooding in IoT networks. The study implemented the reactive part Interlayer clustering of CBF and presented a query control mechanism for detection and termination of redundant curries, based on strength of link signal, query consistency and query limit threshold. The performance evaluation of different algorithms revealed that the QCM algorithm outperformed the existing state of the art defensive algorithms in terms of average energy consumption, traffic delay, and quality of service.

In context of average energy consumption, the proposed method can detect flooder motes and can detach them from the network, thus reducing the energy consumption levels (arising under redundant (unwanted) flooding attacks. While it is noticed that the average energy consumption of prevailing DnC, and SLA as about 21 and 18%, respectively, under 1–5 s intervals, resulting in rise of this ratio continuously as the interval increases. Nevertheless, the QCM method consumes energy under 6% compared to the 13% of the prevailing Hy-IoT method. In terms of traffic delays, the study investigated the QCM as a good detector, pause manager, and detacher for flooding motes from the network. These measures greatly improved the performance. Further, the redundant (unwanted) queries could also remove by disengaging the flooding motes. DnC and SLA exhibited a traffic delay of 26% and 20%, respectively for interval 1. The proposed algorithm reduced the traffic delay to about 10% in the network at the same interval. QCM could achieve a drop in traffic delay about 4% lower than the conventional Hy-IoT. In terms of network throughput (QoS), the throughput of DnC and

SLA decreased up to approximately (44% and 61%), respectively at an interval of 1 s. While, with the QCM the QoS boosts to approximately 85%, this result is 9% better as compared to existing Hy-IoT at the same time interval 1 s.

CHAPTER 7: CONCLUSION

This Chapter concludes the thesis by presenting and reappraising the research questions and objectives presented in Section 1.4 and Section 1.5. The problems identified in this study, the research objectives, and accomplishments have been mapped to highlight the research. Furthermore, this chapter presents limitations of the current study and research directions worthy of pursuing future research directions.

7.1 Reappraisal of the Research Objectives and Research Questions

In this section of the thesis, various objectives are mapped against the research to discuss the finding of the study.

Objective 1: To investigate the state-of-the-art solution and identify the issues and limitations to prioritize, detect and terminate the redundant and unwanted flooding queries over the sensor and network layer of IoT network.

To achieve this objective, the study conducted an extensive review of literature employing numerous academic databases namely, the web of science, Scopus, ScienceDirect, IEEE Xplore, Medline, PubMed, SpringerLink, and ACM. The area of interests conducted using these academic databases include query flooding, Quality-of-Service (QoS) aspects of clustering, query control mechanism in IoT networks, and prioritization of vital queries. Moreover, the study investigated the limitations of existing solutions and identified the research gap to propose a robust and enhanced solution.

Objective 2: To design cross-layered Cluster Based Flooding (CBF) technique for priority and redundant queries. Two new algorithms are introduced as below:

- Interlayer Clustering (IELC) algorithm for network layer that uses advance query control mechanism (QCM) for detecting and terminating the redundant and unwanted queries and network flooding.
- Intralayer Clustering (IALC) algorithm for physical layer that maintain priority queries information locally.

The study achieved this research objective by developing a QoS enabled cross-layered clustering technique for mitigating flooding queries in IoT networks. The proposed method that based on an interoperable solution worked well for both physical and network layer devices. Since cross-layered CBF segments the entire network into different clusters, the IALC maintained the local query information proactively, while IELC is responsible for reactive achievement of routing queries to destinations (outside the cluster). The study presented CBF as a hybrid approach, having the capability to be more effective against conventional schemes in context of query traffic generation. Interlayer clustering (IELC) was found effective since it contained advanced query detection and termination techniques (QCM) that associated the signal strength and QueryLimitThreshold (QLT) values to detect flooding. It was found significant to minimize the energy consumption, network flooding, and identification and elimination of redundant routing queries in IoT networks.

A query control mechanism is essentially required for prioritizing, detection and termination of the redundant/unwanted flooding queries. The QCM technique employed a change in QueryLimitThreshold (QLT) for detection and termination of redundant query request packets. The mechanism was found elegant in enhancing the functioning of the IoT's network in context of strength of signal of query packets, and enhancing the location consistency verifying connected motes, thus shielding the network for reactive flooding attacks. The core idea of CBF was to segment the whole network into different

routing clusters. Proactive maintenance was done by the Intralayer clustering (IALC) by exploiting route query exchange and update query packets. Interlayer clustering (IELC) contributed in reactively transfer route query packets to motes, residing outside of the mote's cluster through query-reply packets.

Objective 3: To formulate the cross-layered Cluster Based Flooding (CBF) using the Sets and (Pro B).

The study achieved this objective by designing a refinement-based process in design of a formal specification of cluster-based flooding. The border casting service was adopted to examine the route discovery process. Since every mote in the dynamic network environment sent information to its neighbors in the cluster radius, the proposed model was not only limited to formally described stated issues but also took account of the system's stabilization property. To achieve the objective, the study also defined some invariants to validate the route discovery properties. It is to be noted that the CBF's target was to use a border casting service rather than broadcasting or flooding to find the appropriate routes. The statistics for discharged proof obligations are summarized in Table 7.1 as mentioned below.

Table 7.1: A summary of proof obligation

Model	Total number of POs	Automatically discharged	Manually discharged
Initial Model	15	13	2
Refinement 1	16	14	2
Refinement 2	60	37	23
Refinement 3	29	17	12
Refinement 4	97	54	43
Refinement 5	79	38	41
Refinement 6	21	14	7
Refinement 7	19	13	6
Refinement 8	65	40	25
Total	401	240 (60%)	161 (40%)

It had formed approximately more than 300 proof obligations, of which half were automatically proven. Discharging the generated proof obligation ensured that the refinements were correct, and the properties (invariants) were preserved.

Since the existing studies lack the formal validation of cross-layered routing protocols, this study emphasized on formal specification of cross-layered cluster-based flooding CBF at event B and proved the correctness of the route discovery mechanism. As a refinement-based method, an improved way was required to add system details to the corresponding model gradually. It made modeling and authentication easier for the user by allowing later versions to keep all the proven attributes in the previous model. Each node in the CBF cluster broadcasted link-state queries regularly. To model periodic broadcast/flooding activity, the study applied constraints in the formalization. As formalization of CBF is more complicated than formalizing a sole proactive or reactive approach protocol, this study took it as an important issue needed to be formally specified in a significant way. By improving the method, it permitted to develop a system from abstract to concrete. The correctness for refinements was assured through discharging some proof obligations. The study adopted the ProB, an animation tool to validate the model and ensured the formalization of the device specifications.

Objective 4: To design real time QoS enabled Query Control Mechanism (QCM) testbed used to detect and terminate the redundant and unwanted queries in IoT networks. The proposed testbed aims to reduce the number of duplicate/overlapping queries in IoT networks to improve QoS.

This research objective was achieved by design of a QoS enabled QCM testbed to detect and mitigate the redundant and unwanted queries in IoT networks. The proposed

testbed significantly reduced the number of duplicate/overlapping queries in IoT networks to improve QoS. The designed solution investigated the overlapping clusters in the whole query space promptly. Further, the proposed system employed heterogeneous IoT Sensors, i.e., IR, Ultrasonic, RFID, temperature, sound, LDR, and water sensing for generating queries. It acted as the input unit for the proposed system that monitors and sensed each query in their vicinity.

The severe drawbacks of flooding appear as the overlapping of query threads. This phenomenon massively propagates redundant and unwanted queries, harvest excessive resource utilization, and thus reduce. The real-time QoS enabled testbed significantly analyzed the smart query detection and mitigation by effectively managing the unwanted querying. The solution also did not generate any additional control traffic and required a negligible computational overhead. The study further observed that an intercluster control traffic can be achieved by preventing thread overlap locally through selective flooding. Since the query control mechanism, being aware of the query information, the testbed could significantly manage all the overlapping clusters in the entire query space.

Objective 5: To evaluate our proposed approach using simulation tools under realistic scenarios and compare the results with the state-of-the-art approaches in the literature as well as validate the results using a statistical analysis tool.

This research objective was achieved by evaluating the performance of proposed model employing Contiki Cooja network simulator as a state-of-the-art simulation tool for redundant flooding scenario. To achieve the best outcomes of this evaluation, this study considered the customized parameterization of simulator on realistic scenarios by considering energy consumption, delay, and throughput with different intervals of traffic, malicious mote, malicious mote with realistic condition, different simulation

areas, pause time, and mobility speed. The QCM technique outperformed compared to DnC, SLA, and Hy-IoT approaches in terms of dropping the average consumption of energy, traffic delay, and by boosting the considerable throughput.

Further, to evaluate the performance of QCM, this study employed ANOVA and t-tests in context of special scenarios i.e., in terms of energy consumption, delay, and throughput. These three scenarios were evaluated in terms of intervals of traffic, malicious mote, malicious mote with realistic conditions, different simulation area, pause time, and mobility speed. The inferential analysis statistics depicted that "F and t" values are sufficiently larger than corresponding critical values. In addition, the "P values" were less than 0.05, which achieved the 95% confidence interval, showing that the QCM algorithm outperformed the existing algorithms.

A deep and exhaustive review of existing solutions, this study investigated that an undesirable side effect of flooding is the overlapping of query threads. Which may lead to the propagation of redundant and unwanted queries, resulting in excessive resource utilization and may reducing QoS in term of energy consumption, network delay and throughput. The QCM mechanism analyzed the smart query detection and mitigation techniques to effectively combat the redundant querying, while generating no additional control traffic and requiring negligible computational overhead. Further reduction of the intercluster control traffic could be achieved by preventing thread overlap locally through selective flooding. When the CBF was configured to minimize total routing control traffic, the study found that full flooding responded to route queries at least three times faster than a selective flooding implementation.

In context of average energy consumption, the proposed method can detect flooder motes and can detach them from the network, thus reducing the energy consumption levels (arising under redundant (unwanted) flooding attacks. We noticed the average

energy consumption of prevailing DnC, and SLA as about 21 and 18%, respectively, under 1–5 s intervals, resulting in rise of this ratio continuously as the interval increases. Nevertheless, the QCM method consumes energy under 6% compared to the 13% of the prevailing Hy-IoT method. In terms of traffic delays, the study investigated the QCM as a good detector, pause manager, and detacher for flooding motes from the network. These measures greatly improved the performance. Further, the redundant (unwanted) queries could also remove by disengaging the flooding motes. DnC and SLA exhibited a traffic delay of 26% and 20%, respectively for interval 1. The proposed algorithm reduced the traffic delay to about 10% in the network at the same interval. QCM could achieve a drop in traffic delay about 4% lower than the conventional Hy-IoT. In terms of network throughput (QoS), the throughput of DnC and SLA decreased up to approximately (44% and 61%), respectively at an interval of 1 s. While with the QCM the QoS boosts to approximately 85%, this result is 9% better as compared to existing Hy-IoT at the same time interval 1 s.

7.2 Research Scope and Limitation

The scope of this study is contained under IoT sensors/devices, delay, QoS, Energy Consumption, throughput, redundant query, flooding, and cloud.

This study is limited to two layers only and are covering all the layers of IoT. Further, the study requires to build a cross-layered prototype, the IALC Intralayer only focus on priority queries need to improve it further. Moreover, the flooding created by the priority queries needs to be rectified.

7.3 Future Work

In this Section, enlist a possible future work that can be elevated from this research study.

1. The testbed needs to handle additional functionality and more possibilities for research and needs to adapt continually. If the extension of testbeds is an obvious path forward for more diverse IoT-related innovations and protocols, additional attractive aspects are often considered.

2. To build all kinds of test cases, it is suggested to keep the project as open-source project for scientific community and IoT developers. The testbed can be made publicly available through web with a well-managed monitoring of motes using their correct logging

3. In future, the employment of customized Machine Learning, and Artificial Intelligence approaches can handle sophisticated IoT network QoS analyses and can provide appropriate redundant query recognition systems.

7.4 Summary

This Chapter demonstrated the accomplishments of research objectives of this study, along with mapping the research objectives with the research questions. In addition, it also provided a proximity association between the problem statement, objectives, and the proposed solutions. Further, the scope and limitations of this were discussed. The Chapter glimpsed the future directions of work in context of testbed enhancement to mitigate the query flooding using machine learning techniques.

REFERENCES

Aazam, M., St-Hilaire, M., Lung, C. H., & Lambadaris, I. (2016, June). MeFoRE: QoE based resource estimation at Fog to enhance QoS in IoT. *2016 23rd International Conference on Telecommunications, ICT 2016*. https://doi.org/10.1109/ICT.2016.7500362

Abdalzaher, M. S., Seddik, K., Elsabrouty, M., Muta, O., Furukawa, H., & Abdel-Rahman, A. %J S. (2016). *Game theory meets wireless sensor networks security requirements and threats mitigation: A survey*. *16*(7), 1003.

Abdelaal, M., Theel, O., Kuka, C., Zhang, P., Gao, Y., Bashlovkina, V., Nicklas, D., & Fränzle, M. %J I. J. of D. S. N. (2016). *Improving energy efficiency in QoS-constrained wireless sensor networks*. *12*(5), 1576038.

Abdollahzadeh, S., & Navimipour, N. J. (2016). Deployment strategies in the wireless sensor network: A comprehensive review. In *Computer Communications* (Vols. 91–92, pp. 1–16). Elsevier B.V. https://doi.org/10.1016/j.comcom.2016.06.003

Abdulridha, M., Adday, G., Jiaotong, I. A.-J. of S., & 2019, undefined. (2019). Fast simple flooding strategy in wireless sensor networks. *Jsju.Org*, *54*(6). https://doi.org/10.35741/issn.0258-2724.54.6.12

Abrial, J. R., Butler, M., Hallerstede, S., Hoang, T. S., Mehta, F., & Voisin, L. (2010). Rodin: An open toolset for modelling and reasoning in Event-B. *International Journal on Software Tools for Technology Transfer*, *12*(6), 447–466. https://doi.org/10.1007/s10009-010-0145-y

Accettura, N., Palattella, M. R., Boggia, G., Grieco, L. A., & Dohler, M. (2013). Decentralized Traffic Aware Scheduling for multi-hop Low power Lossy Networks in the Internet of Things. *2013 IEEE 14th International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2013*. https://doi.org/10.1109/WoWMoM.2013.6583485

Adjih, C., Baccelli, E., Fleury, E., Harter, G., Mitton, N., Noel, T., Pissard-Gibollet, R., Saint-Marcel, F., Schreiner, G., Vandaele, J., & Watteyne, T. (2015). FIT IoT-LAB: A large scale open experimental IoT testbed. *IEEE World Forum on Internet of Things, WF-IoT 2015 - Proceedings*. https://doi.org/10.1109/WF-IoT.2015.7389098

Ahmad, M., Jung, L. T., Bhuiyan, A.-A. %J B. S. P., & Control. (2017). *From DNA to protein: Why genetic code context of nucleotides for DNA signal processing? A review*. *34*, 44–63.

Ahmad, M., Jung, L. T., Bhuiyan, M. A.-A. %J C. in biology, & medicine. (2016). *On fuzzy semantic similarity measure for DNA coding*. *69*, 144–151.

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., & Cayirci, E. (2002). Wireless sensor networks: A survey. *Computer Networks*. https://doi.org/10.1016/S1389-1286(01)00302-4

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys and Tutorials*, *17*(4), 2347–2376. https://doi.org/10.1109/COMST.2015.2444095

Alam, S., Chowdhury, M. M. R., & Noll, J. (2010). SenaaS: An event-driven sensor

virtualization approach for internet of things cloud. *2010 IEEE International Conference on Networked Embedded Systems for Enterprise Applications, NESEA 2010*. https://doi.org/10.1109/NESEA.2010.5678060

Alamri, A., & Abdullah, M. (2016). *Cross-layer quality of service protocols for wireless multimedia sensor networks 02\70*2 View project Wireless Multimedia Sensor Networks View project Cross-layer quality of service protocols for wireless multimedia sensor networks*.

Alghamdi, A., Alshamrani, M., Alqahtani, A., Al Ghamdi, S. S. A., & Harrathi, R. (2016, November). Secure data aggregation scheme in wireless sensor networks for IoT. *2016 International Symposium on Networks, Computers and Communications, ISNCC 2016*. https://doi.org/10.1109/ISNCC.2016.7746071

Alkhamisi, A., Nazmudeen, M. S. H., & Buhari, S. M. (2016, September). A cross-layer framework for sensor data aggregation for IoT applications in smart cities. *IEEE 2nd International Smart Cities Conference: Improving the Citizens Quality of Life, ISC2 2016 - Proceedings*. https://doi.org/10.1109/ISC2.2016.7580853

Alqahtani, A., Solaiman, E., Buyya, R., & Ranjan, R. (2016). End-to-End QoS Specification and Monitoring in the Internet of Things. *Https://Eprints.Ncl.Ac.Uk*.

Ameigeiras, P., Ramos-Munoz, J. J., Navarro-Ortiz, J., Mogensen, P., & Lopez-Soler, J. M. (2010). QoE oriented cross-layer design of a resource allocation algorithm in beyond 3G systems. *Computer Communications*, *33*(5), 571–582. https://doi.org/10.1016/j.comcom.2009.10.016

Appavoo, P., William, E. K., Chan, M. C., & Mohammad, M. (2019). Indriya2: A

Heterogeneous Wireless Sensor Network (WSN) Testbed. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*. https://doi.org/10.1007/978-3-030-12971-2_1

Arduino - ArduinoBoardMega2560. (n.d.).

Arkian, H. R., Atani, R. E., Pourkhalili, A., & Kamali, S. %J J. I. S. E. (2015). *A Stable Clustering Scheme Based on Adaptive Multiple Metric in Vehicular Ad-hoc Networks*. *31*(2), 361–386.

Atzori, L., Iera, A., Morabito, G., & Nitti, M. (2012). The social internet of things (SIoT) - When social networks meet the internet of things: Concept, architecture and network characterization. *Computer Networks*, *56*(16), 3594–3608. https://doi.org/10.1016/j.comnet.2012.07.010

Awan, I., Younas, M., & Naveed, W. (2014). Modelling qos in iot applications. *2014 17th International Conference on Network-Based Information Systems*, 99–105.

Baccelli, E., Hahm, O., Gunes, M., Wahlisch, M., & Schmidt, T. (2014). *RIOT OS: Towards an OS for the Internet of Things*. 79–80. https://doi.org/10.1109/infcomw.2013.6970748

Badamasi, Y. A. (2014). The working principle of an Arduino. *Proceedings of the 11th International Conference on Electronics, Computer and Computation, ICECCO 2014*. https://doi.org/10.1109/ICECCO.2014.6997578

Baddeley, M., Raza, U., Stanoev, A., Oikonomou, G., Nejabati, R., Sooriyabandara, M., & Simeonidou, D. (2019). Atomic-SDN: Is Synchronous Flooding the Solution to

Software-Defined Networking in IoT? *IEEE Access*, *7*, 96019–96034. https://doi.org/10.1109/ACCESS.2019.2920100

Baker, T., Al-Dawsari, B., Tawfik, H., Reid, D., & Ngoko, Y. (2015). GreeDi: An energy efficient routing algorithm for big data on cloud. *Ad Hoc Networks*, *35*, 83–96. https://doi.org/10.1016/j.adhoc.2015.06.008

Bandyopadhyay, D., & Sen, J. (2011). Internet of things: Applications and challenges in technology and standardization. *Wireless Personal Communications*, *58*(1), 49–69. https://doi.org/10.1007/s11277-011-0288-5

Barrett, S. F. (2013). Arduino Microcontroller Processing for Everyone! Third Edition. *Synthesis Lectures on Digital Circuits and Systems*, *43*, 1–515. https://doi.org/10.2200/S00522ED1V01Y201307DCS043

Benenson, Z., Freiling, F. C., Hammerschmidt, E., Lucks, S., & Pimenidis, L. (2006). Authenticated query flooding in sensor networks. *IFIP International Information Security Conference*, 38–49.

Bernard, M. S., Pei, T., Li, Z., & Li, K. (2019). QoS strategies for wireless multimedia sensor networks in the context of IoT. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, *275*, 228–253. https://doi.org/10.1007/978-3-030-16042-5_21

Bhandary, V., Malik, A., & Kumar, S. (2016). Routing in wireless multimedia sensor networks: A survey of existing protocols and open research issues. In *Journal of Engineering (United Kingdom)* (Vol. 2016). Hindawi Limited. https://doi.org/10.1155/2016/9608757

Bourke, T., van Glabbeek, R., Höfner, P., & van Glabbeek, R. (2014). A mechanized proof of loop freedom of the (untimed) AODV routing protocol. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8837*, 47–63. https://doi.org/10.1007/978-3-319-11936-6_5

Bu, S., Richard Yu, F., Cai, Y., & Liu, X. P. (2012). When the smart grid meets energy-efficient communications: Green wireless cellular networks powered by the smart grid. *IEEE Transactions on Wireless Communications*, *11*(8), 3014–3024. https://doi.org/10.1109/TWC.2012.052512.111766

Burin des Rosiers, C., Chelius, G., Fleury, E., Fraboulet, A., Gallais, A., Mitton, N., & Noël, T. (2012). *SensLAB*. https://doi.org/10.1007/978-3-642-29273-6_19

Cansell, D., & Méry, D. (2006). Formal and incremental construction of distributed algorithms: On the distributed reference counting algorithm. *Theoretical Computer Science*, *364*(3), 318–337. https://doi.org/10.1016/j.tcs.2006.08.015

Cerpa, A., Elson, J., Estrin, D., Girod, L., Hamilton, M., & Zhao, J. (2001). Habitat monitoring: Application driver for wireless communications technology. *Computer Communication Review*. https://doi.org/10.1145/844193.844196

Cerpa, A., Elson, J., Hamilton, M., Zhao, J., Estrin, D., & Girod, L. (2001). Habitat monitoring: Application driver for wireless communications technology. *SIGCOMM LA 2001 - Workshop on Data Communication in Latin America and the Caribbean*. https://doi.org/10.1145/371626.371720

Chao, C. M., & Hsiao, T. Y. (2014). Design of structure-free and energy-balanced data

aggregation in wireless sensor networks. *Journal of Network and Computer Applications*, *37*(1), 229–239. https://doi.org/10.1016/j.jnca.2013.02.013

Cheng, L., Niu, J., Luo, C., Shu, L., Kong, L., Zhao, Z., & Gu, Y. (2018). Towards minimum-delay and energy-efficient flooding in low-duty-cycle wireless sensor networks. *Computer Networks*, *134*, 66–77. https://doi.org/10.1016/j.comnet.2018.01.012

Choi, C., Park, J. H., Na, M., & Jo, S. (2015). Low latency 5G architecture for mission-critical IoT (Internet of Things). *ITC-CSCC :International Technical Conference on Circuits Systems, Computers and Communications*, 58–59.

Clark, J., Bendisposto, J., Hallerstede, S., Hansen, D., & Leuschel, M. (2016). Generating event-B specifications from algorithm descriptions. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9675*, 183–197. https://doi.org/10.1007/978-3-319-33600-8_11

Conti, M., Di Pietro, R., & Spognardi, A. (2014). Clone wars: Distributed detection of clone attacks in mobile WSNs. *Journal of Computer and System Sciences*, *80*(3), 654–669. https://doi.org/10.1016/j.jcss.2013.06.017

Cui, Y. R. (2009). Data query protocol with restriction flooding in wireless sensor networks. *Proceedings - International Conference on Networks Security, Wireless Communications and Trusted Computing, NSWCTC 2009*. https://doi.org/10.1109/NSWCTC.2009.325

Dagar, M., Mahajan, S. (2013). Data aggregation in wireless sensor network: A survey.

Int. J. Inf. Comput. Technol. (IJICT), *3*(3), 167–174.

David, D. R., Nait-Sidi-moh, A., Durand, D., & Fortin, J. (2015). Using Internet of Things technologies for a collaborative supply chain: Application to tracking of pallets and containers. *Procedia Computer Science*, *56*(1), 550–557. https://doi.org/10.1016/j.procs.2015.07.251

Delgado-Rajo, F., Melian-Segura, A., Guerra, V., Perez-Jimenez, R., & Sanchez-Rodriguez, D. (2020). Hybrid RF/VLC Network Architecture for the Internet of Things. *Sensors 2020, Vol. 20, Page 478*, *20*(2), 478. https://doi.org/10.3390/S20020478

Dhand, G., & Tyagi, S. S. (2016). Data Aggregation Techniques in WSN:Survey. *Procedia Computer Science*, *92*, 378–384. https://doi.org/10.1016/j.procs.2016.07.393

Dhumane, A., Prasad, R., & Prasad, J. (2016). Routing issues in internet of things: a survey. *Proceedings of the International Multiconference of Engineers and Computer Scientists*, *1*, 16–18.

Dietzel, S., Gürtler, J., & Kargl, F. (2016). A resilient in-network aggregation mechanism for VANETs based on dissemination redundancy. *Ad Hoc Networks*, *37*, 101–109. https://doi.org/10.1016/j.adhoc.2015.09.002

Doddavenkatappa, M., Chan, M. C., & Ananda, A. L. (2012). Indriya: A low-cost, 3D wireless sensor network testbed. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*. https://doi.org/10.1007/978-3-642-29273-6_23

Duan, R., Chen, X., & Xing, T. (2011). A QoS architecture for IOT. *Proceedings - 2011 IEEE International Conferences on Internet of Things and Cyber, Physical and Social Computing, IThings/CPSCom 2011*, 717–720. https://doi.org/10.1109/iThings/CPSCom.2011.125

El Mimouni, S., & Bouhdadi, M. (2018). A mechanized formal refinement proof of modbus communication using Event-B proof system. *International Journal of Intelligent Engineering and Systems*, *11*(4), 97–106. https://doi.org/10.22266/ijies2018.0831.10

Elsayed, E., El-Sharawy, G., & El-Sharawy, E. (2013). INTEGRATION OF AUTOMATIC THEOREM PROVERS IN EVENT-B PATTERNS. *International Journal of Software Engineering & Applications (IJSEA)*, *4*(1). https://doi.org/10.5121/ijsea.2013.4103

Fasolo, E., Rossi, M., Widmer, J., & Zorzi, M. (2007). In-network aggregation techniques for wireless sensor networks: A survey. *IEEE Wireless Communications*. https://doi.org/10.1109/MWC.2007.358967

Fathallah, K., Abid, M. A., & Hadj-Alouane, N. Ben. (2019). Routing of spatial queries over IoT enabled wireless sensor networks. *2019 15th International Wireless Communications and Mobile Computing Conference, IWCMC 2019*. https://doi.org/10.1109/IWCMC.2019.8766512

Foh, C. H., Zhang, Y., Ni, Z., Cai, J., & Ngan, K. N. (2007). Optimized cross-layer design for scalable video transmission over the IEEE 802.11e networks. *IEEE Transactions on Circuits and Systems for Video Technology*, *17*(12), 1665–1678.

https://doi.org/10.1109/TCSVT.2007.903808

Fredj, S. Ben, Boussard, M., Kofman, D., & Noirie, L. (2013). A scalable IoT service search based on clustering and aggregation. *Proceedings - 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, GreenCom-IThings-CPSCom 2013*, 403–410. https://doi.org/10.1109/GreenCom-iThings-CPSCom.2013.86

Gammarano, N., Schandy, J., & Steinfeld, L. (2018). Q-SAND: A quick neighbor discovery protocol for wireless networks with sectored antennas. *2018 9th Argentine Symposium and Conference on Embedded Systems, CASE 2018*, 19–24. https://doi.org/10.23919/SASE-CASE.2018.8542163

Gluhak, A., Krco, S., Nati, M., Pfisterer, D., Mitton, N., & Razafindralambo, T. %J I. C. M. (2011). *A survey on facilities for experimental internet of things research*. *49*(11), 58–67.

González-Manzano, L., Fuentes, J. M. D., Pastrana, S., Peris-Lopez, P., & Hernández-Encinas, L. (2016). PAgIoT - Privacy-preserving Aggregation protocol for Internet of Things. *Journal of Network and Computer Applications*, *71*, 59–71. https://doi.org/10.1016/j.jnca.2016.06.001

Goudarzi, S., Kama, N., Anisi, M. H., Zeadally, S., & Mumtaz, S. (2019). Data collection using unmanned aerial vehicles for Internet of Things platforms. *Computers and Electrical Engineering*, *75*, 1–15. https://doi.org/10.1016/j.compeleceng.2019.01.028

Gračanin, D., Adams, K. P., & Eltoweissy, M. (2006). Data replication in collaborative

sensor network systems. *Conference Proceedings of the IEEE International Performance, Computing, and Communications Conference*. https://doi.org/10.1109/.2006.1629431

Guan, Q., Yu, F. R., & Jiang, S. (2010). Prediction-based topology control and routing in cognitive radio mobile ad hoc networks. *Proceedings - IEEE INFOCOM*. https://doi.org/10.1109/INFCOMW.2010.5466716

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, *29*(7), 1645–1660. https://doi.org/10.1016/j.future.2013.01.010

Haddad, H., Bouyahia, Z., & Jabeur, N. %J P. C. S. (2017). *Towards a Three-Level Framework for IoT Redundancy Control through an Explicit Spatio-Temporal Data Model*. *109*, 664–671.

Hajizadeh, R., & Jafari Navimipour, N. (2017). A method for trust evaluation in the cloud environments using a behavior graph and services grouping. *Kybernetes*, *46*(7). https://doi.org/10.1108/K-02-2017-0070

Hoang, T. S., Fürst, A., & Abrial, J. R. (2013). Event-B patterns and their tool support. *Software and Systems Modeling*, *12*(2), 229–244. https://doi.org/10.1007/s10270-010-0183-7

Huang, J., Duan, Q., Zhao, Y., Zheng, Z., & Wang, W. (2017). Multicast Routing for Multimedia Communications in the Internet of Things. *IEEE Internet of Things Journal*, *4*(1), 215–224. https://doi.org/10.1109/JIOT.2016.2642643

Huysmans, S., Rigole, P., Berbers, Y., & Huysmans Peter Rigole Yolande Berbers, S. (2008). *Query by Combination in the Internet of Things*. David Coudert.

Ikram, A., Anjum, A., Hill, R., Antonopoulos, N., Liu, L., & Sotiriadis, S. (2015). Approaching the Internet of things (IoT): a modelling, analysis and abstraction framework. *Concurrency and Computation: Practice and Experience*, *27*(8), 1966–1984. https://doi.org/10.1002/cpe.3131

IR Sensor : Circuit Diagram, Types Working with Applications. (n.d.).

Jacob, L., & Shamna, H. R. (2015). Efficient Cooperative MAC and Routing in Wireless Networks. *Transactions on Networks and Communications*, *3*(5), 79–79. https://doi.org/10.14738/tnc.35.1586

Jafari Navimipour, N., & Fouladi, P. (2017). Human resources ranking in a cloud-based knowledge sharing framework using the quality control criteria. *Kybernetes*, *46*(5). https://doi.org/10.1108/K-01-2017-0007

Jamalipour, A., & Azim, M. A. (2006). Two-layer optimized forwarding for cluster-based sensor networks. *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, *May*. https://doi.org/10.1109/PIMRC.2006.254170

Jastram, Michael, and P. M. B. (2014). *Rodin User's Handbook: Covers Rodin v. 2.8.*

Jiang, H., Shen, F., Chen, S., Li, K. C., & Jeong, Y. S. (2015). A secure and scalable storage system for aggregate data in IoT. *Future Generation Computer Systems*, *49*, 133–141. https://doi.org/10.1016/j.future.2014.11.009

Jin, J., Gubbi, J., Luo, T., & Palaniswami, M. (2012). Network architecture and QoS issues in the internet of things for a smart city. *2012 International Symposium on Communications and Information Technologies (ISCIT)*, 956–961.

Jin, X., Wah, B. W., Cheng, X., & Wang, Y. (2015). Significance and Challenges of Big Data Research. *Big Data Research*, *2*(2). https://doi.org/10.1016/j.bdr.2015.01.006

Jing, Q., Vasilakos, A. V., Wan, J., Lu, J., & Qiu, D. (2014). Security of the Internet of Things: perspectives and challenges. *Wireless Networks*, *20*(8), 2481–2501. https://doi.org/10.1007/s11276-014-0761-7

Johnson, A. P., Patranabis, S., Chakraborty, R. S., & Mukhopadhyay, D. (2017). Remote dynamic partial reconfiguration: A threat to Internet-of-Things and embedded security applications. *Microprocessors and Microsystems*, *52*, 131–144. https://doi.org/10.1016/j.micpro.2017.06.005

Keitt, T. H., T. Urban, D. & Milne, & B. (1997). Detecting critical scales in fragmented landscapes: Detecting critical scales in fragmented landscapes. - Conservation Ecology 1. *Ecology and Society*.

Khan, A. R., & Chishti, M. A. (2020). Data aggregation mechanisms in the internet of things: A study, qualitative and quantitative analysis. *International Journal of Computing and Digital Systems*, *9*(2), 289–297. https://doi.org/10.12785/IJCDS/090214

Khan, R., Khan, S. U., Zaheer, R., & Khan, S. (2012). Future internet: The internet of things architecture, possible applications and key challenges. *Proceedings - 10th International Conference on Frontiers of Information Technology, FIT 2012*, 257–

260. https://doi.org/10.1109/FIT.2012.53

Kharche, S., & Pawar, S. (2017). Node level energy consumption analysis in 6LoWPAN network using real and emulated Zolertia Z1 motes. *2016 IEEE International Conference on Advanced Networks and Telecommunications Systems, ANTS 2016*. https://doi.org/10.1109/ANTS.2016.7947870

Kiran Maraiya, Kamal Kant, N. G. (n.d.). *Wireless_Sensor_Network_A_Review_on_Data_Aggregation*.

Koike, A., Ohba, T., & Ishibashi, R. (2016). IoT network architecture using packet aggregation and disaggregation. *Proceedings - 2016 5th IIAI International Congress on Advanced Applied Informatics, IIAI-AAI 2016*, 1140–1145. https://doi.org/10.1109/IIAI-AAI.2016.221

Krishnapriya, S and Joby, P. (2015). QoS aware resource scheduling in internet of things-cloud environment. *International Journal of Scientific \& Engineering Research*, *6*, 4.

Kumar Kumar, S., Sheikh, A., Ambhaikar, A., & Kumar, S. (2019). Quality of Services Improvement for Secure Iot Networks. *International Journal of Engineering and Advanced Technology (IJEAT)*, *9*, 2249–8958. https://doi.org/10.35940/ijeat.B3757.129219

Kumar, S., & Chaurasiya, V. K. (2019). A Strategy for Elimination of Data Redundancy in Internet of Things (IoT) Based Wireless Sensor Network (WSN). *IEEE Systems Journal*, *13*(2), 1650–1657. https://doi.org/10.1109/JSYST.2018.2873591

Kyung, Y., & Kim, T.-K. (2020). QoS-Aware Flexible Handover Management in Software-Defined Mobile Networks. *Applied Sciences 2020, Vol. 10, Page 4264*, *10*(12), 4264. https://doi.org/10.3390/APP10124264

Lahane, S. R., & Jariwala, K. N. (2021). Secured cross-layer cross-domain routing in dense wireless sensor network: A new hybrid based clustering approach. *International Journal of Intelligent Systems*, *36*(8), 3789–3812. https://doi.org/10.1002/INT.22438

Landt, J. (2005). The history of RFID. *IEEE Potentials*. https://doi.org/10.1109/MP.2005.1549751

Latif, K., Javaid, N., Ullah, I., Kaleem, Z., Malik, Z. A., & Nguyen, L. D. (2020). DIEER: Delay-Intolerant Energy-Efficient Routing with Sink Mobility in Underwater Wireless Sensor Networks. *Sensors 2020, Vol. 20, Page 3467*, *20*(12), 3467. https://doi.org/10.3390/S20123467

Laxmi, P and Deepthi, G. L. (2017). Smart Water Management Process Architecture with IoT Based Reference. *Int. J. Comput. Sci. Mob. Comput*, *6*, 271–276.

Leuschel, M., & Butler, M. (2008). ProB: An automated analysis toolset for the B method. *International Journal on Software Tools for Technology Transfer*, *10*(2), 185–203. https://doi.org/10.1007/s10009-007-0063-9

Levi, A., & Sarimurat, S. (2017). Utilizing hash graphs for key distribution for mobile and replaceable interconnected sensors in the IoT context. *Ad Hoc Networks*, *57*, 3–18. https://doi.org/10.1016/j.adhoc.2016.08.013

Li, L., Li, S., & Zhao, S. (2014). QoS-Aware scheduling of services-oriented internet of things. *IEEE Transactions on Industrial Informatics*, *10*(2), 1497–1507. https://doi.org/10.1109/TII.2014.2306782

Li, M., Jing, Y., & Li, C. (2013). A Robust and Efficient Cross-Layer Optimal Design in Wireless Sensor Networks. *Wireless Pers Commun*, *72*, 1889–1902. https://doi.org/10.1007/s11277-013-1111-2

Li, S.-F., Handziski, V., Köpke, A., Kubisch, M., & Wolisz, A. (2005). A Wireless Sensor Network Testbed Supporting Controlled In-building Experiments. *12. Sensor Kongress*.

Li, S., Xu, L. Da, & Zhao, S. (2015). The internet of things: a survey. *Information Systems Frontiers*, *17*(2), 243–259. https://doi.org/10.1007/s10796-014-9492-7

Li, W., Bao, J., & Shen, W. (2011). Collaborative wireless sensor networks: A survey. *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*. https://doi.org/10.1109/ICSMC.2011.6084070

Li, X., Liu, W., Xie, M., Liu, A., Zhao, M., Xiong, N. N., Zhao, M., & Dai, W. (2018). Differentiated Data Aggregation Routing Scheme for Energy Conserving and Delay Sensitive Wireless Sensor Networks. *Sensors 2018, Vol. 18, Page 2349*, *18*(7), 2349. https://doi.org/10.3390/S18072349

Li, Z., Yu, F. R., & Huang, M. (2010). A distributed consensus-based cooperative spectrum-sensing scheme in cognitive radios. *IEEE Transactions on Vehicular Technology*, *59*(1), 383–393. https://doi.org/10.1109/TVT.2009.2031181

Li, Z., Zhang, W., Qiao, D., & Peng, Y. (2017). Lifetime balanced data aggregation for the internet of things. *Computers and Electrical Engineering*, *58*, 244–264. https://doi.org/10.1016/j.compeleceng.2016.09.025

Liang, J. M., Chen, J. J., Cheng, H. H., & Tseng, Y. C. (2013). An energy-efficient sleep scheduling with QoS consideration in 3GPP LTE-advanced networks for internet of things. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, *3*(1), 13–22. https://doi.org/10.1109/JETCAS.2013.2243631

Liang, O., Ahmet Şekercioğlu, Y., & Mani, N. (2007). A low-cost flooding algorithm for wireless sensor networks. *IEEE Wireless Communications and Networking Conference, WCNC*, 3495–3500. https://doi.org/10.1109/WCNC.2007.641

Liao, Y. P., & Hsiao, C. M. (2014). A secure ECC-based RFID authentication scheme integrated with ID-verifier transfer protocol. *Ad Hoc Networks*, *18*, 133–146. https://doi.org/10.1016/j.adhoc.2013.02.004

Lim, R., Ferrari, F., Zimmerling, M., Walser, C., Sommer, P., & Beutel, J. (2013). FlockLab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems. *IPSN 2013 - Proceedings of the 12th International Conference on Information Processing in Sensor Networks, Part of CPSWeek 2013*. https://doi.org/10.1145/2461381.2461402

Liu, Y., Gong, X., & Xing, C. (2014). A novel trust-based secure data aggregation for Internet of Things. *Proceedings of the 9th International Conference on Computer Science and Education, ICCCSE 2014*, 435–439. https://doi.org/10.1109/ICCSE.2014.6926499

Liu, Y., Liu, A., Hu, Y., Li, Z., Choi, Y. J., Sekiya, H., & Li, J. (2016). FFSC: An Energy Efficiency Communications Approach for Delay Minimizing in Internet of Things. *IEEE Access*, *4*, 3775–3793. https://doi.org/10.1109/ACCESS.2016.2588278

Lu, X., Spear, M., Levitt, K., Matloff, N. S., & Wu, S. F. (2008). Using soft-line recursive response to improve query aggregation in wireless sensor networks. *IEEE International Conference on Communications*. https://doi.org/10.1109/ICC.2008.440

Luo, C., Yu, F. R., Ji, H., & Leung, V. C. M. (2010). Cross-layer design for TCP performance improvement in cognitive radio networks. *IEEE Transactions on Vehicular Technology*, *59*(5), 2485–2495. https://doi.org/10.1109/TVT.2010.2041802

Ma, L., Yu, F., Leung, V. C. M., & Randhawa, T. (2004). A new method to support UMTS/WLAN vertical handover using SCTP. In *IEEE Wireless Communications* (Vol. 11, Issue 4, pp. 44–51). https://doi.org/10.1109/MWC.2004.1325890

Magagula, L. A., & Chan, H. A. (2008). IEEE802.21-assisted cross-layer design and PMIPv6 mobility management framework for next generation wireless networks. *Proceedings - 4th IEEE International Conference on Wireless and Mobile Computing, Networking and Communication, WiMob 2008*, 159–164. https://doi.org/10.1109/WiMob.2008.46

Mainetti, L., Patrono, L., & Vilei, A. (2011). Evolution of wireless sensor networks towards the Internet of Things: A survey. *2011 International Conference on Software, Telecommunications and Computer Networks, SoftCOM 2011*.

Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., & Anderson, J. (2002). *Wireless sensor networks for habitat monitoring*. https://doi.org/10.1145/570738.570751

Mao, Y., Li, J., Chen, M. R., Liu, J., Xie, C., & Zhan, Y. (2016). Fully secure fuzzy identity-based encryption for secure IoT communications. *Computer Standards and Interfaces*, *44*, 117–121. https://doi.org/10.1016/j.csi.2015.06.007

Masoud, M. Z., Jaradat, Y., Zaidan, D., & Jannoud, I. (2019). To Cluster or Not to Cluster: A Hybrid Clustering Protocol for WSN. *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology, JEEIT 2019 - Proceedings*, 678–682. https://doi.org/10.1109/JEEIT.2019.8717524

Meguerdichian, S., Koushanfar, F., Potkonjak, M., & Srivastava, M. B. (2001). Coverage problems in wireless ad-hoc sensor networks. *Proceedings - IEEE INFOCOM*.

Melodia, T., & Akyildiz, I. F. (2010). Cross-layer QoS-aware communication for ultra wide band Wireless Multimedia Sensor Networks. *IEEE Journal on Selected Areas in Communications*, *28*(5), 653–663. https://doi.org/10.1109/JSAC.2010.100604

Merad Boudia, O. R., Senouci, S. M., & Feham, M. (2015). A novel secure aggregation scheme for wireless sensor networks using stateful public key cryptography. *Ad Hoc Networks*, *32*, 98–113. https://doi.org/10.1016/j.adhoc.2015.01.002

Mishra, S. (2012). Features of WSN and Data Aggregation techniques in WSN: A Survey Opinion Mining View project machine learning View project. *International Journal of Engineering and Innovative Technology (IJEIT)*, *1*(4).

Moschakis, I. A., & Karatza, H. D. (2015). Towards scheduling for Internet-of-Things

applications on clouds: a simulated annealing approach. *Concurrency and Computation: Practice and Experience*, *27*(8), 1886–1899. https://doi.org/10.1002/cpe.3105

Mukherjee, A., Jain, D. K., & Yang, L. (2021). On-Demand Efficient Clustering for Next Generation IoT Applications: A Hybrid NN Approach. *IEEE Sensors Journal*, *21*(22), 25457–25464. https://doi.org/10.1109/JSEN.2020.3026647

Mustafee Navonil, N. B. (2015). The Internet of Things: shaping the new Internet space. *Concurrency and Computation: Practice and Experience*, *27*(8), 1815–1818.

Nguyen, T. D., Le, D. T., Vo, V. V., Kim, M., & Choo, H. (2021). Fast Sensory Data Aggregation in IoT Networks: Collision-Resistant Dynamic Approach. *IEEE Internet of Things Journal*, *8*(2), 766–777. https://doi.org/10.1109/JIOT.2020.3007329

Noury, N., Hervé, T., Rialle, V., Virone, G., Mercier, E., Morey, G., Moro, A., & Porcheron, T. (2000). Monitoring behavior in home using a smart fall sensor and position sensors. *1st Annual International IEEE-EMBS Special Topic Conference on Microtechnologies in Medicine and Biology - Proceedings*. https://doi.org/10.1109/MMB.2000.893857

Özdoğan, E., & Ayhan, O. (2019). *HYBRID APPLICATION LAYER PROTOCOL DESIGN FOR IOT ENVIRONMENTS*. 267–286. https://doi.org/10.5121/csit.2019.91322

Papandriopoulos, J., Dey, S., & Evans, J. (2008). Optimal and distributed protocols for cross-layer design of physical and transport layers in MANETs. *IEEE/ACM*

Transactions on Networking, *16*(6), 1392–1405. https://doi.org/10.1109/TNET.2008.918099

Parmar, K., & Jinwala, D. C. (2016). Concealed data aggregation in wireless sensor networks: A comprehensive survey. In *Computer Networks* (Vol. 103, pp. 207–227). Elsevier B.V. https://doi.org/10.1016/j.comnet.2016.04.013

Peng, K., Huang, H., Wan, S., & Leung, V. C. M. (2020). End-edge-cloud collaborative computation offloading for multiple mobile users in heterogeneous edge-server environment. *Wireless Networks 2020*, 1–12. https://doi.org/10.1007/S11276-020-02385-1

Piccialli, F., Chianese, A., & Jung, J. J. (2017). Cultural Heritage on Internet of Things (IoT) systems: Trends and challenges. In *Concurrency Computation* (Vol. 29, Issue 11). https://doi.org/10.1002/cpe.4155

Piri, E., & Pinola, J. (2016). Performance of LTE uplink for IoT backhaul. *2016 13th IEEE Annual Consumer Communications and Networking Conference, CCNC 2016*, 6–11. https://doi.org/10.1109/CCNC.2016.7444723

Polyvyanyy, A., Ouyang, C., Barros, A., & van der Aalst, W. M. P. %J D. S. S. (2017). *Process querying: Enabling business intelligence through query-based process analytics*. *100*, 41–56.

Pourghebleh, B., & Navimipour, N. J. (2017). Data aggregation mechanisms in the Internet of things: A systematic review of the literature and recommendations for future research. In *Journal of Network and Computer Applications* (Vol. 97, pp. 23–34). Academic Press. https://doi.org/10.1016/j.jnca.2017.08.006

Prakash, T. S., Badrinatht, G. S., Venugopal, K. R., & Patnaik, L. M. (2006). Energy aware topology management in ad hoc wireless networks. *Second International Conference on Systems and Networks Communications, ICSNC 2006*. https://doi.org/10.1109/ICSNC.2006.36

Premila, D., Research Scholar, B. T., Rabara, A., & Research Scholar, J. A. (2015). Quality of Service Architecture for Internet of Things and Cloud Computing. In *International Journal of Computer Applications* (Vol. 128, Issue 7).

Qin, Y., Sheng, Q. Z., Falkner, N. J. G., Dustdar, S., Wang, H., & Vasilakos, A. V. (2016). When things matter: A survey on data-centric internet of things. *Journal of Network and Computer Applications*, *64*. https://doi.org/10.1016/j.jnca.2015.12.016

Qiu, T., Ding, Y., Xia, F., & Ma, H. %J S. (2012). *A search strategy of level-based flooding for the internet of things*. *12*(8), 10163–10195.

Quang, V. T., & Miyoshi, T. (2008). Adaptive Routing Protocol with Energy Efficiency and Event Clustering for Wireless Sensor Networks. *IEICE Transactions on Communications*, *E91.B*(9), 2795–2805. https://doi.org/10.1093/IETCOM/E91-B.9.2795

Rahman, H., Ahmed, N., & Hussain, I. (n.d.). Comparison of data aggregation techniques in Internet of Things (IoT). *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*.

Rajendranath, U. N. V. P., & Hency, V. B. (2019). Priority-based Task Pre-processing in IoT Sensory Environments. *Recent Patents on Engineering*, *14*(3), 357–365. https://doi.org/10.2174/1872212113666190515120232

Ramachandran, S., Presnell, B., & Richards, R. (2016). Serious games for team training and knowledge retention for long-duration space missions. *IEEE Aerospace Conference Proceedings*, *2016-June*. https://doi.org/10.1109/AERO.2016.7500503

Rao, S., & Shama, K. (2012). CROSS LAYER PROTOCOLS FOR MULTIMEDIA TRANSMISSION IN WIRELESS NETWORKS. *International Journal of Computer Science & Engineering Survey (IJCSES)*, *3*(3). https://doi.org/10.5121/ijcses.2012.3302

Raza, S., Misra, P., He, Z., & Voigt, T. (2017). Building the Internet of Things with bluetooth smart. *Ad Hoc Networks*, *57*, 19–31. https://doi.org/10.1016/j.adhoc.2016.08.012

Ren, J., Yu, G., He, Y., & Li, G. Y. (2019). Collaborative Cloud and Edge Computing for Latency Minimization. *IEEE Transactions on Vehicular Technology*, *68*(5), 5031–5044. https://doi.org/10.1109/TVT.2019.2904244

Riadi, I., Prayudi, Y., & Rizal, R. (n.d.). *Network Forensics for Detecting Flooding Attack on Internet of Things (IoT) Device*.

Robinson, K. (2010). *A Concise Summary of the Event B mathematical toolkit*.

Romanovsky, A., & Thomas, M. (2013). Industrial deployment of system engineering methods. In *Industrial Deployment of System Engineering Methods* (Vol. 9783642331). Springer-Verlag Berlin Heidelberg. https://doi.org/10.1007/978-3-642-33170-1

Romdhani, I., Qasem, M., Al-Dubai, A. Y., & Ghaleb, B. %J E. N. U. (2016). *Cooja*

simulator manual.

Ruan, C., Wang, J., Jiang, W., Min, G., & Pan, Y. (2020). PTCP: A priority-based transport control protocol for timeout mitigation in commodity data center. *Future Generation Computer Systems*, *102*, 619–632. https://doi.org/10.1016/J.FUTURE.2019.08.036

Sadek, R. A. (2018). Hybrid energy aware clustered protocol for IoT heterogeneous network. *Future Computing and Informatics Journal*, *3*(2), 166–177. https://doi.org/10.1016/j.fcij.2018.02.003

Salehi, S. A., Razzaque, M. A., Naraei, P., & Farrokhtala, A. (2013). Detection of sinkhole attack in wireless sensor networks. *2013 IEEE International Conference on Space Science and Communication (IconSpace)*, 361–365.

Sánchez López, T., Ranasinghe, D. C., Harrison, M., & McFarlane, D. (2012). Adding sense to the Internet of Things. *Personal and Ubiquitous Computing*. https://doi.org/10.1007/s00779-011-0399-8

Sándor, H., Genge, B., & Gál, Z. (2015). Security Assessment of Modern Data Aggregation Platforms in the Internet of Things. *Undefined*.

Sanyal, S., & Zhang, P. (2018). Improving quality of data: IoT data aggregation using device to device communications. *IEEE Access*, *6*, 67830–87840. https://doi.org/10.1109/ACCESS.2018.2878640

Shafagh, H., Hithnawi, A., Dröscher, A., Duquennoy, S., & Hu, W. (2015). Talos: Encrypted query processing for the Internet of Things. *SenSys 2015 - Proceedings*

of the 13th ACM Conference on Embedded Networked Sensor Systems, 197–210. https://doi.org/10.1145/2809695.2809723

Shafique, A., Cao, G., Aslam, M., Asad, M., & Ye, D. (2020). Application-Aware SDN-Based Iterative Reconfigurable Routing Protocol for Internet of Things (IoT). *Sensors 2020, Vol. 20, Page 3521*, *20*(12), 3521. https://doi.org/10.3390/S20123521

Sheng, Z., Yang, S., Yu, Y., Vasilakos, A., McCann, J., & Leung, K. (2013). A survey on the ietf protocol suite for the internet of things: Standards, challenges, and opportunities. *IEEE Wireless Communications*, *20*(6), 91–98. https://doi.org/10.1109/MWC.2013.6704479

Sirsikar, S., & Anavatti, S. (2015). Issues of data aggregation methods in Wireless Sensor Network: A survey. *Procedia Computer Science*, *49*(1), 194–201. https://doi.org/10.1016/j.procs.2015.04.244

Souza, AMC da and Amazonas, J. de. (2015). *A new internet of things architecture with cross-layer communication*. Proceedings of the 7th International Conference on Emerging Networks and Systems Intelligence Emerging.

Sruthi, S. S., & Geethakumari, G. (2016). An Efficient Secure Data Aggregation Technique for Internet of Things Network: An Integrated Approach Using DB-MAC and Multi-path Topology. *Proceedings - 6th International Advanced Computing Conference, IACC 2016*, 599–603. https://doi.org/10.1109/IACC.2016.116

Su, Jian and Liu, Alex X and Sheng, Zhengguo and Chen, Y. (2020). A partitioning approach to RFID identificatione. *IEEE/ACM Transactions on Networking*, *28*(5), 2160--2173.

Su, X., Riekki, J., Nurminen, J. K., Nieminen, J., & Koskimies, M. (2015). Adding semantics to internet of things. *Concurrency and Computation: Practice and Experience*, *27*(8), 1844–1860. https://doi.org/10.1002/cpe.3203

Sun, G., Chang, V., Ramachandran, M., Sun, Z., Li, G., Yu, H., & Liao, D. (2017). Efficient location privacy algorithm for Internet of Things (IoT) services and applications. *Journal of Network and Computer Applications*, *89*, 3–13. https://doi.org/10.1016/j.jnca.2016.10.011

Sun, G., Qi, J., Zang, Z., & Xu, Q. (2011). A reliable multipath routing algorithm with related congestion control scheme in wireless multimedia sensor networks. *ICCRD2011 - 2011 3rd International Conference on Computer Research and Development*, *4*, 229–233. https://doi.org/10.1109/ICCRD.2011.5763902

Talal, M., Zaidan, A. A., Zaidan, B. B., Albahri, A. S., Alamoodi, A. H., Albahri, O. S., Alsalem, M. A., Lim, C. K., Tan, K. L., Shir, W. L., & Mohammed, K. I. (2019). Smart Home-based IoT for Real-time and Secure Remote Health Monitoring of Triage and Priority System using Body Sensors: Multi-driven Systematic Review. *Journal of Medical Systems 2019 43:3*, *43*(3), 1–34. https://doi.org/10.1007/S10916-019-1158-Z

Tan, J., & Koo, S. G. M. (2014). A survey of technologies in internet of things. *Proceedings - IEEE International Conference on Distributed Computing in Sensor Systems, DCOSS 2014*, 269–274. https://doi.org/10.1109/DCOSS.2014.45

Tan, L., & Wang, N. (2010). Future Internet: The Internet of Things. *ICACTE 2010 - 2010 3rd International Conference on Advanced Computer Theory and*

Engineering, Proceedings. https://doi.org/10.1109/ICACTE.2010.5579543

Tandon, A., Kumar, P., Rishiwal, V., Yadav, M., & Yadav, P. (2021). A bio-inspired hybrid cross-layer routing protocol for energy preservation in WSN-assisted IoT. *KSII Transactions on Internet and Information Systems*, *15*(4), 1317–1341. https://doi.org/10.3837/tiis.2021.04.008

Thakare, A., Lee, E., Kumar, A., Nikam, V. B., & Kim, Y. G. (2020). PARBAC: Priority-Attribute-Based RBAC Model for Azure IoT Cloud. *IEEE Internet of Things Journal*, *7*(4), 2890–2900. https://doi.org/10.1109/JIOT.2019.2963794

Thomson, C., Romdhani, I., Al-Dubai, A., Qasem, M., Ghaleb, B., & Wadhaj UK, I. %J E. N. U. E. (2016). *Cooja Simulator Manual*.

Tripathi, A., Gupta, S., Chourasiya, B. (2014). Survey on data aggregation techniques for wireless sensor networks. *Int. J. Adv. Res. Comput. Commun. Eng,* 7366–7371.

Tsai, C. W., Lai, C. F., & Vasilakos, A. V. (2014). Future Internet of Things: open issues and challenges. *Wireless Networks*, *20*(8), 2201–2217. https://doi.org/10.1007/s11276-014-0731-0

Ullah, M. F., Imtiaz, J., & Maqbool, K. Q. (2019). Enhanced Three Layer Hybrid Clustering Mechanism for Energy Efficient Routing in IoT. *Sensors 2019, Vol. 19, Page 829*, *19*(4), 829. https://doi.org/10.3390/S19040829

Upadhyayula, S., & Gupta, S. K. S. (2007). Spanning tree based algorithms for low latency and energy efficient data aggregation enhanced convergecast (DAC) in wireless sensor networks. *Ad Hoc Networks*, *5*(5), 626–648.

https://doi.org/10.1016/j.adhoc.2006.04.004

VasudevanVijay, PhanishayeeAmar, ShahHiral, KrevatElie, G., A., R., G., A., G., & MuellerBrian. (2009). Safe and effective fine-grained TCP retransmissions for datacenter communication. *ACM SIGCOMM Computer Communication Review*, *39*(4), 303–314. https://doi.org/10.1145/1594977.1592604

Verma, N., & Singh, D. (2018). Data Redundancy Implications in Wireless Sensor Networks. *Procedia Computer Science*, *132*, 1210–1217. https://doi.org/10.1016/j.procs.2018.05.036

Vithya, G., & Vinayagasundaram, B. (2014). QOS by priority routing in internet of things. *Research Journal of Applied Sciences, Engineering and Technology*, *8*(21), 2154–2160. https://doi.org/10.19026/rjaset.8.1213

Wan, P., & Lemmon, M. D. (2009). Event-triggered distributed optimization in sensor networks. *2009 International Conference on Information Processing in Sensor Networks, IPSN 2009*.

Wan, S., Zhao, Y., Wang, T., Gu, Z., Abbasi, Q. H., & Choo, K. K. R. (2019). Multi-dimensional data indexing and range query processing via Voronoi diagram for internet of things. *Future Generation Computer Systems*, *91*, 382–391. https://doi.org/10.1016/j.future.2018.08.007

Wang, Y., & Zhang, S. (2016). An enhanced dynamic priority packet scheduling algorithm in wireless sensor networks. *Proceedings - 2016 UKSim-AMSS 18th International Conference on Computer Modelling and Simulation, UKSim 2016*, 311–316. https://doi.org/10.1109/UKSIM.2016.31

Werner-Allen, G., Swieskowski, P., & Welsh, M. (2005). MoteLab: A wireless sensor network testbed. *2005 4th International Symposium on Information Processing in Sensor Networks, IPSN 2005*. https://doi.org/10.1109/IPSN.2005.1440979

White, G., Nallur, V., Clarke, S. %J J. of S., & Software. (2017). *Quality of service approaches in IoT: A systematic mapping*. *132*, 186–203.

Whitmore, A., Agarwal, A., & Da Xu, L. (2015). The Internet of Things—A survey of topics and trends. *Information Systems Frontiers*, *17*(2). https://doi.org/10.1007/s10796-014-9489-2

Wu, M., Lu, T. J., Ling, F. Y., Sun, J., & Du, H. Y. (2010). Research on the architecture of Internet of Things. *ICACTE 2010 - 2010 3rd International Conference on Advanced Computer Theory and Engineering, Proceedings*, *5*. https://doi.org/10.1109/ICACTE.2010.5579493

Xie, F. (2014). CaCa: Chinese Remainder Theorem Based Algorithm for Data Aggregation in Internet of Things on Ships. *Applied Mechanics and Materials*, *701–702*, 1098–1101. https://doi.org/10.4028/www.scientific.net/amm.701-702.1098

Xie, F., & Ye, X. H. (2015). Endada: An Efficient Network Design Algorithm Based on Weighted Graph for Data Aggregation in Internet of Things on Marine Ships. *Applied Mechanics and Materials*, *740*, 648–651. https://doi.org/10.4028/www.scientific.net/amm.740.648

Xie, R., Yu, F. R., & Ji, H. (2012). Dynamic resource allocation for heterogeneous services in cognitive radio networks with imperfect channel sensing. *IEEE Transactions on Vehicular Technology*, *61*(2), 770–780.

https://doi.org/10.1109/TVT.2011.2181966

Xiong, Y., Sun, Y., Xing, L., & Huang, Y. (2018). Extend cloud to edge with KubeEdge. *Proceedings - 2018 3rd ACM/IEEE Symposium on Edge Computing, SEC 2018*, 373–377. https://doi.org/10.1109/SEC.2018.00048

Xu, L. Da, He, W., & Li, S. (2014). Internet of things in industries: A survey. In *IEEE Transactions on Industrial Informatics* (Vol. 10, Issue 4, pp. 2233–2243). IEEE Computer Society. https://doi.org/10.1109/TII.2014.2300753

Xu, X., Yuan, M., Liu, X., Liu, A., Xiong, N. N., Cai, Z., & Wang, T. (2018). A Cross-Layer Optimized Opportunistic Routing Scheme for Loss-and-Delay Sensitive WSNs. *Sensors 2018, Vol. 18, Page 1422*, *18*(5), 1422. https://doi.org/10.3390/S18051422

Xue, Y., Ramamurthy, B., & Vuran, M. C. (2011). SDRCS: A service-differentiated real-time communication scheme for event sensing in wireless sensor networks. *Computer Networks*, *55*(15), 3287–3302. https://doi.org/10.1016/j.comnet.2011.06.018

Yamazaki, S., Abiko, Y., & Mizuno, H. (2020). A Simple and Energy-Efficient Flooding Scheme for Wireless Routing. *Wireless Communications and Mobile Computing*, *2020*. https://doi.org/10.1155/2020/8832602

Yan-e, D. (2011). Design of intelligent agriculture management information system based on IoT. *2011 Fourth International Conference on Intelligent Computation Technology and Automation*, *1*, 1045–1049.

Yan, J., Zhou, M., & Ding, Z. (2016). Recent Advances in Energy-Efficient Routing Protocols for Wireless Sensor Networks: A Review. In *IEEE Access* (Vol. 4, pp. 5673–5686). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ACCESS.2016.2598719

Yan, Z., Zhang, P., & Vasilakos, A. V. (2014). A survey on trust management for Internet of Things. *Journal of Network and Computer Applications*, *42*, 120–134. https://doi.org/10.1016/j.jnca.2014.01.014

Yang, H., Zhang, X., & Wang, Y. (2006). A correctness proof of the SRP protocol. *20th International Parallel and Distributed Processing Symposium, IPDPS 2006*, *2006*, 7–13. https://doi.org/10.1109/IPDPS.2006.1639687

Yick, J., Mukherjee, B., & Ghosal, D. (2008). Wireless sensor network survey. *Computer Networks*. https://doi.org/10.1016/j.comnet.2008.04.002

Young Han Nam, Zeehun Halm, Young Joon Chee, & Kwang Suk Park. (2002). *Development of remote diagnosis system integrating digital telemetry for medicine*. https://doi.org/10.1109/iembs.1998.747079

Yu, F., & Krishnamurthy, V. (2007). Optimal joint session admission control in integrated WLAN and CDMA cellular networks with vertical handoff. *IEEE Transactions on Mobile Computing*, *6*(1), 126–139. https://doi.org/10.1109/TMC.2007.250676

Yu, F., & Leung, V. (2002). Mobility-based predictive call admission control and bandwidth reservation in wireless cellular networks. *Computer Networks*, *38*(5), 577–589. https://doi.org/10.1016/S1389-1286(01)00269-9

Yuan, K. L., Qiao, L., & Han, L. (2013). Level and cluster based routing for wireless sensor network. *Applied Mechanics and Materials*, *321–324*, 515–522. https://doi.org/10.4028/www.scientific.net/AMM.321-324.515

Zappi, P., Farella, E., & Benini, L. (2010). Tracking motion direction and distance with pyroelectric IR sensors. *IEEE Sensors Journal*, *10*(9), 1486–1494. https://doi.org/10.1109/JSEN.2009.2039792

Zhang, C., Lai, C. F., Lai, Y. H., Wu, Z. W., & Chao, H. C. (2017). An inferential real-time falling posture reconstruction for Internet of healthcare things. *Journal of Network and Computer Applications*, *89*, 86–95. https://doi.org/10.1016/j.jnca.2017.02.006

Zhang, J., Ren, F., Shu, R., & Cheng, P. (2016). TFC: Token flow control in data center networks. *Proceedings of the 11th European Conference on Computer Systems, EuroSys 2016*. https://doi.org/10.1145/2901318.2901336

Zhang, J., Ren, F., Tang, L., & Lin, C. (2013). Taming TCP incast throughput collapse in data center networks. *Proceedings - International Conference on Network Protocols, ICNP*. https://doi.org/10.1109/ICNP.2013.6733609

Zhang, J., Ren, F., Yue, X., Shu, R., & Lin, C. (2014). Sharing bandwidth by allocating switch buffer in data center networks. *IEEE Journal on Selected Areas in Communications*, *32*(1), 39–51. https://doi.org/10.1109/JSAC.2014.140105

Zhang, P., Wang, J., Guo, K., Wu, F., & Min, G. (2018). Multi-functional secure data aggregation schemes for WSNs. *Ad Hoc Networks*, *69*, 86–99. https://doi.org/10.1016/j.adhoc.2017.11.004

Zhou, M., & Ma, Y. (2013). QoS-aware computational method for IoT composite service. *Journal of China Universities of Posts and Telecommunications*, *20*(SUPPL. 1), 35–39. https://doi.org/10.1016/S1005-8885(13)60252-6

Zhou, Z., Tang, J., Zhang, L. J., Ning, K., & Wang, Q. (2014). EGF-tree: An energy-efficient index tree for facilitating multi-region query aggregation in the internet of things. *Personal and Ubiquitous Computing*, *18*(4), 951–966. https://doi.org/10.1007/s00779-013-0710-y

Zhu, L., Yu, F. R., Ning, B., & Tang, T. (2011). Cross-layer design for video transmissions in metro passenger information systems. *IEEE Transactions on Vehicular Technology*, *60*(3), 1171–1181. https://doi.org/10.1109/TVT.2011.2107927

Zhu, L., Yu, F. R., Ning, B., & Tang, T. (2012). Cross-layer handoff design in MIMO-enabled WLANs for Communication-Based Train Control (CBTC) systems. *IEEE Journal on Selected Areas in Communications*, *30*(4), 719–728. https://doi.org/10.1109/JSAC.2012.120506

Zhu, T., Dhelim, S., Zhou, Z., Yang, S., & Ning, H. (2017). An architecture for aggregating information from distributed data nodes for industrial internet of things. *Computers and Electrical Engineering*, *58*, 337–349. https://doi.org/10.1016/j.compeleceng.2016.08.018

Zimos, E., Mota, J. F. C., Rodrigues, M. R. D., & Deligiannis, N. (2016, June). Internet-of-Things data aggregation using compressed sensing with side information. *2016 23rd International Conference on Telecommunications, ICT 2016*.

https://doi.org/10.1109/ICT.2016.7500418

Zorzi, M., Gluhak, A., Lange, S., & Bassi, A. (2010). From today's INTRAnet of things to a future INTERnet of things: A wireless- and mobility-related view. *IEEE Wireless Communications*. https://doi.org/10.1109/MWC.2010.5675777