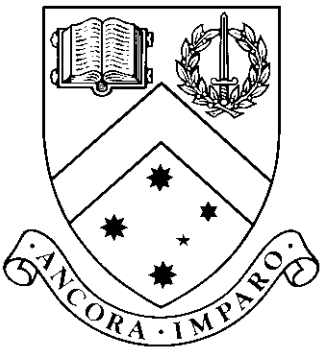
Nearest Neighbour Query Processing in Mobile P2P Networks

by

Thao P. Nghiem, MSc



Thesis

Submitted by Thao P. Nghiem

for fulfilment of the requirements for the degree of

Doctor of Philosophy (0190)

Supervisor: A/Prof. David Taniar

Associate Supervisor: Prof. David Green

External Supervisor: Dr. Agustinus Borgy Waluyo

Clayton School of Information Technology
Monash University

April, 2014

Contents

List of Tables

List of Figures

Nearest Neighbour Query Processing in Mobile P2P Networks

Thao P. Nghiem, MSc
[redacted]
Monash University, 2014

Supervisor: A/Prof. David Taniar
Associate Supervisor: Prof. David Green
External Supervisor: Dr. Agustinus Borgy Waluyo

Abstract

The mature growth of mobile technologies in both hardware and software has paved the way for applying Peer-to-Peer (P2P) to location-aware applications in general and to nearest neighbour query processing in particular. Indeed, mobile devices powered by energy-efficient and high-performance microprocessors now allow people to share messages, sounds and pictures. In addition, wireless communication technologies have been developing dramatically, which enables mobile devices from anywhere to communicate at any time without requiring a wireless access point. P2P query processing not only harnesses the power of collaboration between peers, but also overcomes the drawbacks of centralised systems, such as bottleneck issues and low fault-tolerant problems. Therefore, making use of P2P query processing is a natural and essential move in spatial database management.

The study in this thesis revolves around the central question: "Can the P2P approach solve Spatial Nearest Neighbour Queries in Mobile Environments?". In order to address the question, a pure mobile P2P query processing system is proposed to answer spatial queries. Although the proposed system is able to provide services to a wide range of spatial query types, the focus of the thesis presents as a case study, one of the most common types of spatial queries: Nearest Neighbours in a two-dimension Euclidean space with mobile peers and static objects of interest.

In particular, this thesis introduces the P2P query processing system to tackle k Nearest Neighbours (kNNs), Reverse Nearest Neighbours (RNNs) and Group Nearest Neighbours (GkNNs). This system eliminates the role of a base station and focuses on the power of peer collaboration. Rather than queries are sent to all peers, a time-out mechanism, result validation algorithms and peer pruning algorithms are proposed to filter unnecessary peer communication. As a result, the proposed system can decrease query processing time and energy consumption while maintaining a high rate of accuracy. The experimental study, which uses both real and synthetic data sets, shows the practical feasibility of the P2P approach in solving Nearest Neighbour queries for mobile networks with the high density of moving objects.

Nearest Neighbour Query Processing in Mobile P2P Networks

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Thao P. Nghiem
April 19, 2014

Acknowledgments

I wish to thank, first and foremost, my supervisor, Associate Professor David Taniar. He was the one who supported my PhD application to Monash University when I finished my Master degree in South Korea. Since then, he has provided valuable advice on my PhD study. This thesis would have remained a dream had it not been for his continuous guidance, patience, encouragement and immense knowledge. I also would like to express my sincere gratitude to Associate Professor Wenny Rahayu from Latrobe University for introducing me such a great supervisor.

My deepest appreciation and respect go to my co-supervisor, Professor David Green for his helpful advice and insight. His attitude to research inspired me to continue to a PhD program and want to be a member of the academic family. He is not only a responsible and knowledgeable professor but also a friendly and considerate colleague when we worked together for a few units at Monash University. Special thanks to Dr Marc Cheong who recommended me to appoint Professor David Green as my co-supervisor.

It gives me great pleasure in acknowledging the supervision of Dr Agustinus Borgy Waluyo, Adjunct Research Fellow (Monash University). His insightful comments and warm encouragement have been a great help to provide a foundation of my study. Even though he is now working in industry, every now and then he sends me useful feedbacks on my research papers and this thesis.

I would like to offer my special thanks to my cheerful and friendly research group of fellow students: Geng Zhao, Haidar Al-Khalidi, Kefeng Xuan, Kiki Maulana Adhinugraha and Sultan Amri. Our group meetings and discussions are valuable input to my research. It has been a very fun and nice experience to work with them throughout the years. I share the credit of my work with Kiki Maulana Adhinugraha for his valuable help in developing Reverse Nearest Neighbour search algorithms. He is also the co-author of two research papers, extracted from this thesis.

I would also like to thank Faculty of IT and MIGR, Monash University for providing research facilities and support. In particular, I would like to thank Professor Bala Srinivasan, Professor Graham Farr, Associate Professor Vincent Lee, Dr Nandita Bhattacharjee and Dr John Betts for their valuable input and discussion through my annual candidature review and pre-submission seminar. Also thank

you to Ms Leeanne Evans, Ms Sevim Zongur, Ms Kerry Mcmanus and Mr Akamon Kunkongkapun, for their excellent faculty research, administration and financial services that make my PhD life easier. I am particularly grateful for the assistance given by HDR lunch-time seminar committee who give me a forum to discuss my research with other research students.

I am lucky to have many interesting PhD fellows in my office and next to my office around. Thank you to Upuli, Hiran, Fatima, Nabeel, River, Daniel, Parman and the other for all the fun we have for the last three years. In addition, I would like to gratefully and sincerely thank Monash Rockstars, a bunch of great guys, including Ricky, Ashwin, Arnie, Hoang, Aisa, Michael, Himanshu, Sepehr and many more, from different countries, different background for our badminton nights, our day trips, interesting conversations and warm friendship that can keep my work-life balance. Special thanks to again Ashwin and Ricky for their excellent proofreading this thesis.

Finally, and most importantly, I would like to express my deepest gratitude to my partner, Hasn (Bazon), who is always beside me through the long journey of my PhD candidature. This thesis would not happen to be completed on time unless he has been greatly tolerant and supportive in many ways. He is my love, brother, best friend and sometimes a super responsible supervisor, all in one. I sincerely thank my family in Vietnam. My Dad is always concerned about my research and he gives me invaluable advice as an experienced researcher. My Mom always asks me to eat more and take care of my health. My sister is always a best friend whenever I need her most. I would like to express my gratitude to them in Vietnamese.

Cuối cùng con cũng viết xong luận văn. Đi được đến bước này là nhờ bệ phóng vững chắc từ những ngày quấn quýt bên gia đình ở nhà mình. Cảm ơn ba mẹ đã luôn yêu thương, chăm sóc, giúp đỡ con. Cảm ơn chị đã luôn lắng nghe những tâm sự buồn vui của em gái. Cảm ơn anh Hòa và Rô, Na làm gia đình mình thêm hạnh phúc và trọn vẹn hơn. Mong gia đình mình mãi vui vẻ và thật nhiều sức khỏe. Lời cuối dành gửi lời cảm ơn đến dì Bình, người thương con như người mẹ thứ hai và luôn bên cạnh mẹ con; cậu Thanh, người đầu tiên, ngoài ba mẹ, luôn động viên và khích lệ con trên con đường học vấn.

Thao P. Nghiem

Monash University

April 2014

"If we knew what it was we were doing, it would not be called research, would it?"

Albert Einstein

1 Introduction

1.1 Overview

Recently we have experienced a fast evolution in both mobile hardware and software. Energy-efficient microprocessors power mobile devices that allow people to share words, sounds, and images. In fact, our smart phone or tablet has become more powerful than our parents' computer. In 2012, Google introduced its Nexus tablet featuring quad-core processor in San Francisco while LG, HTC and Huawei have recently presented their quad-core phones running Ice Cream Sandwich (Google's Android 4.0) from the outset. In addition, wireless communication technologies have been developing dramatically. Some examples of familiar technologies to support mobile communication are Wireless Local Area Network (WLAN), Bluetooth,

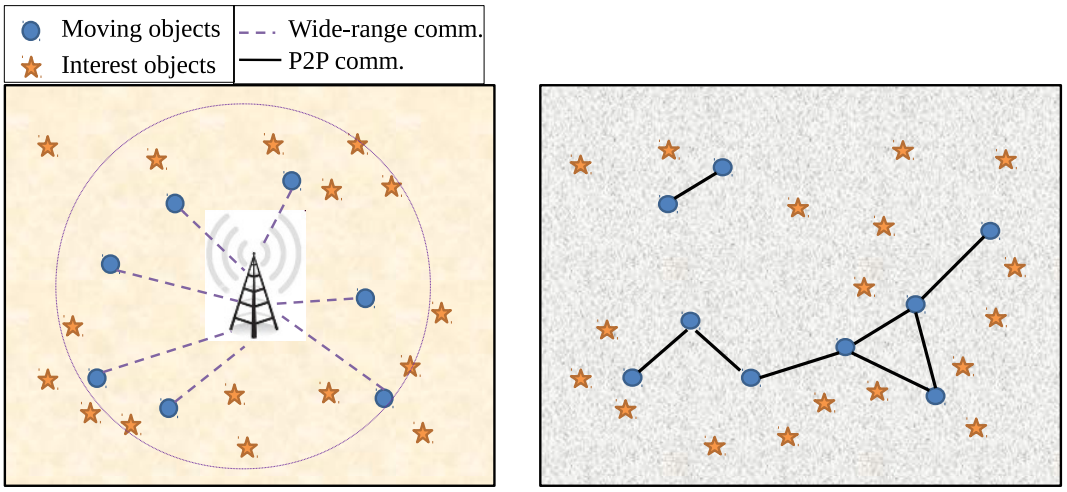
IEEE802.15.4/ZigBee and Ultra-Wide Band (UWB). Furthermore, Wi-Fi Direct, the emerging form of Peer-to-Peer communication, makes data sharing among mobile peers easy from anywhere at any time without requiring a wireless access point. It is time to harness the computing power, intelligence and functionality of mobile devices for geographic searching in general and for spatial database management in particular.

Traditionally, spatial data management has been achieved by centralised database systems that rely on a central base station to store data and handle all queries from clients. Based on spatial data management, location-aware services which retrieve or answer spatial queries have received much interest due to their increased importance in advanced database applications (Sistla et al., 1997; Kollios et al., 1999). One of the most important branches of location-aware services is nearest neighbour search (Li et al., 2010).

In the literature, much research deals with conventional spatial query processing based on a centralised base station (Dittrich et al., 2009; Mouratidis et al., 2005; Saltenis et al., 2000; Xiong et al., 2005; Yu et al., 2005) as seen in Fig. 1.1a.

Scalability is the biggest limitation of those centralised approaches. This is due to the high cost of frequent location updates and query processing. Bottlenecks and low fault-tolerant problems are inevitable consequences in large-scale systems. In particular, those systems only contain a central point of failure which is likely to be corrupt in several circumstances. For example, if there is an accident, such as a natural disaster, communication to the headquarters may not be available or jammed or even attacked by hackers (Ku and Zimmermann, 2008; Ali et al., 2008).

In response to the limitations of centralised query processing systems and given the advances of mobile technologies, the emergence of mobile peer-to-peer (P2P) query processing systems provides a promising solution. Unlike client-server computing, Mobile P2P is a high dynamic environment where mobile peers can join and leave networks at any time. P2P systems adopt a completely decentralised approach to resource management. By distributing data storage, processing, and bandwidth



(a) Centralised Systems **(b) P2P Systems**

Figure 1.1: Centralised Systems versus P2P Systems. Circles represent moving objects; stars: objects of interest; dashed lines: wide-range communication; continuous lines: P2P communication; dots: the Base station network range.

across autonomous peers in the network, they can scale without the need for powerful servers.

A typical mobile P2P network consists of a collection of moving objects (Fig. 1.1b) that are equipped with a Global Positioning System (GPS) and are able to communicate with each other in a P2P manner to share common interests via short-range wireless technology standards such as Bluetooth, WiFi, WiFi-Direct or UWB, etc. (Luo et al., 2010; Tao, Papadias and Lian, 2004; Xu et al., 2009). When a moving object q invokes a Nearest Neighbour query, it sends a query message to surrounding peers that are in its communication range instead of performing wide-range communication to the base station. In order to reply to the query from q, neighbouring peers return their cached data, for example, their nearest neighbours retrieved two minutes ago. Using data from its cache and data received from its peers about objects of interest, q makes a selection to answer its queries.

Let us consider a real-life example which can be addressed in this thesis. In March 2011, the earthquake and tsunami struck Japan. The access infrastructure that is essential to support communication systems was severely affected. 879,000 telephony lines, 475,400 fibre-optic and more than 11,000 wireless base stations

belonging to DoCoMo, KDDI and Softbank were out of service as of the following day of the devastating disaster (TeleGeography, 2011; timesofindia, 2011; NBCNews, 2011). As a consequence, local people were disconnected from the centralised base station. The system which is proposed in this thesis can be a communication solution in this situation. A victim in that area can take advantage of his smart phone to contact other people within his communication range and find out the nearest shelters via Bluetooth or WLAN. It is likely that some people around him know such information and their mobile devices could store the answer for his query. This is an illustration of kNN query processing in a Mobile P2P environment, where collaborative communication among peers is the only possible way of contact when a natural disaster happens.

Based on the above needs, the central research question in this thesis is **"Can P2P approach solve Spatial Nearest Neighbour Queries in Mobile Environments?"**. To address this question, I will investigate several of the challenges, which are described in the section below, and detail the matters I will investigate in Section 1.3 Study Scope.

1.2 Major Challenges

To answer the central research question above, first it is essential to be aware of features of P2P approaches that make it difficult to provide advanced data management services over mobile P2P networks. Although the characteristics of Mobile P2P Query Processing Systems bring many benefits for location-aware services and resolve existing problems in Centralised Query Processing Systems, they have to face three major challenges as below:

- **Challenge 1**: Mobile P2P Networks consist of moving peers such as mobile phones and PDA, which have short-life battery and limited computational power and memory. In addition, network bandwidth restriction should also be taken into account. Heavy centralised indexing and searching algorithms

(Roussopoulos et al., 1995; Cheung and Fu, 1998; Hjaltason and Samet, 1999) are, therefore, no longer suitable for our Mobile P2P Query Processing System since they consume much energy, require much space for data storage such as R-Tree and process slowly when implemented in each peer. The first challenge is to design a light framework that is suitable for processing complex spatial queries in Mobile P2P Networks while it saves memory space, query processing time and reduces power consumption.

- **Challenge 2**: In P2P systems, the answers to queries are typically incomplete. The reason is that some peers may be absent at query execution time since mobile peers can join and leave the network at any time. There is also no centralised catalogue that can be used to determine which peers that hold data relevant to a query. Hence, maintaining the high accuracy rate is a real challenge when developing spatial query processing for Mobile P2P Networks.

- **Challenge 3**: Last but not least, due to the very large scale of the network, forwarding a query to all peers can be very inefficient in term of energy consumption and processing time. Therefore, it is vital to select appropriate peers to be involved in query processing.

One of the main location-aware services which are needed for supporting advanced P2P applications is a query processing service that deals with Nearest Neighbour queries. However, providing such a service in P2P systems is challenging because of the specific features listed above. Most techniques designed for centralised spatial database systems, which statically exploit base station and network information, no longer apply. New techniques are needed and should be decentralised, dynamic and self-adaptive. Therefore, novel techniques will be proposed in this thesis to gather and filter spatial data from mobile peers to answer queries, especially complex spatial queries such as Nearest Neighbour queries, in a fully distributed fashion while taking into account the dynamic behaviour of peers.

My contribution to the central question stated in Section 1.1 will be to investigate 3 cases arising from nearest neighbour queries.

1.3 Study Scope

Although the P2P systems that will be proposed in this thesis are able to provide services to a wide range of spatial query types, we focus on researching one of the most common types of spatial queries which is Nearest Neighbours in two dimension Euclidean space due to time constraint of doing a PhD candidature. In particular, three types of Nearest Neighbours are selected and investigated in P2P Systems with mobile peers and static objects of interest. The concept of each type of query is briefly introduced as below:

1.3.1 k Nearest Neighbours

This is one of the most conventional problems in location based services. The problem of finding k nearest neighbours (kNN) is identifying k objects of interest from a set of all interest objects in the network such that these k objects are the nearest to a given query location q according to a specific distance metric (see Fig. 1.2). For instance, a traveller needs to find two nearest hotels. Another example of a P2P scenario: In case of an earthquake, local people are disconnected from the centralised base station. A victim in that area can take advantage of his smart phone to contact other people within his communication range and find out the nearest shelters.

1.3.2 Reverse Nearest Neighbours

The problem is raised from the objects' point of view. Instead of finding the nearest objects from the query point q, it asks which objects consider q as their nearest neighbour. There are two types of RNNs; firstly monochromatic RNN, in which query points and objects of interest are of the same category, and secondly bichromatic RNN (BRNN), in which they are of different categories (see Fig. 1.3). Within the

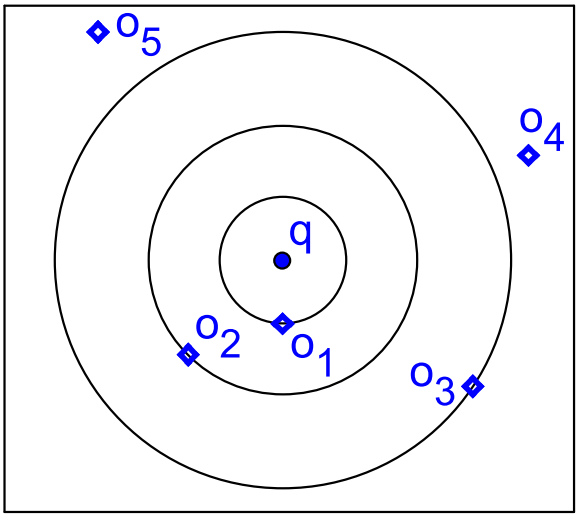


Figure 1.2: k Nearest Neighbours. o_1, o_2, o_3 are 3NNs of query node q (k=3)

scope of our research, we focus on the latter category, which has recently attracted a growing number of studies in a wide range of applications, such as decision support systems, mobile navigation systems and resource allocation.

For instance, in everyday applications, police patrols can communicate with each other to distribute their team members to locations that need them most, for example, sites of car accidents or intersections congested with traffic. As another example in a commercial application, Outbreaks Near Me in 2009, allows users to use their iPhone to see all current outbreaks new strain of A(H1N1) virus or swine flu in their neighbourhood, including news about swine flu. In relation to our work, these applications can run BRNN queries to find the set of sites taking the victims' location as the nearest neighbours. As a result, this information is helpful not only for international travellers but also for governments to detect and prevent infectious diseases in time.

1.3.3 Group k Nearest Neighbours

The growing importance of location-based services has led to a new range of real-time services such as location-based social networking (e.g Friend Finder (Strassman and Collier, 2004)) that enable a group of users to be involved in a single location-based

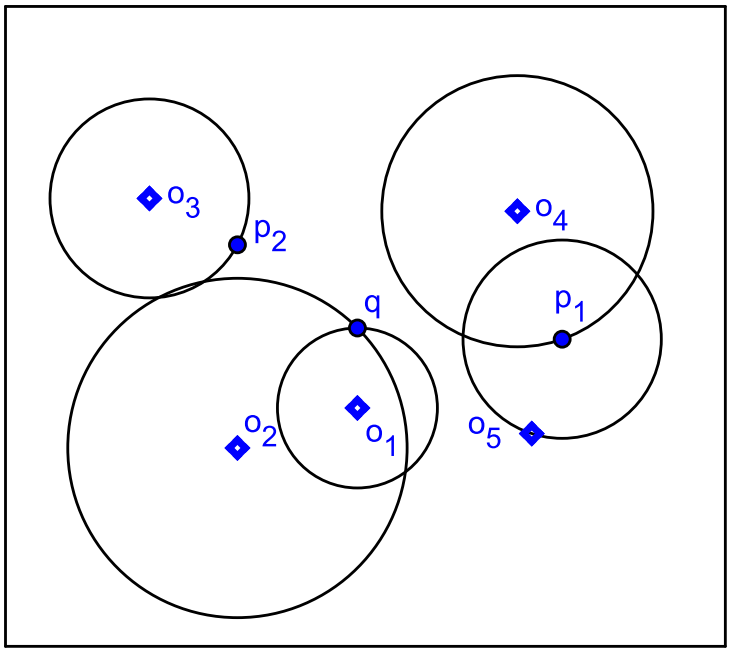


Figure 1.3: Reverse Nearest Neighbours. Nodes o_1, o_2 are the RNNs of query node q because q is the NN of o_1 and o_2. Similarly, o_4 and o_5 are the RNNs of p_1; and o_3 is the RNN of p_2

query (see Fig. 1.4). For example, a group of users may want to meet at a place that minimises the total travel distance for them.

1.4 Contributions

In order to address the central research question above, this thesis introduces a new approach to solve nearest neighbour queries. Several novel, pure P2P, search algorithms are proposed to harness the collaborative power of mobile P2P and eliminate the dependence on a centralised unit. In particular, the contribution of this thesis is to investigate three cases as below.

1.4.1 kNN P2P Query Processing

We propose a pure mobile P2P query processing system which focuses primarily on the search and validation algorithm for kNN queries. The proposed system is designed for pure mobile P2P environments with the absence of a base station

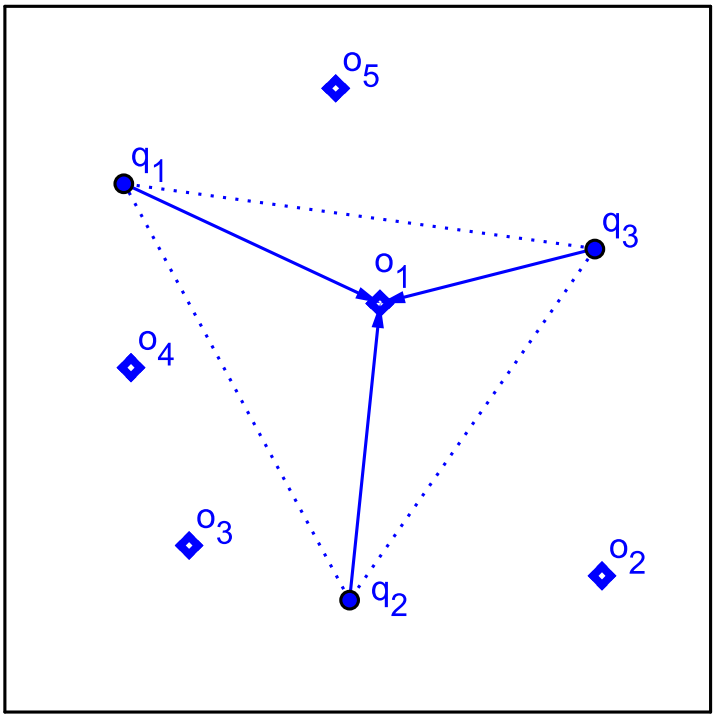


Figure 1.4: An illustration of a GNN query. $Q = \{q_1, q_2, q_3\}$ is a group of query points. o_1 is the GNN since $\sum_{i=1}^{3} distance(q_i, o_1)$ is the minimum compared to other objects of interest (rhombus symbols).

support. Compared with centralised and hybrid systems, my system can significantly reduce energy consumption in a reasonable mean latency of processing time for networks with high density of moving objects. The main underlying idea is to make use of data sharing from peers. More specifically, the heavy central index tree is no longer required. The answer of queries is based on cached data from peers. Therefore, the peers can save memory space and locally process queries via short-range communication. In addition, the queries are not broadcast to all peers but subsequently sent to the closest peers in the priority queue until the queries are answered or the pre-set time-out expires. The determination of whether the answer from peers are sufficient is based on validation algorithm, which tells whether the retrieved objects of interest from peers can be target kNNs of q . Hence, the system

can minimise both mean latency time of query processing and the overhead cost of communication.

This research (Nghiem and Waluyo, 2011) was presented in *International Conference on Advances in Mobile Computing and Multimedia (MoMM)* 2011. The extended version (Nghiem, Waluyo and Taniar, 2013) appears in *Journal of Personal and Ubiquitous Computing* with extensive experiments to investigate the effects of various factors such as expected number of of queries, number of queried nearest objects, number of interest objects and number of moving objects for all measures that are communication cost, mean latency of query processing and the percentage of queries resolved. Also in this paper, rather than running the simulation in a small-scale network which is far from the reality, we focus on collecting real data of Melbourne Inner City, Australia from Australian Bureau of Statistics. In addition, the connection between the moving object and the base station is conducted via 3G(WCDMA), band I-2100 which is used by Vodaphone, Australia.

1.4.2 RNN P2P Query Processing

Two original P2P algorithms for bichromatic RNN queries are proposed in this study. In this type of query, mobile query objects and static objects of interest are of two different categories. To reduce the cost of processing query, the algorithms, based on a boundary polygon around the mobile query object, pruned peers that are less likely to provide useful data. In particular, the Brute-Force Search Algorithm (BFA) makes use of all information from the peers to aim at high accuracy rate. The Regular Boundary Search Algorithm (RBA) reduces the number of queried peers. The results show the practical feasibility of the P2P approach in solving bichromatic RNN queries for mobile networks.

This research was published in *IEEE International Conference on Pervasive Computing and Communications (PERCOM)* 2013 **(Nghiem, Maulana, Waluyo, Green and Taniar, 2013)**. From the initial work, we introduce a new search

algorithm named Tight Boundary Search Algorithm (TBA) in which a significant number of peers are filtered in processing an RNN query.

From our experimental study involving a real data set, we found that our proposed system is substantially more energy-efficient and save up to 43% latency time compared with the centralised system. Our simulation of three proposed algorithms also shows that the accuracy rate of BFA is higher than that of RBA in most scenarios. Vice versa, the RBA reduces query response time when a number of queried peers are pruned. Nevertheless, TBA is the most optimal algorithm when the movement of moving objects is not too fast. This extended work was accepted with revision by *Journal of Parallel and Distributed Computing (JPDC)* (Nghiem, Maulana, Green, Waluyo and Taniar, 2013).

1.4.3 GkNN P2P Query Processing

We developed a novel P2P algorithm focusing on GkNN queries. P2P-GkNN makes use of all cached nearest neighbours from the peers to answer GkNN in the most efficient way in terms of overhead cost, processing time and accuracy rate. From our experimental study involving both real and synthetic data set, we found that our proposed algorithm is substantially more energy-efficient and save at least 32% latency time compared with the centralised GkNN algorithm while maintaining high accuracy rate more than 82%.

The research was presented in *IEEE International Conference on Parallel and Distributed Systems (ICPADS)* in December, 2013 (Nghiem, Green and Taniar, 2013). The work can be extended to filter the number of peers involved in the query processing to reduce overhead costs of communication and computation and also processing time.

1.5 Study Organisation

This study is organised as in Fig. 1.5. The main components are listed as below:

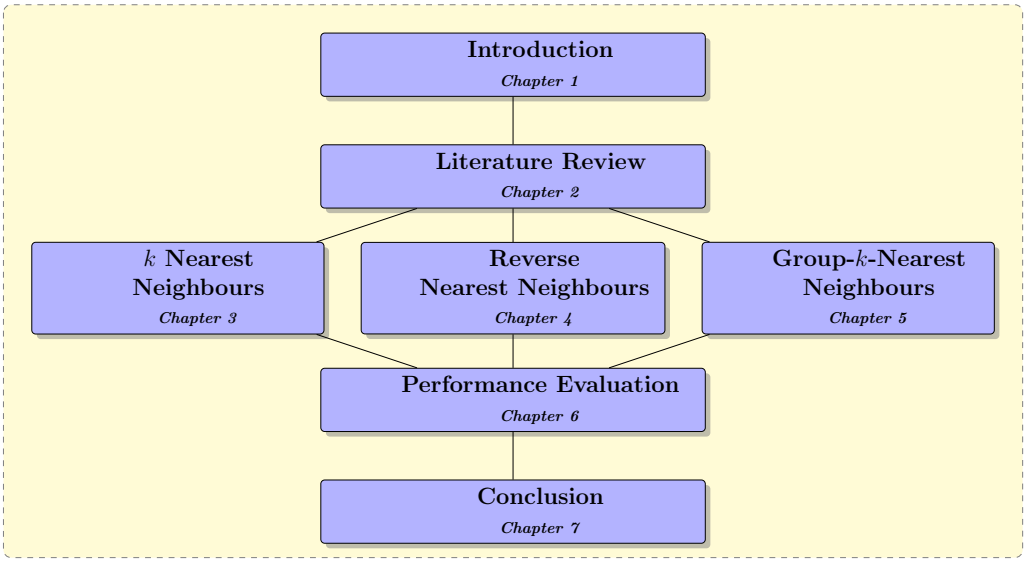


Figure 1.5: Study Organisation

- A survey of the state-of-the-art work is provided in Chapter 2.
- We propose models for each of the three nearest neighbour queries above in Chapter 3, 4 and 5, respectively. More specific descriptions are as below:
  - Two lightweight validation algorithms to filter results from mobile peers and answer kNN queries are presented (Chapter 3).
  - A new direction in mobile P2P query processing in solving bichromatic RNN queries with three different algorithms to search cached spatial data of objects of interest from mobile peers is introduced (Chapter 4).
  - Subsequently, our research on GNN is presented with an energy and time efficient algorithm (Chapter 5)
- The models are tested by a simulated environment (Chapter 6).
- We summarise our study, discoveries, findings and outcomes as well as discuss implications, significance and potential impacts on the big picture (Chapter 7). In addition, we identify limitations and discuss potential research arising from this study in this chapter.

"How puzzling all these changes are! I'm never sure what I'm going to be, from one minute to another."

Lewis Carroll, Alice's Adventures in Wonderland

2

Literature Review

2.1 Overview

This chapter reviews the literature dealing with research into P2P networks. Each section deals with a different issue addressed in this thesis. The first section will provide a common ground in Mobile P2P Networks in term of wireless technologies and collaborative caching techniques. Section 2.3 is a brief survey of the state-of-art mobile P2P query processing. Subsequently, a big picture of Nearest Neighbours will be described with a number of basic spatial indexing structures. A taxonomy of nearest neighbours will be examined in Section 2.5. In Section2.6, existing Mobile P2P Query Processing in Nearest Neighbours context will be discussed. The final section (Section 2.8) will summarise the chapter.

2.2 Network Infrastructure for Mobile P2P Networks

To provide a common understanding of the network infrastructure in Mobile P2P Networks, this section will provide an introduction of wireless technologies and mobile collaborative caching techniques that support this type of networks.

2.2.1 Wireless Technologies

There are many types of wireless technologies deployed on mobile devices. In centralised architectures, the second, third and fourth mobile communication systems (2G, 3G and 4G) have been extensively used. The word 'G' here stands for "generation" in mobile technology, installed on mobile devices and cellular networks.

2G technologies, such as GSM CSD/GPRS, support voice and text messaging services. 3G technologies, including HSDPA, support broadband Internet access with the minimum consistent Internet speeds of 144Kbps. The computational power of mobile devices is double every two years. 4G technologies including HSPA+ 21/42, WiMAX, and LTE introduces new services, for example: online games, music and video players. Each generation offers faster Internet speeds, and then, better services than the last on the same carrier (Fitzek and H., 2007).

However, there are many drawbacks in the centralised approach. The first issue is battery capacity. Even though battery capacity has been almost doubled in the last decade, it is still far from keeping pace with the development of computational power. The operational time of 3G mobile devices is only a half that of 2G mobile devices because of the increasing number of services and hardware (Fitzek and H., 2007). Second, mobile devices today are equipped with large memory and high computational processors. Their high capability paves the way for services not in the centralised networks but from other mobile peers. In other words, services on mobile P2P networks have been great potential.

Table 2.1: IEEE 802.11 comparison

802.11 versions	Freq.(GHz)	Data rate(Mbit/s)	Outdoor range(m)
First in 1997	2.4	1, 2	100
a	5	6 - 54	120
b	2.4	1 - 11	140
g	2.4	7.2 - 150	140
n	2.4, 5	6 - 54	250
ac (DRAFT)	5	up to 866.7	250

In the context of mobile P2P networks, the focus in this study is on short-range technologies including IEEE 802.11, Bluetooth, IEEE 802.15.4/ZigBee and the emerging technology: Wi-Fi Direct.

IEEE802.11

The IEEE standard IEEE802.11 relies on a mac access protocol (MAC) and various physical layers for implementing wireless local networks (WLANs) in the 2.4, 3.6, 5 and 60 GHz frequency band. The first version was released in 1997 with three forms of realisation at the physical layer, including direct sequence spreading (data rate: 1Mbit/s or 2Mbit/s), frequency hopping (1Mbit/s or 2Mbit/s), and infrared (1Mbit/s) in the 2.4GHz band. In 1999, 802.11b was introduced. This version offers data rates of up to 11Mbit/s and also uses the 2.4GHz band. Shortly after that, IEEE802.11a was introduced in 2003 to avoid the crowd on 2.4GHz by working with the 5 GHz band with maximum 54Mbits/s. However, the overall effective range of IEEE802.11a is less than that of IEEE802.11b. IEEE802.11g was introduced in the 2.4GHz band with data rate up to 150Mbit/s. The summary of IEEE802.11 family is presented in Table2.1

IEEE802.11 uses both unicast and broadcast messages. Whereas a unicast message is for communication between two stations, broadcast supports communication originated by one station and received by multiple stations. When a broadcast is dispatched, IEEE802.11 tends to use the minimum data rate. In addition, the opportunistic listening approach, a combination of unicast and multicast messaging, was also introduced. Specifically, when two devices are in communication in normal

Table 2.2: Bluetooth Power Classification

Class	Maximum Power	Operating Range
Class 1	100mW (20dBm)	100 metres
Class 2	2.5mW (4dBm)	10 metres
Class 3	1mW (0dBm)	1 metres

unicast mode, the neighbours are listening to the communication. Therefore, there is always at least one acknowledgement received by the sender if those two devices are surrounded by at least some neighbours.

Bluetooth

Bluetooth is named after the king Harald Bluetooth, who ruled Denmark between 940 and 985 AD (Loo, 2007). Based on a radio technology in the 2.4GHz band, is a common short-range communication supported by the majority of to date smart mobile devices. The IEEE has reviewed and approved a standard for Bluetooth wireless technology, which is IEEE 802.15.1 in 2002 (IEEE802.15.1, 2002). However, this standard is no more maintained. The range of communication is based on the power class of the bluetooth model. In general, Bluetooth technologies are categorised into three classes as shown in Table 2.2.

To communicate with another Bluetooth device in its proximity area, a Bluetooth device first starts service discovery and then categorise the devices found into categories, such as phones, PCs or TVs. The next step is pairing up two Bluetooth devices by shared passwords. Recently, Apple has introduced iOS 7, which supports easy “one touch” setup. Accordingly, users need only touch a device running iOS 7 to a newer Apple set-top box to have it automatically set up Wi-Fi networks, region settings and Apple Store accounts.

IEEE 802.15.4/ZigBee

Like Bluetooth, IEEE 802.15.4 is derived from IEEE 802.15 standard for wireless personal area networks (WPANs) with low data-rate and low power consumption and very low complexity (IEEE802.15.4, 2003). It specifies the physical layer and

MAC layer with little to no underlying infrastructure. Network topologies can be either peer-to-peer or star networks. Peer-to-peer networks can serve as a basic for ad hoc network capable of performing self-management and self-organisation. A star pattern is also supported, where the coordinator of the network will necessarily be the central node.

It operates in an unlicensed, international frequency band. 27 channels, including: 16 channels in the 2.4GHz ISM band, 10 channels in the 915MHz I and one channel in the 868MHz band, are used (Hackmann, 2006).

Being built on the top of physical and MAC layers of IEEE 802.15.4, ZigBee specifies network, security and application layers. It provides Bluetooth-like device and service discovery via broadcast and unicast messages (Hackmann, 2006). In addition, ZigBee support automatically creating and maintaining network links. Depending on power output and environmental characteristics, the transmission range is from 1m to more than 100m, which is comparable to Bluetooth class 1 and IEEE 802.11a,b. Its applications focus on monitoring applications and wireless control (Sturek, 2005; Alliance, n.d.).

Moreover, wake-up delays of ZigBee nodes is less than 30 ms while Bluetooth nodes takes around three seconds to change from sleep mode to active mode. As a result, ZigBee devices can be very responsive. The average power consumption of ZigBee nodes is also low.

Due to the above characteristics, IEEE 802.15.4 is suitable for Peer-To-Peer networks. Hence, the simulation in this thesis is based on this standard. Nevertheless, the algorithms proposed in this thesis can be implemented with any other short-range wireless technologies.

Wi-Fi Direct

A conventional Wi-Fi network requires a wireless access point as a central hub so that all other devices with Wi-Fi capability can be connected. The wireless access

point provides physical support for wireless networking, routing messaging between devices, and administrating the joining or leaving of wireless devices.

Wi-Fi Direct was introduced by Wi-Fi Alliance as a new direction of connecting Wi-Fi devices to transfer content and share applications without any access point (Option®, 2011). In other words, Wi-Fi devices can be connected in an ad-hoc mode provided that a software access point is embedded in the devices. Wi-Fi Direct supports both a one-to-one connection between two devices or a simultaneous group connection among many devices.

When a device goes into the communication range of another Wi-Fi Direct host, they will be connected and set-up information is gathered in a simple way like Bluetooth. The pairing process is similar to Bluetooth technology as it also uses Bluetooth signal. An alternative for pairing up two devices is pressing a physical button on each device subsequently. Data are transferred directly between peers to reduce overhead cost. Another advantage of Wi-Fi Direct is that it can set up Peer-to-Peer communication between devices from different manufacturers as long as at least one of the devices is compliant to Wi-Fi Direct,

To the best of our knowledge, the first implementation of Wi-FI Direct was in Centrino Platform by Intel, 2008. As a result, their notebooks can provide Internet connection to other devices. This became a trend in the Wi-Fi support device market. Google included Wi-Fi Direct in their Android 4.0 Ice Cream Sandwich, which is the platform of Samsung Galaxy Nexus. Digital Living Network Alliance (DLNA) "expects DLNA Certified and Wi-Fi Certified Wi-Fi Direct smart-phones to grow strongly through 2016." (Option®, 2011)

The above advances in wireless communication technologies have paved the way for the developments collaborative caching techniques, which play a major part in Mobile P2P Query Processing. The next section will describe the existing techniques for collaborative caching that have been used in mobile ad-hoc in general and also in mobile P2P networks in particular (Section 2.2.2). The problem of caching in spatial queries will also be discussed in Section 2.2.2.

2.2.2 Mobile Collaborative Caching Techniques

Caching technique was first introduced in 1965 to balance the speed between the processor and main memory (Wilkes, 1965). To date caching has widely been used in wired networks such as file systems and web performance (Dahlin et al., 1994; Wessels and Claffy, 1998; Nekovee and Bogason, 2007). With the fast evolution of wireless communication technologies, such as the Wi-Fi Direct (initially called Wi-Fi P2P), Wireless LAN and Bluetooth, mobile P2P collaborative caching has been drawn increasingly attention as an alternative for information sharing among mobile hosts over standard client-server models. In general, it is a technique to improve data retrieval performance by allowing moving objects to access a local cache of their peers (Chow et al., 2007).

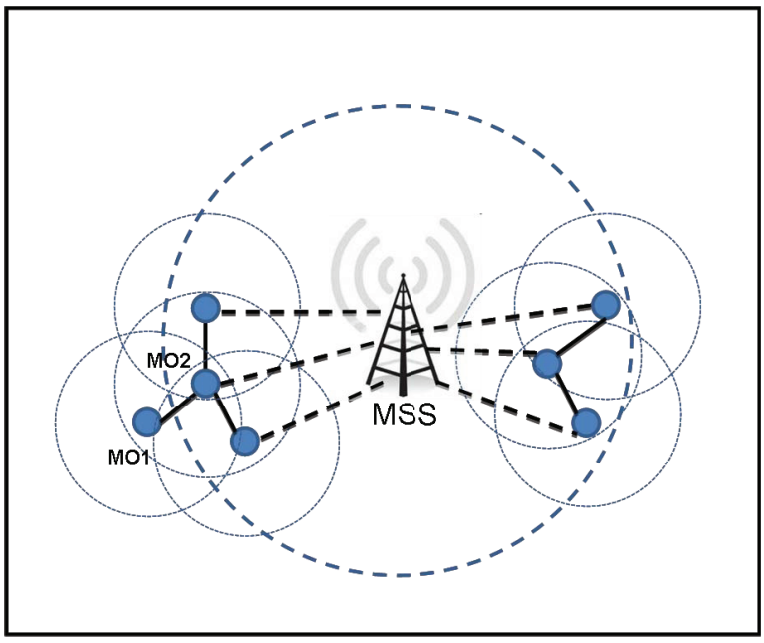


Figure 2.1: COCA architecture. The biggest circle is transmission range of MSS. The other circles are those of moving objects. Dashed lines are connection from moving objects to the MSS while continuous lines are connections between peers. As MO_1 is out of transmission range of MSS, it connects to its peer, MO_2 for requested data.

Mobile Collaborative Caching Framework

An intuitive framework for collaborative caching in mobile P2P is proposed in (Chow et al., 2007) where a moving object communicates with its peers for interested data

if it is unable connect to a mobile support station (MSS), an infrastructure support. Figure 2.1 illustrates the system architecture of COCA in (Chow et al., 2007).

Research on caching usually addresses three primary issues as below:

1. *Information Search* (or *Cache Resolution* or *Cache Discovery*) determines where moving objects should request data.

2. *Cache Management* is divided into three modules: *Cache Admission Control* deciding which data should be cached; *Cache Replacement* dealing with replacing data when the cache is full; and *Cache Consistency* (or *Cache Invalidation*) synchronising cached data with data in data source.

3. *Prefetching* is in charge of selecting which data should be prefetched from the source for future use.

Existing Techniques for Collaborative Mobile Caching

Research in (Yin and Cao, 2006) suggested two different *Information Search* schemes: CacheData and CachePath. In CacheData, forwarding nodes check whether a data item is popular. If yes, it will cache data for future requests. Since this method requires a huge cache space on each forwarding node, CachePath is proposed where a path to the closest cached node is stored instead of the data itself. Although CachePath can overcome the cache space problem, it suffers extra processing overhead from updating paths since moving objects are moving frequently. There are two policies of *Cache Admission Control* in (Yin and Cao, 2006). First, priority of data accessibility is higher than that of query delay. Second, a node will not cache data from its direct neighbour. For *Cache Replacement*, access interest and data size are taken into account in CacheData and CachePath to decide which data should be removed when the cache is full. Like most research, *Cache Consistency* techniques in CacheData and CachePath are based on time-to-live (TTL), which is a value initialised for each data item. After TTL is over, cached data item becomes invalid. Their disadvantages are that the popularity of data could be incorrect since mobile

nodes are dynamic and a node cannot share its cached information if it is not on the forwarding path.

HybridCache in (Cao et al., 2007) combines both methods in (Yin and Cao, 2006) where forwarding nodes decide to cache data or path based on data item size and TTL of each item. The similar *Cache Admission Control* is used. It also proposed a relay P2P cache consistency as a hybrid and flexible invalidation technique.

The Internet-based aggregate schemes in (Lim et al., 2006; Sailhan and Issarny, 2003) search data item by a broadcast based algorithm. Accordingly, a moving object sends a broadcast message to all its neighbour nodes, which are other moving objects or access points, when it cannot locate the requested data item. The same *Cache Resolution* policy as that in (Yin and Cao, 2006) is used. For *Cache Replacement*, it applies an enhanced Least Recently Used (LRU) algorithm based on distances from the requester to access points or other nodes caching the requested data and access frequency. Recall that in the LRU algorithm, the least recently accessed cached data is replaced. In other words, data may be replaced if it has not been accessed for a long time even though data may be needed by other peers later. This is reasonable since peers can join or leave the network at any time.

In the cluster-based scheme of (Denko and Tian, 2006), a strategy for proactive prefetching was proposed. More specifically, the network is divided into clusters. Each cluster consists of a cluster head, a data source, caching agents and a number of mobile hosts. Data which are accessed very frequently or needed are prefetched based on expiry of TTL. As a result, network performance is improved while communication overheads are decreased. In addition, prefetching cost is compensated by future minimisation of access delay and energy cost in (Nuggehalli et al., 2003); however, this cache prefetching was designed only for static peers.

COOP in (Du et al., 2009) focuses only on *Cache Resolution* and *Cache Management*. To discover a data source, it proposes an algorithm using hop-by-hop resolution, historical profile and zone-based resolution. As a result, communication cost is reduced while data availability and access efficiency are improved. COOP

also uses the same cache admission control policies as (Cao et al., 2007; Yin and Cao, 2006). For *Cache Replacement*, like in (Denko and Tian, 2006), COOP uses LRU based on TTL to make decision with the help of GPS. However, flooding leads to extra discovery cost.

However, those studies above focus on item-based caching, which supports only basic queries but not spatial queries, such as range or nearest neighbours. In order to address this problem, a semantic caching technique was proposed in (Dar et al., 1996) that caches items and their query type correspondingly. Each node maintains the semantic description of the query scope and the resulting data objects in its cache. As an extension of this work, the authors in (Ren et al., 2003) proposed a formal semantic caching model with efficient query processing strategies.

Caching techniques for Nearest Neighbour Queries

In the context of Nearest Neighbour queries, the first work on a semantic caching technique was presented in (Zheng and Lee, 2001) by keeping the validity time of the previous query results to help answer the future Nearest Neighbour queries. An enhancement of this work was proposed in (Zhang et al., 2003) by defining validity regions for two common spatial query types, namely, NN and range queries, and developing efficient algorithms for computing validity regions and influence sets.

Proactive Cache in (Hu et al., 2005) and Complementary Cache in (Lee et al., 2006) are other semantic caching schemes that supports a variety of spatial query types. The query performance was improved by answering part of the query locally using the cached objects of interest and the cached indexes of those objects. The cached cached index makes it possible to reuse objects for common types of spatial queries such as Nearest Neighbour or Range queries in mobile client-server environments. Nevertheless, these cache techniques are only semantic caching that does not harness the collaboration of peers in answering the queries.

To the best of my knowledge, the first research in collaborative caching technique for mobile P2P networks is Structure-Embedded Collaborative Caching (SECC) (Zhu

et al., 2011) whereas, other caching techniques only were designed mobile client-sever networks. There were three phases in SECC:

- Local Processing: trying to solve the query locally by evaluating local cache of the query peer.
- Collaborative Processing: If the query cannot be solved by phase 1, the query peers will ask for help from its peers and then continue forwarding the query to en-route nodes on the path to the server until the answer is acquired or the query reach the server.
- Server Processing:The server provides the final hope to answer the query if none of the above stages can.
- Cache Replacement: After finishing processing the query, the query peer will cache the result objects and supporting indexes collected from the peers and the server.

Although SECC proposes a mechanism for collaborative caching, it still requires a base station to back up and complete the query. Moreover, each peer needs to maintain an index tree and exchange the tree with their neighbours, which involves significant cost in periodical or adaptive update.

Discussion

Table 2.3 shows a summary of the existing mobile collaborative caching techniques. Although they match their own design goals and application scenarios, most of them work on unstructured data items and do not provide any solution for spatial queries that require more specific techniques. Another limitation of all techniques above is that all existing cache strategies rely on the mobile support station which is the weak and single point of failure in the whole network. In addition, employing a centralised server to deliver information to peers increases the time elapsed from the moment a user requests a data item until that item is received due to the multi-hop nature of

Table 2.3: Existing Techniques for Collaborative Mobile Caching

Techniques	Info. Search	Cache Manage.	Prefetching
CacheData	✓	✓	
table CachePath	✓	✓	
HybridCache	✓	✓	
Internet-based Aggregate Schemes	✓	✓	
Cluster-based Schemes			✓
COOP	✓	✓	
space SECC	✓	✓	
space			

information transfer. Moreover, the server workload and the network traffic to the server go up with the increase in the number of requests.

Based on the network infrastructure discussed above, mobile devices to date not only use services but also provide services such as location-based query processing, which will be addressed in the next section.

2.3 Mobile P2P Query Processing

This section will first discuss the state-of-art of Mobile P2P Query Processing in general. Subsequently, the discussion will be narrowed down to Mobile P2P Query Processing in the context of Spatial Queries and, more specifically, Nearest Neighbours.

Advances in collaborative caching techniques for mobile technology (both hardware and software) have enabled mobile P2P systems to become more and more popular. The wide range of applications goes from real-life applications to natural disaster management.

2.3.1 Classification

In general, there are two main directions to answer queries in mobile P2P querying systems. These directions are *Report Pushing* (or *Periodical Broadcast*) and *Report Pulling* (Park and Valduriez, 2011; Waluyo et al., 2003, 2005; Visvanathan et al.,

2005; Park and Valduriez, 2011; Kulik et al., 2002; Pucha et al., 2004; Padhariya et al., 2011; Xu et al., 2009).

Report Pushing

In this direction, reports are broadcast periodically. The query node listens to broadcast channel to get answers (Park and Valduriez, 2011). There are two sub-directions in this main direction. Those are: stateful methods and stateless methods. Stateful methods usually rely on a network topology and maintain states of data dissemination while stateless methods do not.

Topologies in stateful approaches can be a tree constructed by peers as in (Huang and Garcia-Molina, 2004). Alternatively, peers can also be organised as clusters where each cluster consists of a specific number of peers and one selected cluster head (Visvanathan et al., 2005). Through localisation systems such as GPS, location-based methods (for example: Greedy Forwarding, GPSR, GAF, etc.) are also used in stateful paradigm (Karp and Kung, 2000; Mauve et al., 2001; Xu et al., 2001). However, when network topology is dynamic, stateful direction is not a wise choice.

The Energy Efficient Data Access in (Park and Valduriez, 2011) is an example of flooding-based stateless method. This research claimed that periodical broadcast is economical in communication cost and easier to scale up than server-client based, on-demand methods. It also uses indexing to identify the content of data and to forecast when data will arrive. MOBI-DIK in (Luo et al., 2008) is also a stateless approach. However, unlike a flooding method in (Park and Valduriez, 2011), MOBI-DIK is a store-and-forward method in which each peer ranks reports and decides which reports should be stored according to its cache memory, power and network bandwidth. Research in (Xu et al., 2009; Wolfson et al., 2006) stated that there are two basic elements in the rank-based store-and-forward, which are: supply (number of peers caching a report) and demand (number of peers querying report). The higher the supply is, the lower the rank is (or expected utility) and vice versa, the

higher the demand, the higher the rank. Moreover, gossiping and negotiation-based methods are also introduced respectively in (Datta et al., 2004; Kulik et al., 2002).

Report Pulling

Report pulling (or on-demand) direction is a reactive approach in which a peer invokes a query and the query is transmitted in the network. Traditionally, query is flooded or broadcast into the whole network to find resources or peers that can give an answer (Pucha et al., 2004). However, simple flooding is an unnecessarily expensive communication method. In response to this problem, an opportunistic approach is introduced in (Xu and Wolfson, 2005) for resource availability consideration in which each moving object will send a report it caches to the peer in its communication range and also asks for a new report in exchange. Some research also proposed a virtual currency based economic model to encourage peers to participate in exchanging cached data (Xu and Wolfson, 2005). In economic models, peers can be producers, consumers and brokers. For example, a car, as a consumer, needs to find an unoccupied parking lot. A broker can provide information. On the other hand, a car, as a producer, can aggregate its cached information and sell it to a broker.

The main existing issue in methods discussed earlier is that spatial query processing such as kNN or range search is not fully addressed. Most of them are working with regular queries; therefore, their data structure and query processing are not suitable for multi-dimensional queries. The state-of-art of Mobile P2P Query Processing in Nearest Neighbours Context will be discussed in details in Section 2.6.

2.3.2 Discussion

The summary of current research in both directions (Report Pulling and Report Pushing) is shown in Table 2.4. Overall, our work differs from all of the previous research as it is the first one to investigate Nearest Neighbour queries in mobile P2P networks only by harnessing the power of peer collaboration without any central supervision.

Table 2.4: Existing Mobile Query Processing Techniques

Classification	Brief Description	Examples
Report Pushing	*Stateful methods* rely on a topology network	tree-based
	Stateless methods are not based on a topology network	flooding-based store-and forward goshipping and negotiation-based
Report Pulling	on-demand/reactive approaches	flooding-based virtual currency-based tree-based mobile agent-based

One of the applications of Mobile P2P Query Processing is solving location-based queries which is the main topic in the thesis. The next section will briefly introduce Nearest Neighbour Queries with a number of basic indexing structures.

2.4 Nearest Neighbour Queries At a Glance

This section is to give you an overview of Nearest Neighbours, which are the main subject of this study. Subsequently, R-Tree and its variants will be presented as they are basic indexing structures that most traditional Nearest Neighbour query processing algorithms are based on.

The applications of Nearest Neighbours vastly spread in many disciplines including image processing, machine learning, pattern recognition and especially spatial database. For example, in image processing, Nearest Neighbour problem finds the similarity of different digital images which are presented by a string of bits. As an example in spatial database, Nearest Neighbours is an optimisation problem for searching two closest spatial objects for a certain metric such as Euclidean metric or spatial network distance.

The traditional approaches to solve Nearest Neighbour problems are based on indexing structures such as R-Tree or its variations. Hence, the next section will

discuss a number of important spatial indexes in spatial database in order to give an essential background on Nearest Neighbour query processing.

Some basic spatial indexes

The original index structure, R-Tree, was proposed by Guttman in (Guttman, 1984). R-Trees are data-partitioning multidimensional index structures using the bounding rectangles as the index type. In general, objects in R-Trees are represented by tuples. Each tuple has a unique identifier for retrieving (Fig. 2.2a). Leaf nodes contain indexed record entries of the form (I, tuple-identifier) where I is a d-dimensional rectangle, or the so called Minimum Bounding Rectangle (MBR) as in Fig. 2.2b.

The most common nearest neighbour search method on R-Trees is a branch-and-bound search algorithm as described in (Cheung and Fu, 1998). This is a depth-first branch and bound search that recursively visits the nodes until reaching a leaf node. Then the algorithm performs back-tracking to upper levels. Only necessary nodes will be visited in this backtrack procedure. However, depth-first search algorithm is not optimal since it still has to visit unnecessary nodes. In response to the optimal problem, a best-first search also was proposed in (Hjaltason and Samet, 1999) based on a priority queue of sorted visited nodes in an ascending order of minimum distance from nodes to q. The first entry of the queue is recursively de-queued. If it is a leaf, the entry becomes the answer. Otherwise, it will be examined later.

R-Tree is one of the most popular index structures. Based on the idea of R-Tree, there are many other variants. For example, R^+-Tree maintains single path traversal to the search point by eliminating the overlapping issue of MBR (Sellis et al., 1987). R^*-Tree significantly improves search performance by introducing the concept of forced re-insertion (Beckmann et al., 1990). Another variant of R-Tree is Hilbert R-Tree which takes spatial object ordering into consideration (Kamel and Faloutsos, 1994). In particular, data are clustered according to their ordered value; which is beneficial for query performance and space utilisation.

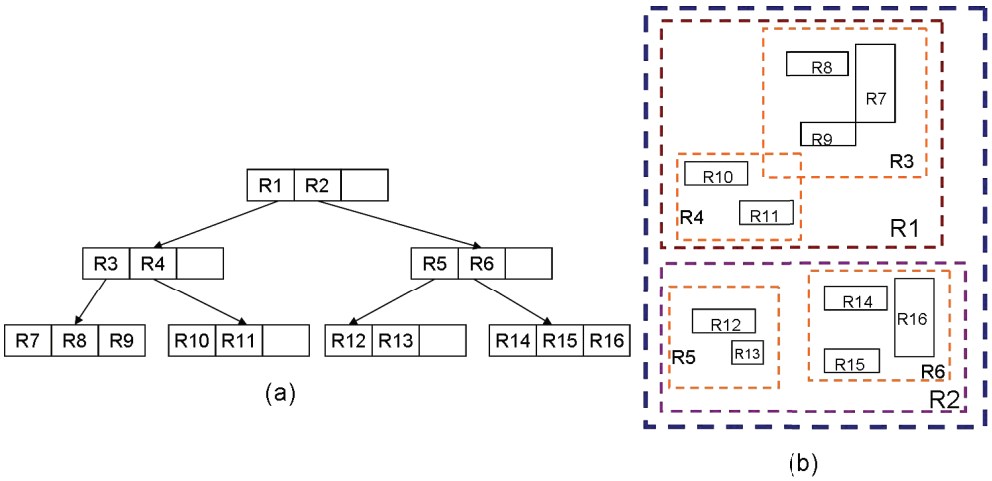


Figure 2.2: (a)R-Tree structure. (b)MBRs

Although indexing structures are commonly used to speed up data searching in database, they are not the focus of this study. The main reason is that in the mobile P2P networks, mobile peers are bounded by energy and memory constraints. Therefore index structures are not appropriate. In addition, there is no base station in the networks for centrally processing spatial queries. As a result, query processing algorithms are our primary targets to improve search performance.

The following discussion is on the taxonomy of Nearest Neighbours. In the scope of this study, the query processing is the focus rather than indexing structures. In addition, the state-of-art of three selected nearest neighbour queries, which are k nearest neighbours, reversed nearest neighbours and lastly group k nearest neighbour, will be elaborated in great detail.

2.5 A Taxonomy for Nearest Neighbour Queries

Nearest neighbour queries in spatial database are classified from various perspectives: spatial perspective query object perspectives: space perspective, objects' movement perspective, query object perspective and relationship perspective (Taniar and Rahayu, 2013). Fig. 2.3 is an illustration of the taxonomy for Nearest Neighbour Queries. The details are going to be discussed further as follow:

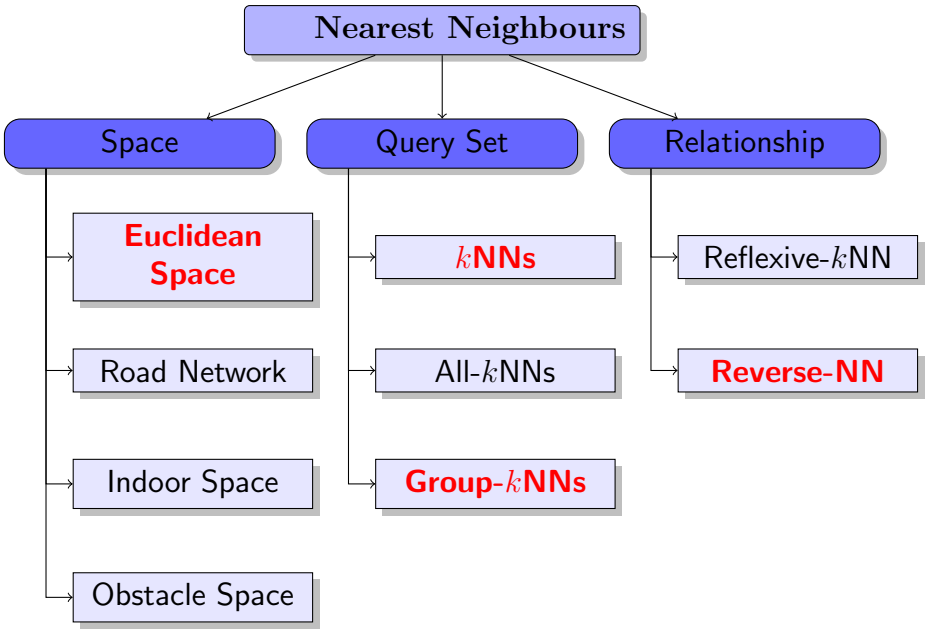


Figure 2.3: A Taxonomy for Nearest Neighbour Queries

2.5.1 Spatial Perspective

Two common metrics are used to measure distance between two spatial objects are: Euclidean space and network space. However, new trends in location-based service have recently targeted on new travel spaces including indoor space and obstacles space.

Euclidean space

The well-known definition of Euclidean distance is straightforward. It is the length of a direct and straight line from one object to another. Here each object is considered as a point in the space. Formally, in a 2-dimensional space, if the coordinates of 2 objects are $O_1(x_1, y_1)$ and $O_2(x_2, y_2)$, the Euclidean distance from O_1 to O_2 is calculated as follow:

$$dis_E(O_1, O_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{2.1}$$

The Fig. 2.4 is an illustration of Euclidean distance in 2-d dimensional space.

The problem can be generalised into d-dimensional space, in which the calculation of Euclidean distance between two objects $O_1(x_1, x_2, x_d)$ and $O_2(x_1, x_2, x_d)$

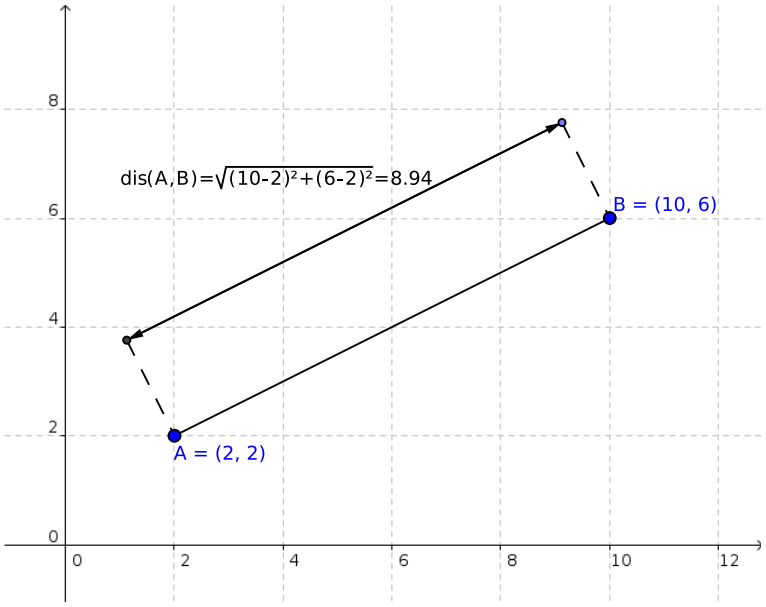


Figure 2.4: Euclidean distance between A and B.

Search algorithms based on Euclidean-distance have been widely applied due to its simplicity, intuition and usefulness in most scenarios such as finding nearest neighbours on air, on walking tracks, and estimating the distance to closest neighbours in any other spaces. Most existing nearest neighbour search algorithms are based on an index tree such as an R-Tree or its variants. Some algorithms such as depth-first branch and bound and best-first search are described in the Section 2.4.

However R-Tree indexing structure or its variants require much memory space and time to store and build up the tree for mobile P2P query processing. In addition, that mobile peers move frequently involves significant update cost to maintain the indexing structure. All algorithms from the study in this thesis are in Euclidean space(Nghiem, Waluyo and Taniar, 2013; Nghiem, Maulana, Waluyo, Green and Taniar, 2013; Nghiem, Green and Taniar, 2013), which rely on caching techniques and P2P mechanism to process mobile query processing. These algorithms will pave the way not only to expand the study in solving other types of spatial queries but also to apply the study in other spaces such as road network space or indoor space in future.

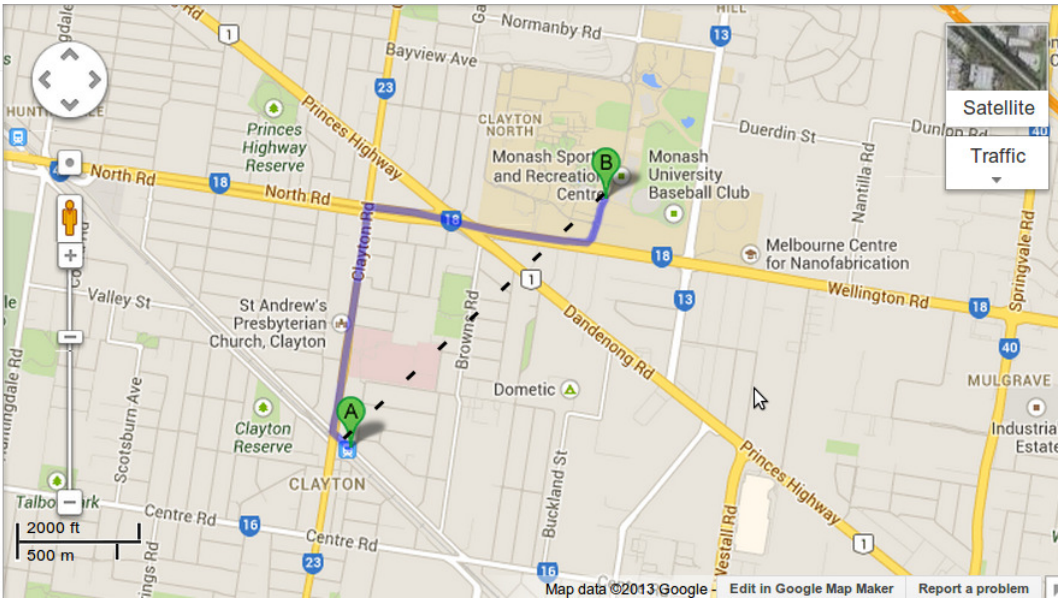


Figure 2.5: The road network distance between Monash University, Clayton Campus to Clayton Station, Clayton, Australia is 2.6km according to the highlighted route whereas the corresponding Euclidean distance, represented by the dashed line, is about 1.5km.

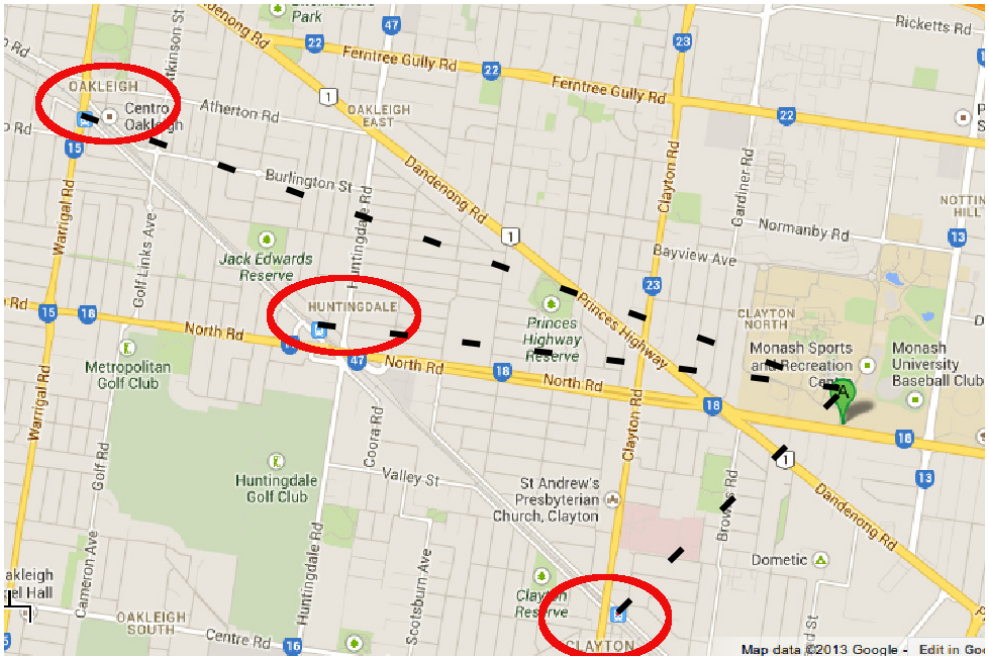


Figure 2.6: A student is standing at Monash University, Clayton campus. She is looking for the nearest train station to go to the city. There are three candidate train stations, which are Clayton, Huntingdale and Oakleigh stations.

Spatial Road Network Space

The nearest neighbour problem can be put in the context of a spatial road network where objects are restricted to move on pre-defined paths (e.g.,roads) that are

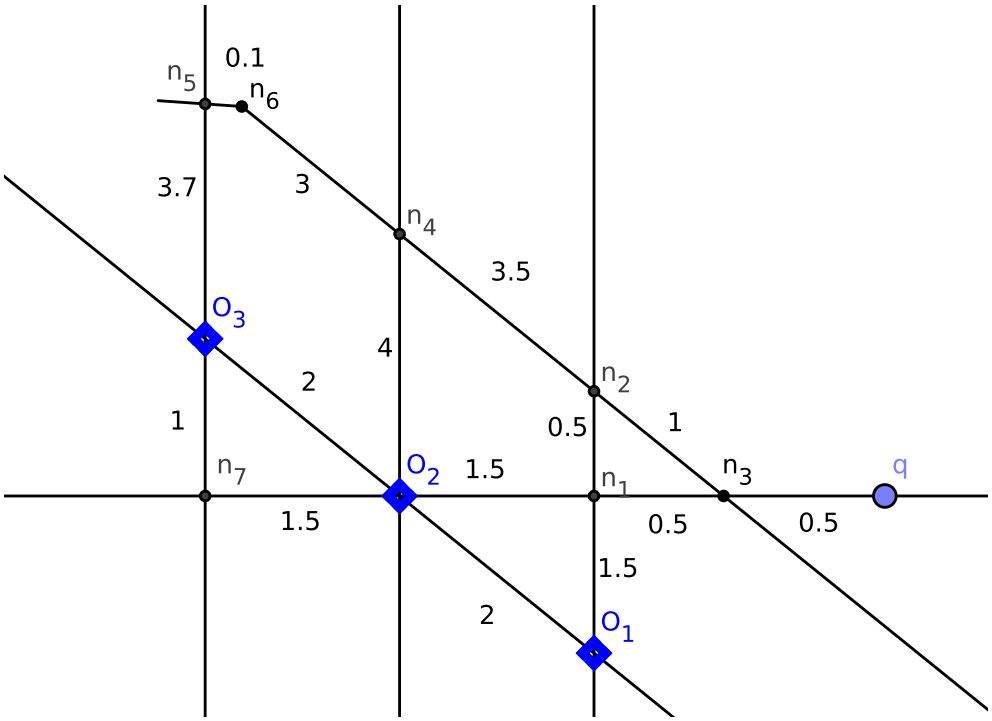


Figure 2.7: A representing graph for the spatial network in Fig. 2.6. q is the student (or the query object in this context), $O_i, i = 1, 2, 3$ are three train stations (or objects of interest. $n_j, j = 1, 2, .., 7$ are intersection nodes.

specified by an underlying network. The distance metric is defined as network distance or the cost of shortest path between two objects. An example of road network distance is illustrated in Fig. 2.5.

Spatial road networks can be formally represented by a graph, in which each vertex is an object and each edge is a road or a link from one object to another. Each edge is assigned a weight that is network distance. For example a student is standing at Monash University, Clayton campus, she is looking for the nearest train station to go to the city. The search algorithms will find the train station to which the cost of the path from the student's current location (at Monash University) is the minimum. The Fig. 2.7 illustrates how the geographic map in Fig. 2.6 is represented formally.

The authors in (Papadias et al., 2003) proposed two basic query processing for nearest neighbour queries on spatial road networks: (i) Incremental Euclidean Restriction (IER) and (ii) Incremental Network Expansion (INE). IER is described as below. First, IER searches for the nearest neighbour o_i in Euclidean space based on Euclidean distance $d_E(q, o_i)$. The search algorithms can be any of the algorithms

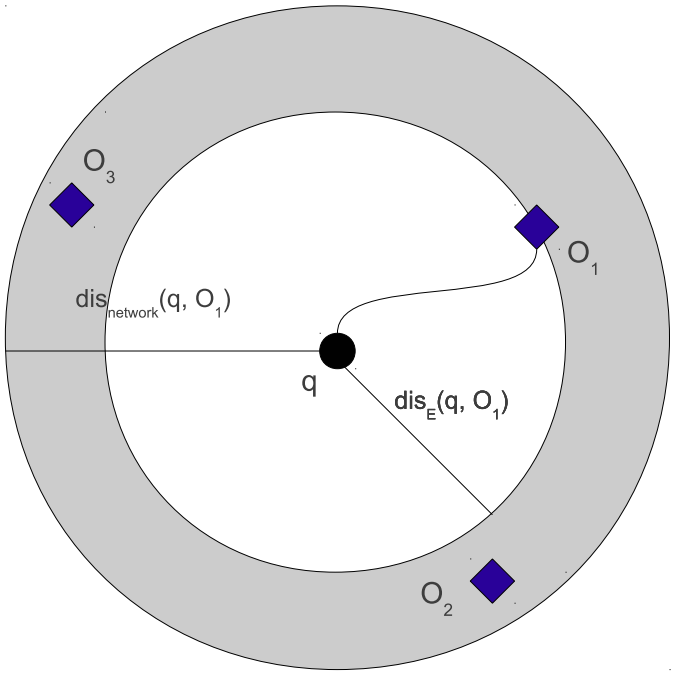


Figure 2.8: Incremental Euclidean Restriction. If O_1 is the nearest neighbour, $d_E(q, O_i) \leq d_{network}(q, O_i) \leq d_{network}(q, O_1)$. The nearest neighbour of q must lie in the shaded area.

discussed in Section 2.4. Then the network distance from q to o_i, $dis_{network}(q, o_i)$ is computed by applying a shortest path search algorithm such as the Dijkstra's algorithm (Dijkstra, 1959). Due to the Euclidean lower bound property, the potential nearest neighbour object o_j, closer (to q) than o_i in the network, should be within Euclidean distance $d_{Emax} = dis_{network}(q, o_i)$ from q. In other words, the nearest neighbour of q falls in the shaded area of Fig. 2.8. The top-down algorithm starts by searching in a large region, surrounded by the upper boundary (the circle $C(q, dis_{Emax})$, and then shrinks to a smaller region, until it finds the nearest neighbour. The extension to kNNs where $k > 1$ was also proposed in a straightforward way (Papadias et al., 2003).

Unlike IER, INE is a bottom-up search algorithm that starts from the query objects and expands en-route nodes in the graph until the nearest neighbour is found. From one node, the next adjacent node to be expanded is the node from which the network distance to q is the smallest. For the illustration in Fig. 2.7, the closest intersection node n_3 to q is the first to be expanded. From n_3, INE finds 2 other intersection nodes n_1 and n_2. Then n_1 and n_2 with their distance to q

are inserted into a queue $Q =< (n_1, 1), (n_2, 1.5) >$ sorted by the network distance to q. The first node n_1 in Q is de-queued and expanded. O_1 and O_2 are found. Since $dis_{network}(q, O_1) < dis_{network}(q, O_2)$, the algorithm returns O_1 as the nearest neighbour. INE performs better than IER if the ranking of the data points by their Euclidean distance are not similar to that with respect to the network distance.This is because in this case IER needs to inspect a large number of nearest neighbour objects in term of Euclidean distance.

Although IER and INE can support exact kNN queries on spatial network databases efficiently, their performance is poor for networks in which the density of objects of interest is low. In addressing this issue, many search algorithms using Network Voronoi diagrams were proposed to handle nearest neighbour queries in spatial road network space (Kolahdouzan and Shahabi, 2004; Safar et al., 2009; Zhao, Xuan, Rahayu, Taniar, Safar, Gavrilova and Srinivasan, 2009; Zhao, Xuan, Taniar, Safar, Gavrilova and Srinivasan, 2009). In network Voronoi diagrams, the network is partitioned into smaller cells. Each cell is centered on an object of interest. (e.g., a restaurant) and contains the nodes that are closest to that object in network distance). The network distances between all the edges (or border points) of the cell to its centre are pre-computed. We also pre-compute distances only across the border points of the adjacent cells.

To find the k nearest-neighbours of a query object q, it is easily done by locating the Voronoi cell that contains q. The centre of that cell is the nearest neighbour as depicted in Fig.2.9. The Network Voronoi Diagram is also extended to kNN, $k > 1$ by expand to the next level of adjacent Voronoi polygons until k objects are retrieved using the precomputed distances. Details of the Network Voronoi Diagram properties and k query processing using the Network Voronoi Diagram were described in (Kolahdouzan and Shahabi, 2004).

As stated before, the study in this thesis is in the Euclidean space where the movement of objects are not restricted in a spatial road network. In addition, constructing and maintaining Voronoi diagrams is involved much overhead cost and

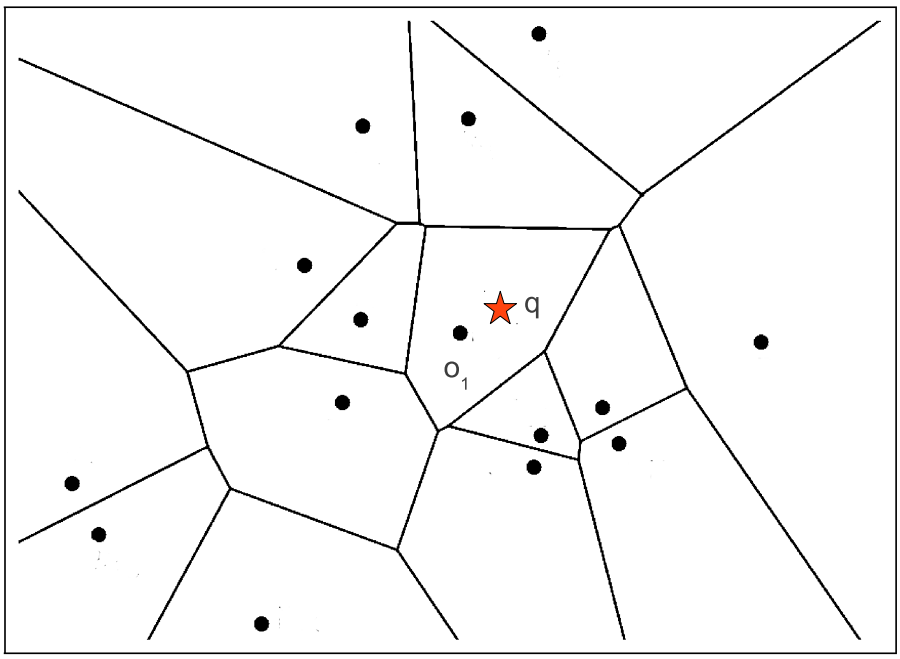


Figure 2.9: An illustration of a Network Voronoi Diagram. The star q is the query object. $o1$ in the cell containing q is the nearest neighbour.

resources especially in a environment that mobile peers move frequently. Nevertheless, the reversed nearest neighbour which will be discussed in Section 2.5.3 of this chapter and also more detailed in Chapter 4 is inspired by the idea of Voronoi diagram in filtering peers in Mobile Peer-to-Peer Network.

Indoor Space

Since people spend most of their lives indoor for working, shopping, entertaining, etc., indoor space has become a growing interest in location-based services. There are many challenges in indoor environments compared to outdoor or open space. First, the indoor systems can be large-scale and complex. For example there are 268 stations with more than 4 million passengers in London's subway network. Second, global navigation satellite systems such as generic GPS are not suitable for indoor space exact velocities or exact positions. Instead, indoor positioning technologies such as RFID, Assisted Bluetooth and Wi-Fi are required. Third, indoor space takes many entities such as rooms, doors and hallways into consideration as constraints, which

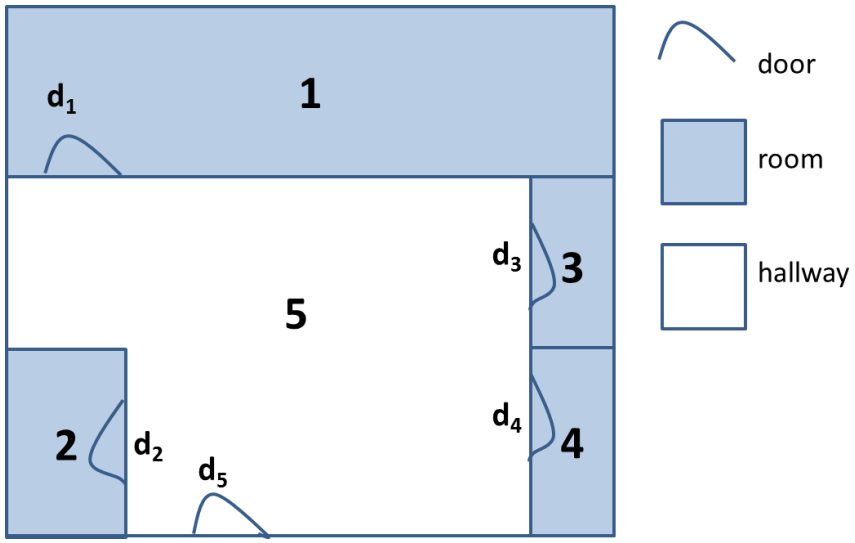


Figure 2.10: An illustration of an indoor space.

render algorithms in Euclidean distance and spatial road network space incorrect. Fig. 2.10 is an example of an indoor space.

Applications of Nearest Neighbours queries in Indoor space plays an important part in location-ware services such as monitoring trains in subway systems or advertising new restaurants to nearest shoppers in a shopping centre.

An indoor space is usually partitioned into rooms, staircases, or hallways. They are connected by doors or staircase entrances. Hallways and staircases could be considered as rooms for simplicity. The doors graph (Yang et al., 2010) has been proposed to represent the connectivity of indoor partitions as well as door-to-door distance as in Fig. 2.11.

Formally, the doors graph is a weighted graph $G =< D, E >$, where:

- D is the set of vertices, each corresponding to a door.
- E is the set of edges. An edge $< d_i, d_j >$ exists if these two doors are associated with a same partition.
- Each edge $< d_i, dj >$ has a weight that is the distance from door d_i to door d_j through their common partition.

The doors graph can be integrated in an indexing structure (Xie et al., 2013). The distance from a query node q to p is the length of the shortest path from q to p

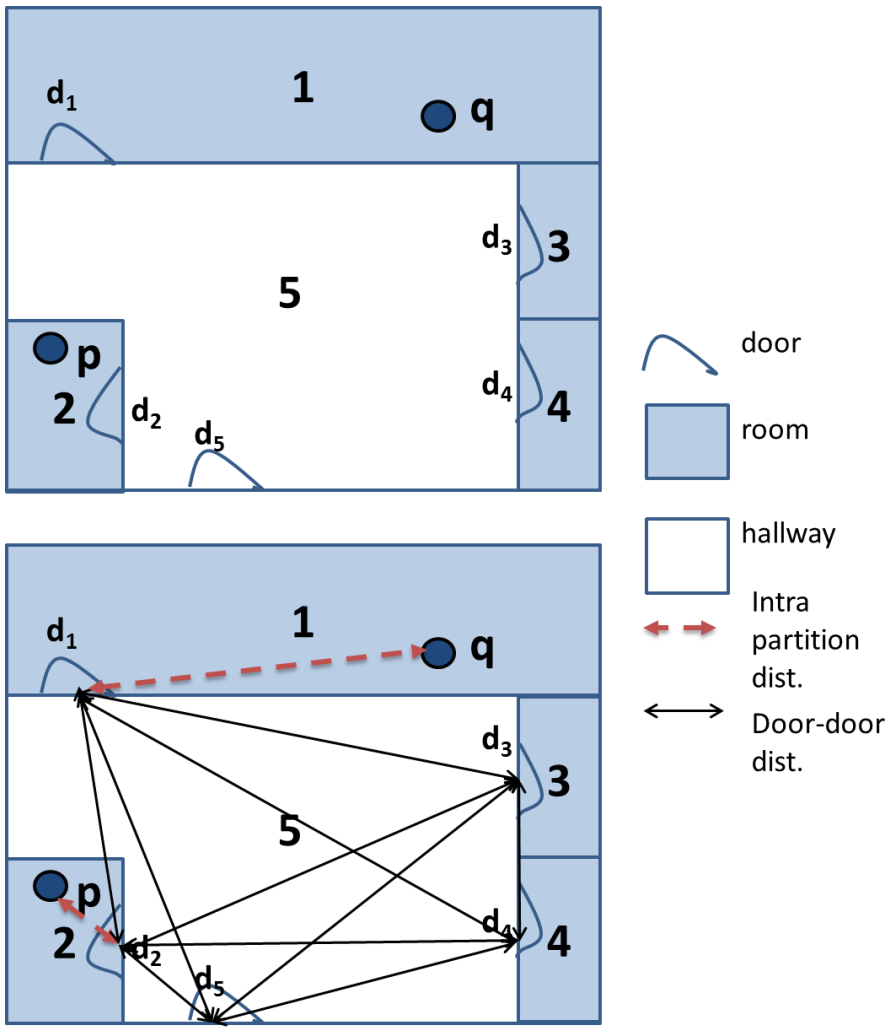


Figure 2.11: A door graph in indoor space. The distance from q to p is calculated by $dis_I(q,p) = dis_E(q,d_1) + dis_I(d_1,d_2) + dis_E(d_2,p)$.

through a number of doors. Formally, it is defined as below:

$$dis_I(q,p) = dis_E(q,d_q) + dis_I(d_q,d_p) + dis_E(d_p,p) \tag{2.2}$$

where $dis_E(q,d_q)$ and $dis_E(p,d_p)$ is intra-partition object-door distances. $dis_I(d_q,d_p)$ is door-door distance, which can be pre-computed (Zhang, Papadias, Mouratidis and Zhu, 2004; Li and Lee, 2008) or calculated on the fly (Xie et al., 2013). For the example in Fig. 2.11, the distance from q to p is calculated by $dis_I(q,p) = dis_E(q,d_1) + dis_I(d_1,d_2) + dis_E(d_2,p)$

Another approach to solve spatial queries is based on connectivity between cells (either rooms or hallways) (Alamri et al., 2012, 2013). The connection between the

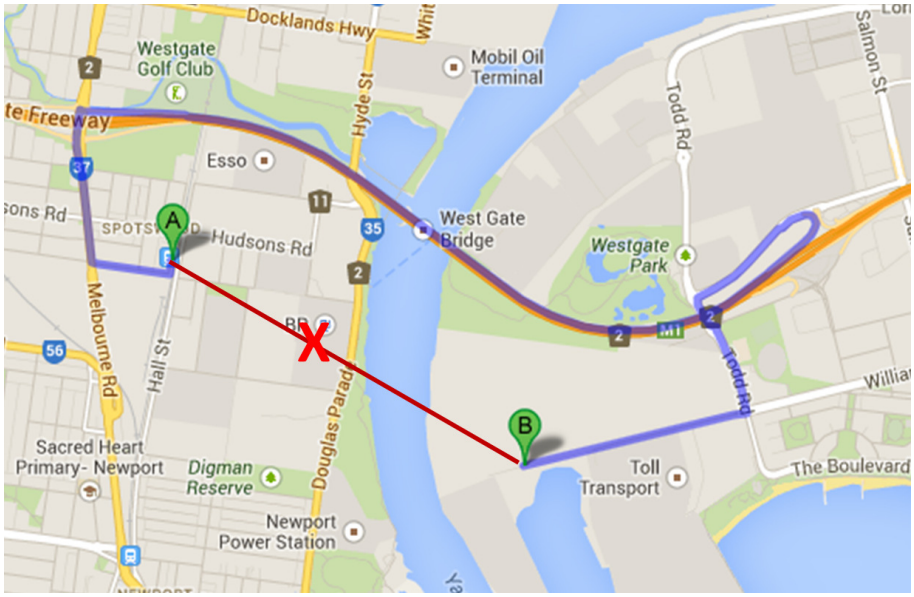


Figure 2.12: An example of obstacle space. To travel from A to B, the query object need to go through West Gate Bridge to cross Yarra river.

cells will be stored in a neighbours distance lookup table of pre-computed neighbours distances between cells. The distance here is the number of hops. For example, if the distance is 0, two cells overlaps; 1, two cells are first level neighbours, 2, the second level neighbour and so on. A neighbours distance lookup table is established to assist in building the expansion algorithm.

Obstacle Space

Nearest neighbour queries can be extended to implement them in obstacle space. Here, Euclidean space takes obstacles into consideration (Xia et al., 2004; Zhang, Papadias, Mouratidis and Zhu, 2004). The path from q to p may be obstructed by one or many obstacles or barrier objects, which renders Euclidean distance become invalid. For example, the distance from A to N cannot be measured by the Euclidean distance as the Yarra river is in the middle.

As in spatial road network space, paths in obstacle space are restricted by underlying roads. The main difference is in the spatial road network space: a viable path needs to follow roads. However, in obstacle space, the path is more flexible as long as it returns the shortest distance by avoiding the obstacle.

Obstacle kNN query processing (OkNN) firstly constructs a visibility graph $V =< E, O >$ .E is a set of all possible paths from q to any objects of interest (e.g. o_1 , o_2). O represent a set of all obstacles, objects of interest and the query point q. A simple network expansion algorithm or shortest path algorithm can be applied to answer OkNN in obstacle space.

2.5.2 Query Set Perspective

A query can be sent from a single object or multiple objects. Traditionally, there is only one query object, which sends out a query to the base station and receives the answer of its nearest neighbours. Actually, there are nearest neighbour queries that are generated from a set of objects. The classification is described as below.

Single Query Object, k Nearest Neighbours

Most nearest neighbour queries are generated from a single query object. For example, a user at a specific location invokes a query q to look for the k nearest seafood restaurants. If k is 1, the problem becomes Nearest Neighbour.

The problem of finding k nearest neighbours (kNN) is identifying k objects of interest from a set of all objects of interest within the network that are nearest to a given query location q according to a specific distance metric, such as Euclidean or network distance metric. For instance, a traveller needs to find two nearest hotels.

Most recent centralised spatial query processing systems are based on R-Tree or its variants to index spatial data of interest objects and improve query performance (Dittrich et al., 2009; Guttman, 1984; Saltenis et al., 2000; Ghadiri et al., 2011). Both depth-first and best-first approaches mentioned in Section2.4 can be extended to k nearest neighbour (kNN) search.

All-kNNs

This type of queries has been discussed in (Clarkson, 1983; Vaidya, 1989; Zhang, Mamoulis, Papadias and Tao, 2004; Chen and Patel, 2007). Accordingly, all of

objects in the spatial network are query objects. The query returns k nearest neighbours to all objects (Chen and Patel, 2007; Zhang, Papadias, Mouratidis and Zhu, 2004). Fig. 2.13 and Fig. 2.14 are two example of All-kNN when k is 1 and 2, respectively. There are a number of applications of All-kNN queries in analysing large multi-dimensional datasets. For example, All-kNN is used in a variety of clustering algorithms (Johnson, 1967; Jarvis and Patrick, 1973; Jain et al., 1999; Bohm and Krebs, 2003). Other applications include co-location pattern mining (Yoo et al., 2005), graph-based computational learning (Koenig and Smirnov, 1996), pattern recognition and classification (Nock et al., 2003) and N-body simulations (Pallavicini et al., 1998). A simple spatial query of this type could be "find all nearest seafood restaurants from each train station".

All-kNN is a computationally expensive operation ($O(kn^2)$) in the worst case). When the size of datasets are growing rapidly, the computation overhead of All$k$NN becomes significant. In addressing this issue, many AkNN algorithms (Corral et al., 2000; Bohm and Krebs, 2003; Corral et al., 2004; Zhang, Mamoulis, Papadias and Tao, 2004) have been proposed. These algorithms are based on an R*-tree indices and a traversing method. During the index traversal, these methods keep track of nodes in the index that need to be considered, and employ a priority queue to determine the order of the index traversal.

Group Nearest Neighbours

This is one of the subjects in our study.

The authors of (Papadias et al., 2004) were the first to introduce the concept of GNNs as illustrated in Fig 2.15. Three query processing methods were developed for this type of queries: multiple query method (MQM), single point method (SPM) and minimum bound method (MBM), which minimises the total distance from a group of query objects to an object of interest. The idea of MQM is to find the nearest neighbour at every query object and calculate the distance to other query objects, which may incur multiple access to the same query object. To avoid this problem,

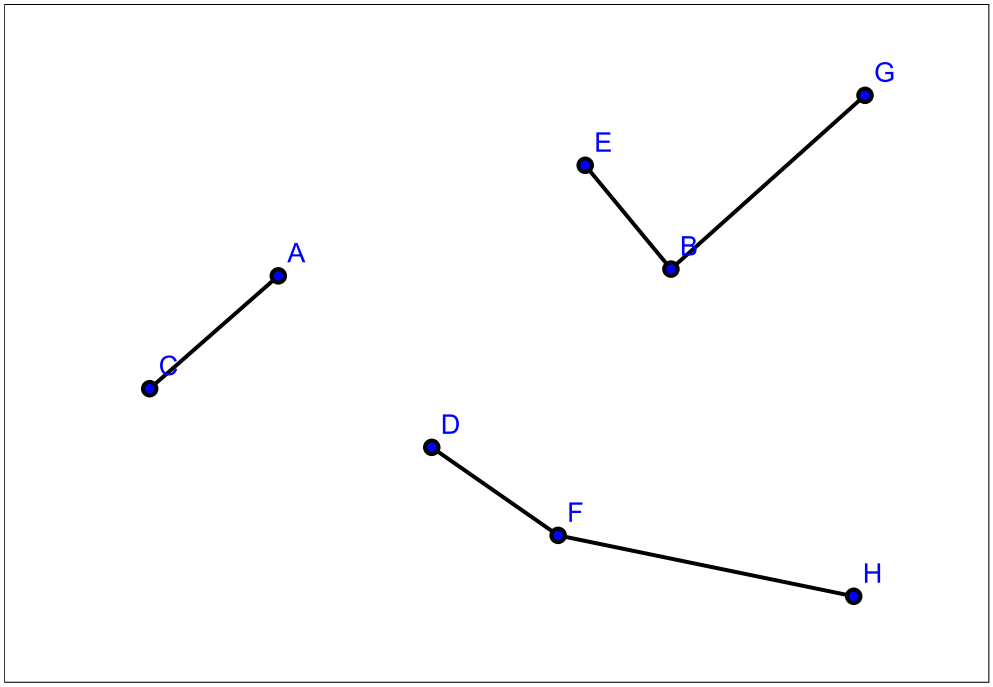


Figure 2.13: An illustration of an All-1NN query. A line is drawn between two objects if at least one object is a nearest neighbour of another.

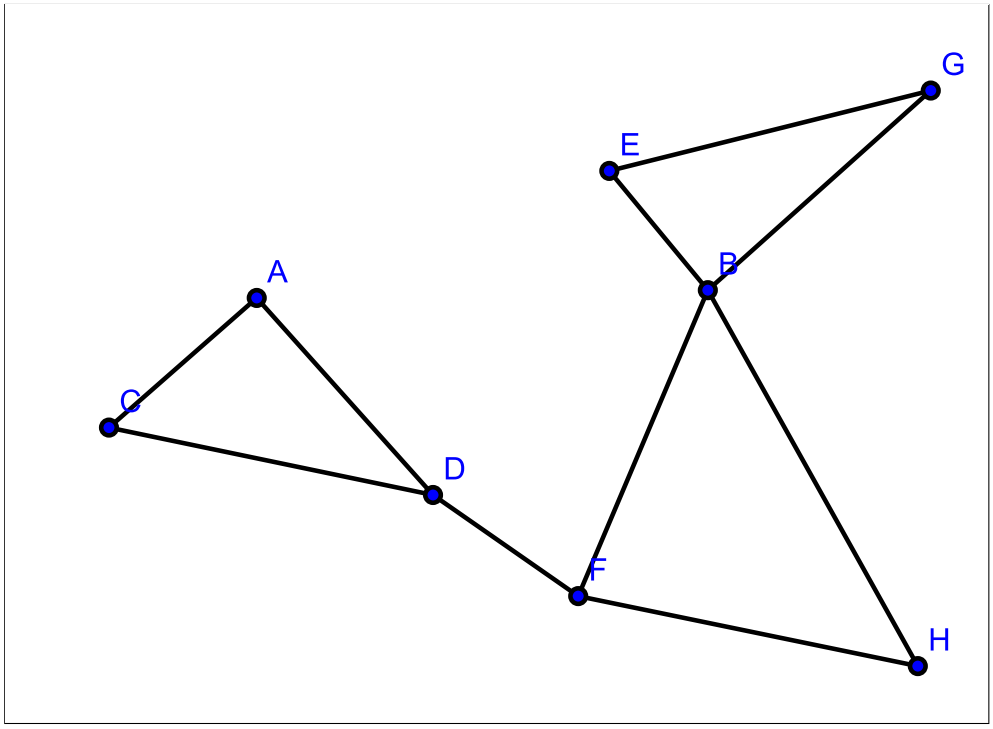


Figure 2.14: An illustration of a All-2NN query. A line is drawn between two objects if one object is one of two nearest neighbours of another.

both SMP and MBM focus on a single traversal to solve GNN queries. Instead of using a centroid point, as SMP does, MBM uses a bounding box that covers all query objects. This work was extended GNN problem to deal with aggregate nearest

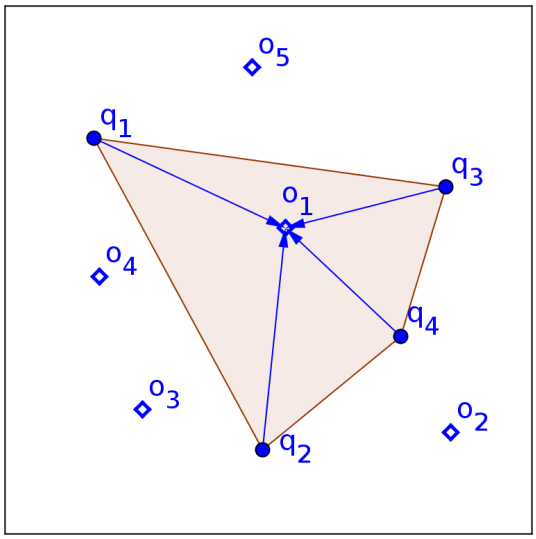


Figure 2.15: An illustration of a GkNN query, when k = 4. $Q = \{q_1, q_2, q_3, q_4\}$ is a group of query points. o_1 is the G4NN since $\sum_{i=1}^{3} distance(q_i, o_1)$ is the minimum compared to other objects of interest (rhombus symbols).

queries (Papadias et al., 2005), which considers minimum distance and maximum distance apart from the total distance.

The authors in (Li et al., 2005) proposed a method based on an ellipse to approximate the query distribution area and used a distance or a minimum bounding box derived from the ellipse to prune the R-tree nodes/data points for processing GkNN queries. In (Luo et al., 2007), an algorithm was proposed to answer a GNN query only for non-indexed data points using projection-based pruning strategies and in (Namnandorj et al., 2008), Namnandorj et al. developed algorithms for both indexed and non-indexed data points by estimating a search space using a vector property. User privacy was taken into consideration in (Hashem et al., 2010).

2.5.3 Relationship Perspective

Traditionally, the relationship between query object q and object of interest o ($q \prec o$) is a mono-directional relationship i.e. o is the nearest neighbour of q. However, many researchers proposed other types of relationship such as: bi-directional relationship in Reflexive kNNs and reversed relationship in Reverse k Nearest Neighbours. Each type is described in details as follows:

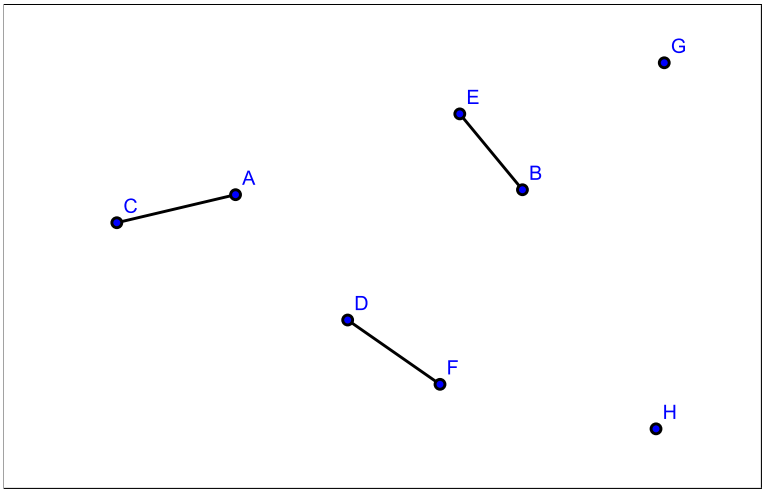


Figure 2.16: An illustration of an Reflexive-1NN query. A line is drawn between two objects if they are nearest neighbours of each other.

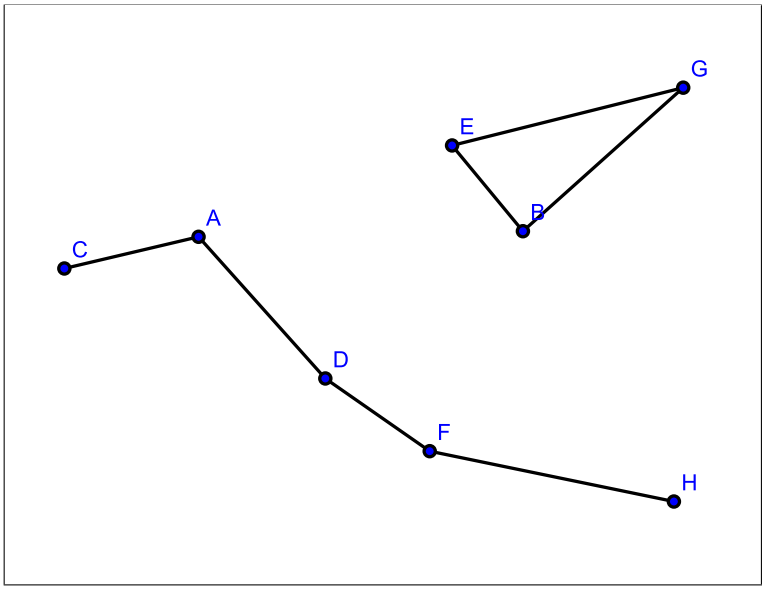


Figure 2.17: An illustration of a Reflexive-2NN query. A line is draw between two objects if one object is one of 2 nearest neighbours of another and vice versa.

Reflexive kNN

A reflexive-kNN (or it is also called a mutual-kNN) query requires a bi-directional relationship between the query point and the nearest neighbours. In other words, if o is an answer of a reflexive kNN generated from q, o is one of the kNNs of q, and vice versa, q is one of the k of o (Zhao, 2012). Even if reflexive-kNNs have been studied extensively in computational geometry (M., 1986), this topic is still rather new in spatial databases.

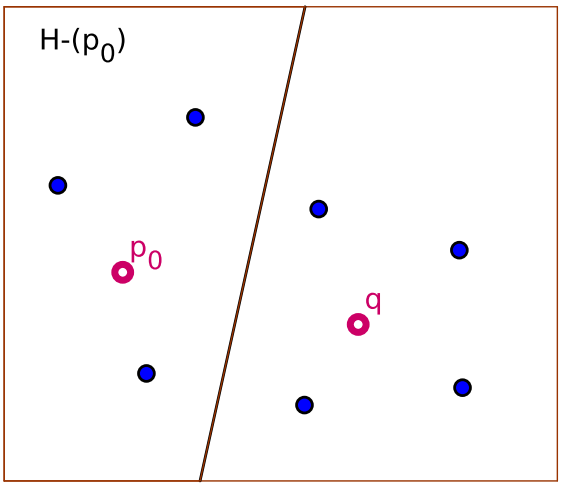


Figure 2.18: Half-space pruning. Any point that lies in the shaded half- space $H^{-}(p_0)$is always closer to p_0 than to q and cannot be the RNN for this reason.

In reflexive-kNN, two objects are connected if they have a mutual relationship which is Nearest Neighbour of each other as in Fig 2.16. Fig. 2.17 is an example of reflexive-2NNs. Compared with All-kNN, the relationship of reflexive-kNN is much stronger and thus harder to break. This property can be applied in clustering. For example, measuring the strength of a cluster based on the number of mutual relationships provided that the relationship here is based on reflexive-kNNs. Or it can be used to determine whether a new object can join a cluster by measuring the strength of relationships in the cluster before and after the joining.

Reverse Nearest Neighbour

This is another subject in this study. Reverse Nearest Neighbour (RNN) queries were first introduced in 2000 by Korn and Muthukrishnan (Korn and Muthukrishnan, 2000). They have since attracted a growing number of studies in a wide range of applications, such as decision support systems, mobile navigation systems and resource allocation. The problem is raised from the objects' point of view. Instead of finding the nearest objects from the query point q, it asks which objects consider q as their nearest neighbour.

Snapshot *RNN* query processing was introduced in 2000 by Flip Korn and S.Muthukrishnan(Korn and Muthukrishnan, 2000). It is based on an R-Tree by

pre-calculating a circle $C(o, N(o))$ for each object of interest o where the radius $N(o))$ is the distance from o to its nearest neighbour. The usage of perpendicular bisectors of the line segment between q and its peer, illustrated in Fig.2.18 to answer RkNN queries, was presented in (Tao, Papadias and Lian, 2004) based on a filter-refinement framework. Specifically, the filter step retrieves a set of candidate results that is guaranteed to include all the actual RNNs; the subsequent refinement step eliminates the false hits. The two steps are integrated in a seamless way that eliminates multiple accesses to the same index node (i.e., each node is visited at most once).

The author in (Stanoi et al., 2000) was the first to introduce a six pie region in solving RNN queries. Accordingly, it was proved that there are at most six RNNs for MRNN queries. The idea was extended to solve continuous MRNNs and BRNN in (Xia and Zhang, 2006; Kang et al., 2007) with consideration of movement of queries objects and objects of interest. The work in (Lian and Chen, 2009) proposed a probabilistic RNN over uncertain data using geometric pruning methods to reduce search space. There are also a number of studies working on RkNN queries (Wu et al., 2008; Cheema et al., 2011) using a convex polygon obtained from intersection of bisectors and influence zone, respectively. The Voronoi diagram was used in solving bichromatic RNN queries(Safar et al., 2009; Tran et al., 2010). To determine the shortest route to the query point, the query-type object or peer were referred as a Generator Point (GP), and the non query-type object or object of interest as Non-Generator Point (NGP).

2.6 Mobile P2P Query Processing in Nearest Neighbours Context

Spatial queries in general, and nearest neighbour queries in particular, are issued from the mobile peers and are expected to exhibit high spatial locality. That is, the results of a spatial query are very likely available on peers near the query peers. Therefore,

Mobile P2P Query Processing is a promising solution in this context. However, there has been not much Mobile P2P Query Processing in Nearest Neighbours Context.

A peer-tree was also presented in (Demirbas and Ferhatosmanoglu, 2003) to work with nearest neighbour queries but in static sensor networks; therefore, it is impossible to apply to mobile P2P environments due to fixed communication infrastructure. A distributed multi-dimensional index structure, called P2PRdNN was introduced in (Chen et al., 2006) to solve RNN queries. However, this research focused on monochromatic queries only and was based on super-peer-based overlay.

In addition, the authors in (Ilarri et al., 2006) defined a continuous location-based query processing system for mobiles agents. In (Chow et al., 2011) a framework for continuous queries based on report pushing direction is proposed. Range queries in mobile P2P environments are also investigated in (Gu and A., 2011). Spatial top-k queries are actually mentioned in (Padhariya et al., 2011); however, in this research, peers are not treated equally.

To the best of our knowledge, the first on-demand data sharing algorithm on kNN queries in mobile P2P networks was introduced in (Ku and Zimmermann, 2008). The study in this thesis is also inspired by this work. The main idea of the work in (Ku and Zimmermann, 2008) is based on the observation that if two mobile peers are in the proximity area of each other, the nearest neighbours of them may contain some objects in common. Therefore, the algorithm in (Ku and Zimmermann, 2008) makes use of cache data of neighbour query peers to answer nearest neighbour queries. Therefore, the next section will be dedicated to describe the data sharing algorithm in (Ku and Zimmermann, 2008).

2.6.1 Algorithms and Assumptions

This is a hybrid approach since the system in (Ku and Zimmermann, 2008) maintains a base station. An R-tree to support k nearest neighbour query processing is implemented at the base station. However, the concern of my research is on the P2P aspect.

In general, there are two types of wireless communications that are required.

- The first is to support medium-range or wide-range communication from peers to the base station. The technologies for wide/medium range communication can be 2G, 3G or 4G , which has been mentioned in the very beginning of Section 2.2.1.

- Second, short-range communications between one peers and another can be any type of Section 2.2.1 such as Bluetooth and IEEE802.11.

The system is designed for stationary objects of interest such as restaurants or gas stations. and mobile peers such as cars or mobile devices.

Overall, there are two main steps as below.

- **Step 1 (Peer-To-Peer Nearest Neighbours):** First, a mobile query q collects cached data of nearest neighbour objects in its peers to process its own Euclidean distance kNN search. If the result of query cannot be fully retrieved and verified by peers, it will be proceeded by the second step.

- **Step 2 (Base Station Query):** Second, based on the result from peers, the query can be forwarded to the base station and perform kNN best-first search algorithm to complete the nearest neighbour search algorithms.

The results collected from Step 1 above are verified by two processes including Single Peer Verification and Multiple Peer Verification.

Single Peer Verification

This process first attempts to verify the validity of k objects by sequentially verifying results obtained from each peer. The verification is to check whether an object of interest, o_1, retrieved from a peer p, is one of the k nearest neighbours of the query peer q or not by using the spatial relationship between the mobile peers and the objects of interest. More specifically, if o_i is one of the nearest neighbours of a peer p,

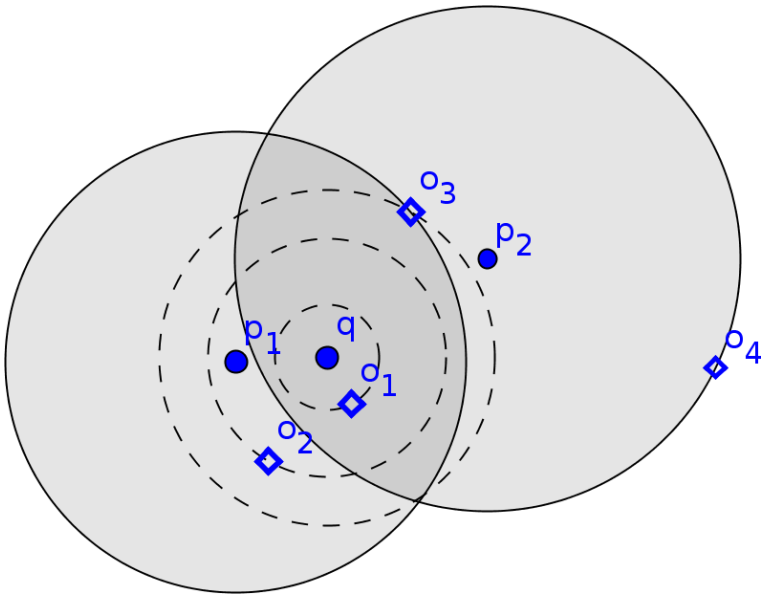


Figure 2.19: Verification Processes. *Single Peer Verification:* o_1 and o_2 can be verified by p_1 because the circles $C(q, dis_E(q, o_1))$ and $C(q, dis_E(q, o_2))$ are included in circle $C(p_1, dis_E(p_1, o_3))$, where o_1, o_2, o_3 are three nearest neighbours of p_1. However, p_1 cannot verify o_3. *Multiple Peer Verification:*$R_v = C(p_1, dis_E(p_1, o_3)) \cup C(p_2, dis_E(p_2, o_4)$. o_3 is verified by the collaboration of q_1 and q_2 as $C(q, dis_E(q, o_3)) \in R_v$

.

o_i is also one of the nearest neighbours of the query peer q if $dis_E(q, oi) + dis_E(q, p) \leq dis_E(p, o_k)$ whereas o_k is the k^{th} nearest neighbours of the peer p.

If the number of verified objects is fewer than k, the multiple peer verification process will be performed to finish the verification process with several peers simultaneously.

Multiple Peer Verification

This process makes use of the collaboration of many peers to verify results from peers. The verified regions that are bounded by the outermost nearest neighbour circle of all the peers p_i of the query peer q are merged into a verified region R_v. If an object of interest falls into the verified region R_v, it is verified as one of the nearest neighbours of q.

To merge multiple regions, the authors in (Ku and Zimmermann, 2008) approximated each circle region by polygonisation technique and then applied MapOverlay algorithm. The complexity of the MapOverlay operation is $O(nlogn + ilogn)$ where

n is the total number of edges of the two merged polygons and i is the number of intersection points.

Fig. 2.19 is an illustration of both verification processes. Accordingly, q is a query object which can collect 4 objects ($o_1, o_2, o_3 and o_4$) of interest from two other peers p_1 and p_2. o_3 is the farthest object among 3NNs of p_1 and o_4 is the one of p_2. It is easy to see that o_1 and o_2 are verified as nearest neighbours of q by p_1 since $C(q, dis_E(q, o_1))$ and $C(q, dis_E(q, o_2))$ are included in circle $C(p_1, o_3)$. However, p_1 cannot tell much about o_3 because a part of $C(p, o_3)$ is outside $C(p_1, dis_E(p_1, o_3))$. Only by the union of $C(p_1, dis_E(p_1, o_3))$ and $C(p_2, dis_E(p_2, o_4)$, o_3 is verified. In other words, o_3 is verified by the collaboration of q_1 and q_2 as $C(q, o_3) \in R_v = C(p_1, dis_E(p_1, o_3)) \cup C(p_2, dis_E(p_2, o_4)$.

2.6.2 Discussion

In general, the scenario is as follow. The query node collects and verifies information from peers. If it cannot verify those results, it will send them to the base station. The base station will finish this job and send the result back to the query node. This approach is efficient in reducing server workload and alleviating traffic congestion at the base station. The heart of this work is a set of verification algorithms that take into account the location of the query node, the peer and cached nearest neighbours of that peer. This research was also extended to be implemented on road networks in (Ku and Zimmermann, 2008).

Although research in (Ku and Zimmermann, 2008) can alleviate server workload in query processing and reduce communication to the base station, it is still a hybrid approach. The base station plays an important role in the network as the last guard to make sure all queries are completed, which leads to high cost of location updating and query processing. In addition, the base station is the weak point, and single point of failure in the whole network. Another drawback lies in its validation algorithms. It applies Algorithm MapOverlay whose complexity increases logarithmically to

validate results collected from peers. This algorithm is difficult to adapt to mobile P2P systems due to short-life battery and limited memory of mobile devices.

2.7 Limitations of Existing Work

2.7.1 Limitations of Existing Mobile Caching Techniques

There are three main limitations (L1-L3) of the existing caching technique in mobile environments.

- **L1**: relying on the centralised MSS.
- **L2**: using unstructured items which is not suitable for spatial queries.
- **L3**: not considering high frequent mobility of moving objects in cache invalidation techniques.

To address those issues, caching techniques in our system are based on a pure P2P approach without MSS. In addition, we define spatial data structure for each data item. In particular, each data item should contain information such as two dimensional location and object type. Cached data in a peer will be maintained in a priority queue which is sorted by the distance to from the data objects to the peer and updated when new queries are revoked from that peers.

2.7.2 Limitations of Existing Mobile P2P Query Processing Systems for Nearest Neighbours

- **L4**: requiring a base station as the centre of communication and query processing.
- **L5**: not addressing nearest queries in a pure and equal P2P systems.
- **L6**: heavy approaches in indexing objects and searching kNNs, which is hard to be adapted in mobile P2P systems due to short-life battery and limited memory.

- **L7**: Cached data is likely to be obsolete due to moving object's mobility. Therefore verification algorithms of cached data in (Ku and Zimmermann, 2006, 2008) lack accuracy.

- **L8**: no solution for Bichromatic-RNN and Group kNN in Mobile Peer to Peer Networks.

In response to the existing issues, the base station is eliminated out of our P2P systems. The whole system is divided into smaller region and each peer caches a regional map of interest objects. Communication between peers are required when a peer needs a new regional map. Indexing and searching methods are also customised in pure P2P manner to reduce storage and computation overhead.

To the best of our knowledge, the study in this thesis is the first to address three types of nearest neighbour queries with : k-Nearest Neighbour, Bichromatic-RNN and Group kNN for mobile ad-hoc networks in a pure P2P manner.

2.8 Chapter Summary

This chapter has provided common ground and discussion about state-of-the-art research in Mobile P2P Query Processing, especially for nearest neighbour queries. Basically, our research is based on short-range wireless technologies such as Bluetooth, WLANs and Wi-Fi Direct. Apart from GPS, each peer in the networks is equipped with a certain communication tool to sharing data. In addition, there is cache memory in each peer to store data locally from previous queries or the last time it has connected to a terminal or base station. The performance of data sharing such as accuracy rate and response time largely relies on mobile caching techniques.

The following chapters will investigate how Mobile P2P Query Processing applies to each type of queries.

"Laughter is sweet when enjoyed alone. But it becomes sweeter when you enjoy it together with the people around you. Your success must lead to the success others."

Israelmore Ayivor

3

k Nearest Neighbours

3.1 Overview

Location-aware services, which retrieve or answer spatial queries, have received more applications due to their increased importance in advanced database application. One of the most important fields of location-aware services is kNN search, which locates k closest objects from a given query object q in accordance to some distance metrics. Basically, it is an example of an optimisation problem whose goal is to find an object that minimises the distance to the query object, as described in Section 2.5.2.

In literature, much research deals with conventional kNN queries; however, their query processing is based on a centralised base station as in Fig. 1.1(a) (Waluyo

et al., 2009; Yu et al., 2005; Safar et al., 2011). Scalability, bottleneck and low fault-tolerant are critical issues of those centralised approaches, especially in large-scale systems.In particular, those systems contain only a central point of failure, which is likely to be corrupt in several scenarios. For example, on a battle field or natural disaster, the head-quarter is vulnerable to unavailability or traffic congestion (Ku and Zimmermann, 2008).

In response to the limitations of centralised query processing systems and the advance of mobile technologies, the emergence of mobile peer-to-peer query processing systems is a promising solution. A typical mobile P2P network consists of a collection of moving objects in Fig. 1.1(b) which are equipped with a GPS and able to communicate with each other in a P2P manner to share common interests via short-range wireless technology standards (Ciou et al., 2011; Goyal et al., 2011; Kim et al., 2011; Li et al., 2011; Luo et al., 2010; Xu et al., 2009). When a moving object q invokes a kNN query, it sends a query message to surrounding peers that are in its communication range instead of performing wide-range communication to the base station. In order to reply to the query from q, neighbouring peers return their cached data, for example, their kNNs retrieved two minutes ago. Validated data from those peers are used as kNN answers.

Generally, energy efficiency and query performance are the most critical issues when dealing Mobile P2P Query Processing Systems. Many researchers have tried to address those issues, but most of them have not considered spatial kNN queries (Lehikoinen et al., 2006; Luo et al., 2010; Padhariya et al., 2011; Park and Valduriez, 2011; Xu et al., 2009). Although the research in (Ku and Zimmermann, 2008) deals with kNN queries, the algorithms are not based on pure P2P approaches, but a hybrid one which needs a base station as a back-up solution. More details of research in (Ku and Zimmermann, 2008) has been discussed in Section 2.6 of Chapter 2. Therefore, developing an energy efficient Mobile P2P Query Processing System that provides location-aware services such as kNN queries and their variants is a real and urgent need.

In this chapter, I introduce a novel mobile P2P system with the following overall contributions:

1. Introduce a new direction in mobile P2P query processing for Wireless Ad-hoc Networks in which the base station is eliminated and only collaboration of moving objects is harnessed.

2. Propose lightweight validation algorithms to filter results from peers and answer kNN queries by both individual peers and the collaboration of peers.

The simulation results show my proposed system is efficient in saving energy consumption to more than six times compared to centralised and hybrid systems in a reasonable mean latency of processing time.

The organisation of the rest of this chapter will be as follows. Definitions and notations that are used in this chapter, and also the rest of thesis, will be shown in Section 3.2. Subsequently, Section 3.3 will investigate the model and assumptions applied in our system. This is followed by detailed description of the proposed kNN query processing system in Section 3.4. An example of the system will be given in Section 3.5. Lastly, Section 3.6 will conclude the chapter.

3.2 Definitions and Notations

This section will present all definitions and notations, which will be use in this chapter and also in the rest of this thesis.

3.2.1 Definitions

In the context of Mobile P2P networks there are two different types of objects: static objects of interest and dynamic peers. Here the nearest neighbour problem can be formally defined as below.

Definition 3.2.1. (**Nearest Neighbour**) Let IO and MO be a set of objects of interest and a set of moving peers, respectively.

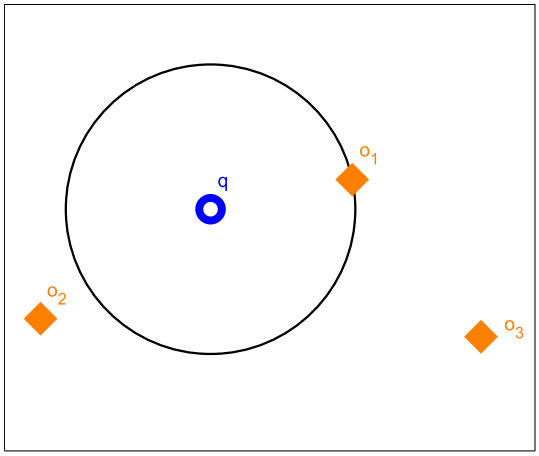


Figure 3.1: A nearest neighbour example. o_1 is the closest object of interest of q.

Let o_i be an arbitrary object of interest in IO, o_i is the **nearest neighbour** of the query node $q \in MO$ if and only if $\forall o_j \in IO - \{o_i\}, dis_E(o_i, q) \leq dis_E(o_j, q)$.

In Fig. 3.1, o_1 is the nearest neighbour of q as all other nodes are farther to q. The problem of Nearest Neighbours is actually a special case of k Nearest Neighbours when the value of k is 1.Fig. 3.2 is an example when k = 3. Here o_1, o_2 and o_3 are three nearest objects of interest of q.

Formally, the definition of Nearest Neighbours can be extended to k Nearest Neighbours as below:

Definition 3.2.2. (k **Nearest Neighbours**) $O = \{o_1, o_2, ..., o_k\}$ is a set of k **nearest neighbours** of q if and only if $\forall o_j \in IO - O, \forall o_i \in O, dis_E(o_i, q) \leq dis_E(o_j, q)$.

3.2.2 Notations

- m_i is a moving object with an ID of i.
- o_i is an object of interest with an ID of i.
- $MO = \{m_1, ..., m_M\}$ is a set of moving objects in the network. $|MO| = M$.
- $IO = \{o_1, ..., o_N\}$ is a set of objects of interest in the network. $|IO| = N$.

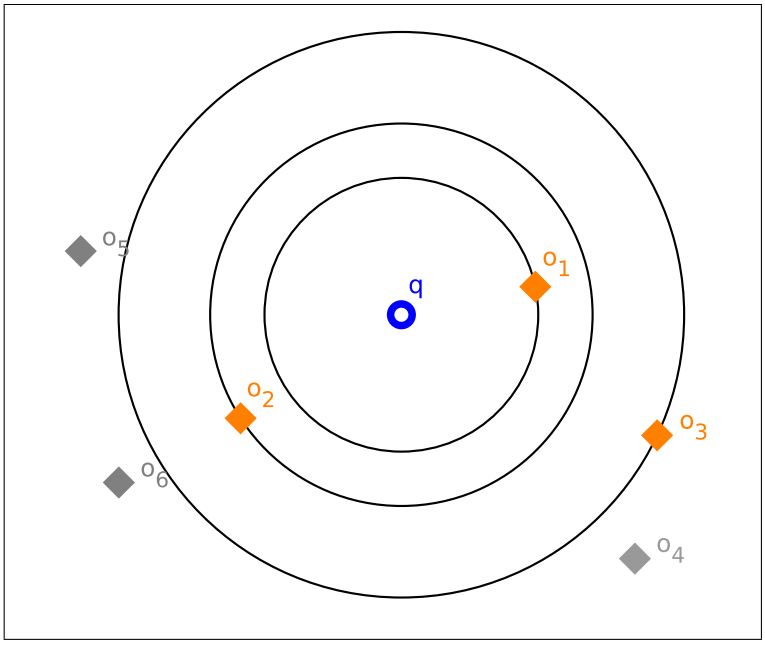


Figure 3.2: kNN when $k = 3$. In this example, o_1, o_2 and o_3 are three nearest objects of interest of q.

- q is a *query node*. When a moving object m_i generates a query and sends to its surrounding moving objects, it plays as a query node q.
- k is the number of objects of interest which query node q wants to retrieve.
- $P = \{p_1, ..., p_H\}$ is a priority queue of peers around query node q. $|P| = H$. P is sorted in ascending order of distance to q.
- p_i is a *peer node* with ID i. When a moving object m_i joins to answer a query from query node q, it plays as a peer node p_i.
- $IO_{p_i} = \{o_j \in IO : o_j \text{ is cached in } p_i\}$ is a set of objects of interest cached in peer p_i.
- $VO = \{o_i \in IO : o_i \text{ is validated}\}$ is a priority queue of objects of interest which are *validated* as one of kNN of q. VO is sorted in ascending order of distance to q.
- $IVO = \{o_i \in IO : o_i \text{ is invalidated}\}$ is a priority queue of objects of interest which are *invalidated* as one of kNN of q. IVO is sorted in ascending order of distance to q.

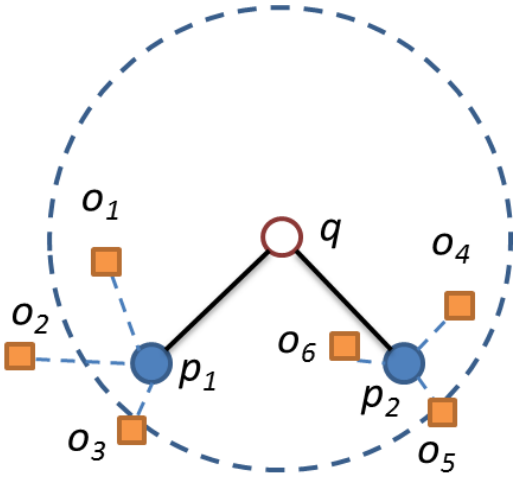


Figure 3.3: P2P Query Processing. q is the query node. $\{p_1$ and $p_2\}$ is the set of peer nodes. $\{o_j, j = 1..6\}$ is the set of interest objects. The circle is the communication range of q.

- R is communication range of a moving object.

For example, in Fig. 3.3, a moving object invoking a 3NN query is denoted as the query node q. p_1 and p_2 are two moving objects which play as peer nodes to answer the query from q. $\{o_1, o_2, o_3\}$ and $\{o_4, o_5, o_6\}$ are cached in p_1 and p_2 respectively as their own 3 nearest restaurants from their previous queries.

3.3 System Model and Assumptions

We consider a mobile network without any central supervision or base station where moving objects are dynamic. It is a symmetric system. Each moving object such as a smart mobile phone or a PDA is able to be both a query node and a peer of other nodes. It also has the ability to be self-aware of its location through an equipped GPS. It is noted that all location of moving objects and interest objects mentioned in this paper are actual physical location. Query point is always at the same location of the moving object issuing the query. In addition, objects of interest are stationary and distributed randomly in the network.

To enhance mobile P2P query processing, each moving object has cache memory to store spatial data of objects of interest from its kNN queries. Also moving objects

are equipped to conduct ad-hoc communication with other neighbour moving objects via Wi-Fi, Bluetooth or ZigBee.

We also assume that all moving objects cache their kNNs from their previous queries which are used to answer queries from their peers. To guarantee the verification of cached kNN results from peers , k is always no more than the size of cache memory.

3.4 System Details

Our ultimate aim is to harness collaborative power of peers in a pure P2P kNN query processing on mobile ad hoc environments in order to answer kNN queries. The core of our system is eliminating the base station by collecting results from peers and validating them by simple but effective algorithms.

There are four distinct types of message between a query node q and its peer p_j:

1. **Beacon message** ($beacon\_msg$) is broadcast from the query node to detect peers in its communication range.

2. **Acknowledgement message** (ack_msg) from peers to the query node is attached location of peers.

3. **Query message** ($query\_msg$) from the query node to the selected peers includes content of query such as: k value and type of objects of interest.

4. **Query reply** ($reply\_msg$) from peers to the query node attaches possible answer from cache of the peer. The answer consists of location and type of objects of interest.

Overall, the proposed system is divided into 2 primary phases: Initialisation and Peer Discovery Phase and Query Processing Phase. Each of them will be presented in details as the following:

3.4.1 Initialisation and Peer Discovery

Each moving object maintains a default map and the associated objects in its cache. Since moving objects move frequently, their peers also change. As a result, before starting to send queries, query node q needs to discover which moving objects around are in their communication range by simply sending a one-hop broadcast message. Moving objects receiving the broadcast message send an acknowledgement message which is attached their ID and location information.

The query node q collects all acknowledgement messages from the surrounding nodes. From the attached location of each acknowledgement message, q calculates the distance between q and the peer to put them into a priority queue which is ascendingly ordered by that distance.

More specifically, this phase is described in Algorithm 1: init. It is a note that, all algorithms is distributed. If the subject that performs methods is not specified, they are done by q. Otherwise, the subject will be indicated explicitly.

Algorithm 1: init

```
   Data: query node q, transmission range R
   Result: at q: priority queue of peers P
1  begin
2      P = ∅;
3      q broadcast a one-hop beacon_msg from q to every neighbour MO_j in
       range R;
4      if a peer p_i receives beacon_msg then
5          p_i sends ack_msg with its location and ID to q;
6      if q receives an ack_msg from p_i then
7          P = P ∪ {p_i};
8          if |P| = 1 then
9              q sends query_msg to p_i;
10 end
```

A priority peer queue is used in Algorithm 1 because the query node q will not always send its query with all of their peer. Instead, to *reduce communication cost and processing time* it sequentially sends queries to the peers in its priority queue and processes the results obtained. Accordingly, if q finds enough kNNs after processing the reply from the peer p_k, q no longer needs to connect with the rest of its peers.

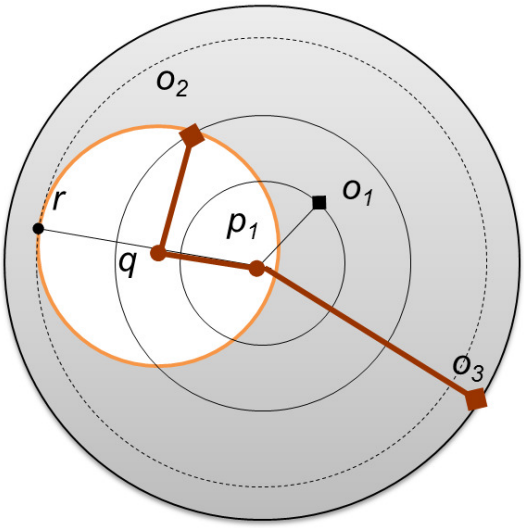


Figure 3.4: Single validation. For $k = 3$, $dis_E(q, o_2) + dis_E(q, p_i) \leq dis_E(q, o_3)$ hence o_2 is validated as one of kNNs of q.

3.4.2 P2P Query Processing

This is the heart of proposed system. After receiving the first acknowledgement message from one of their peers p_i, the query node q starts query processing by sending a kNN query to p_i. p_i, in its turn, attaches its cached set of kNNs, IO_{p_i}, from its prior queries in a reply $reply\_msg$ and sends back to q. After reaching the query node q, each object of interest o_j in IO_{p_i} is validated whether it is also one of nearest neighbour of q or not. If not the query is sent to the next peer in priority queue P.

Single Validation.

To validate the result from p_i, q at first uses Algorithm 2: single_validate enhanced from (Ku and Zimmermann, 2008) which is based on Lemma 1. We improve the performance of the original single validation by immediately stopping searching for kNNs when there are k objects of interest in priority queue VO of interest objects which are validated as one of kNN of q.

Lemma 1. Given a query node q, p_i is an arbitrary peer of q. p_i is assumed to cache a priority queue of its k objects of interest IO_{p_i} collected from previous query. o_k is the k^{th} nearest neighbour to p_i in IO_{p_i}. $\forall o_i \in IO$, if $dis_E(q, o_i) + dis_E(q, p_i) \leq dis_E(p_i, o_k)$ then $o_i \in VO$ or o_i is validated as one of kNNs of q.

Algorithm 2: single_validate

```
Data: query node q; int k; peer p_i; priority queue IO_{p_i}
Result: at q: priority queue VO, IVO; boolean isAnswered
1  begin
2      isAnswered = false;
3      foreach o_i ∈ IO_{p_i} do
4          if dis_E(q, o_i) + dis_E(q, p_i) ≤ dis_E(p_i, o_k) and o_i ∉ VO then
5              q adds o_i into VO;
6              if |VO| = k then
7                  isAnswered = true;
8                  return;
9          else
10             if |IVO| < k and o_i ∉ IVO then
11                 q adds o_i into IVO;
12             else if IOV ∋ o_k : dis_E(o_k, q) > dis_E(o_i, q) then
13                 q replaces o_k with o_i in IOV;
14 end
```

Lemma 1 and Algorithm 2 are illustrated in Fig. 3.4.

However, the disadvantage of both Algorithm 2 and the algorithm in (Ku and Zimmermann, 2008) is it returns enough kNNs only when k-th NN o_k in IO_{p_i} is aligned with q and p_i as consequence of triangle inequality, which seldom happens. In most cases q needs support from many peers to get enough k validated nearest neighbours.

Mutual Validation. In response to the drawback of Algorithm 2, q validates results from peers by taking advantage of the collaboration of peers as in Proposition 1.

Proposition 1. Let $C(p_j, r_j)$ be the circle of centre p_j, radius $r_j = dis_E(p_j, o_{k,j})$ and $p_j \in P$ and $o_{j,k}$ is the k^{th} object of interest in IO_{p_j}.
$\forall o_i \in IO$, if circle $C(q, dis_E(q, o_i))$ is fully covered by $\bigcup_{p_j \in P} C(p_j, r_j)$ then $o_i \in VO$ or o_i is mutually validated by multiple peers of q as one of its kNNs.

To calculate to overlap and verify whether $C(q, dis_E(q, o_i))$ is fully covered by $\bigcup_{p_j \in P} C(p_j, r_j)$, the authors in (Ku and Zimmermann, 2008) use expensive MapOverlay algorithm with logarithmic complexity. While moving objects such as mobile phones and PDAs are very limited in energy, the existing approach becomes unrealistic

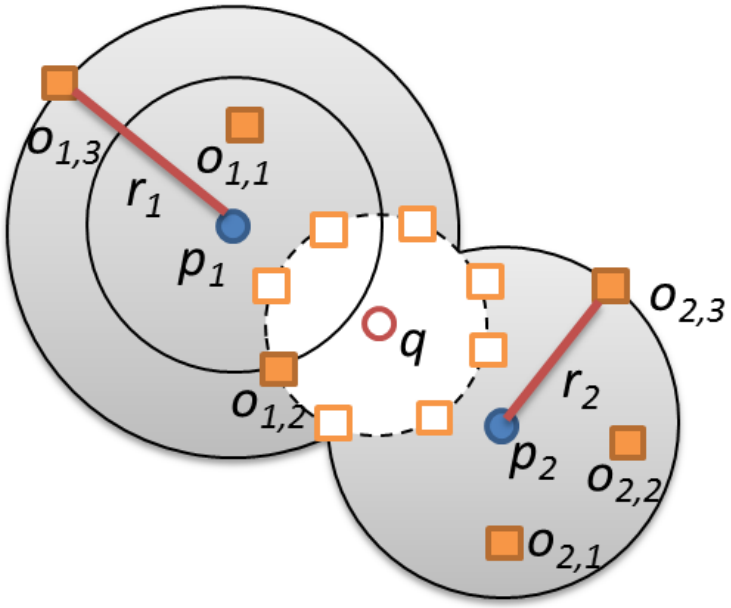


Figure 3.5: Mutual validation. For $k = 3$, all of 8 evenly distributed points on $C(q, dis_E(q, o_{1,2})$ are in $C(p_1, r_1)$ or $C(p_2, r_2)$ hence $o_{1,2}$ is mutually validated as one of kNNs of q.

and expensive. Therefore we develop a simplified approximated mutual validation algorithm to validate interest objects collected from one more many peers of q. Accordingly, we approximate circle $C(q, dis_E(q, o_i))$ by a polygon of h points evenly distributed on the circle. The more points the polygon is made of, the more accurate the algorithm is and the greater the computation cost is as well. It is a trade-off between accuracy and computation cost. Here we choose 8 points to keep it simple and light for implementation on moving objects. The Proposition 1 is re-stated as Proposition 2.

Proposition 2. Let $C(p_j, r_j)$ be the circle of centre p_j, radius $r_j = dis_E(p_j, o_{k,j})$ where $p_j \in P$ and $o_{j,k}$ is the k^{th} object of interest in IO_{p_i}.

With $o_i \in IO$, let $N = \{n_1, n_2, ...n_8\}$ be a set of 8 points that are reflective and evenly distributed on circle $C(q, dis_E(q, o_i))$

$\forall o_i \in IO$, $\forall n_h \in N$, if $n_h \in \bigcup_{p_j \in P} C(p_j, r_j)$ then $o_i \in VO$ or o_i is mutually validated by multiple peers of q as one of its kNNs.

Algorithm 3: mutual_validate

Data: query node q; int k; peer p_i; priority queue P, UVO;
Result: at q: priority queue VO, IVO; boolean isAnswered

```
1  begin
2      isAnswered = false;
3      foreach o_i ∈ UVO do
4          isFulFilled = true;
5          isChecked = false;
6          q find set N of 8 even distributed points on circle C(q, dis_E(q, o_i));
7          foreach n_h ∈ N do
8              foreach p_j ∈ P do
9                  r = dis_E(p_j, o_{k,j}), o_{k,j} is the k^th object of interest in IO_{p_j};
10                 if dis_E(p_j, n_h) <= r then
11                     isChecked = true;
12             if !isChecked then
13                 isFulFilled = false;
14                 break;
15             else
16                 isChecked = false;
17         if isFulFilled and o_i ∉ VO then
18             q adds o_i into VO;
19             q removes o_i from IVO;
20             if |VO| = k then
21                 isAnswered = true;
22                 return;
23 end
```

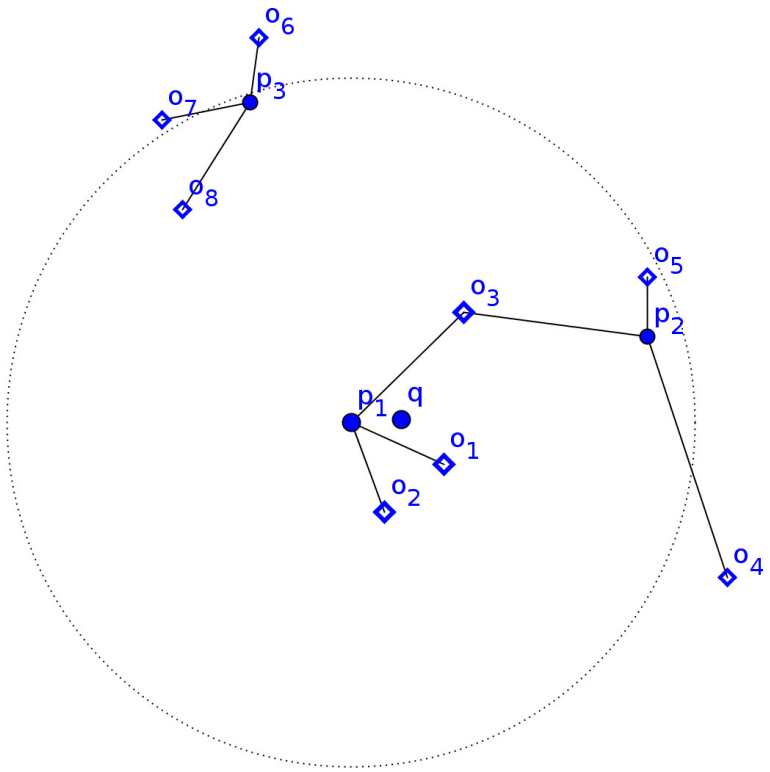


Figure 3.6: An example of kNN in Mobile P2P Networks. q and three of its peers, p_1, p_2 and p_3 with their objects of interest $\{o_1, o_2, .., o_8\}$. The circle represents the communication range of q. The links represents the cache relationship. For example, p_1 caches o_1, o_2 and o_3.

In summary, mutual validation is described in Algorithm 3 and illustrated in Fig. 3.5. The performance evaluation of kNN will be investigated and discussed in details in Chapter 6 together with simulation set-up and results.

3.5 An Example

To give better understanding about the proposed algorithms, this section will work through two kNN query examples in mobile P2P networks (Fig 3.6). It is assumed that a mobile user q invokes a kNN query to find 3 nearest restaurant (kNN where $k = 3$). The circle represents the communication range of q. As p_1, p_2 and p_3 are in the circle, they are potential peers of q. The links represents the cache relationship. For example, p_1 caches o_1, o_2, and o_3.

According to Algorithm 1, q first sends a broadcast message to all other mobile peers within its communication range (the circle in Fig. 3.6). p_1, p_2 and p_3 receive

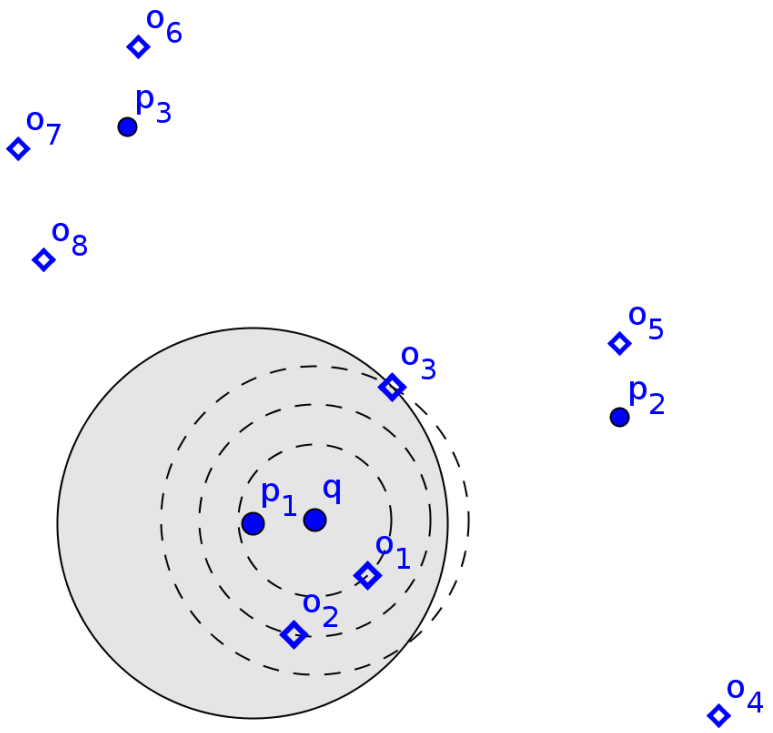


Figure 3.7: Single validation by p_1. o_1 and o_2 are validated while o_3 stays invalidated according to Algorithm 2.

the broadcast message from q and then send back an acknowledgement message to be registered in a priority queue, P, based on the distance from their location to q. Hence, $P =< p_1, p_2, p_3 >$.

Second, q sends p_1, the nearest peer from q in P, a kNN query and collects all cached objects of interest. They are o_1, o_2 and o_3. All these objects of interest are stored in a priority queue, sorted by the distance to p_1. $IO_{p_1} =< o_1, o_2, o_3 >$. Then single validation, based on Algorithm 2, is performed by q. o_1 and o_2 can be verified as nearest neighbours of q by p_1 since $C(q, dis_E(q, o_1))$ and $C(q, dis_E(q, o_2))$ are included in circle $C(p_1, dis_E(p_1, o_3))$. However, a part of $C(q, dis_E(q, o_3)$ is outside $C(p_1, dis_E(p_1, o_3))$ as in Fig. 3.7; therefore, o_3 is invalidated at the moment. Hence, priority queue of validated objects $VO =< o_1, o_2 >$ and that of invalidated objects $IVO =< o_3 >$. As $k = 3$ and there are only 2 objects in VO, q is looking for another object of interest.

Subsequently, p_2 is queried by q. The objects in $IO_{p_2} =< o_3, o_5, o_4 >$ are retrieved and validated by Algorithm 2 Single Validation. Unfortunately, none of objects in

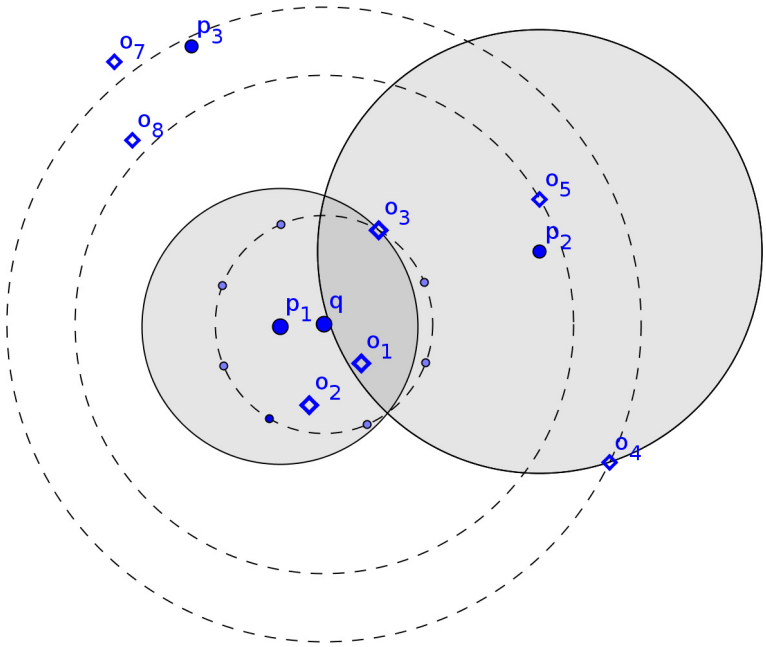


Figure 3.8: Mutual validation by p_1 and p_2. All 8 points on $C(q, dis_E(q, o_3))$ are in $C(p_1, dis_E(p_1, o_3)) \cup C(p_2, dis_E(p_2, o_4))$. Hence, o_3 is validated according to Algorithm 3.

IO_{p_2} can be validated because $C(q, o_3)$, $C(q, o_5)$ and $C(q, o_4)$ are not completely included in circle $C(p_2, dis_E(p_2, o_4))$ where o_4 is the farthest object to q in IO_{p_2}. IVO is updated as $IVO =< o_3, o_5, o_4 >$.

Now Algorithm 3 Mutual Validation can be used to validate objects in IVO. o_3 is the first to validate in IVO. q choose 8 even distributed point on the circle $C(q, dis_E(q, o_3))$ and check whether all of them are in either $C(p_1, dis_E(p_1, o_3))$ or $C(p_2, dis_E(p_2, o_4))$ to approximately validate o_3. As a result in Fig. 3.8, o_3 is validated and inserted to VO queue. In other words, $VO =< o_1, o_2, o_3 >$ and it also the answer for the 3NN query from q. q does not need to continue process o_4 and o_5. p_3 does not need to receive and process the query either. The query returns 3 validated nearest neighbours o_1, o_2 and o_3.

3.6 Chapter Summary

In this chapter, we presented a pure P2P query processing system in Mobile Ad-hoc Network. Based on novel gathering and validating algorithms, the system harnesses

collaboration of peers answer queries without any support from server side. In particular, a query peer collects cached nearest objects of interest from peers and select which objects can be used to answer its kNN query. Two lightweight validation algorithms are proposed to assist the querying process. Single Validation is done by a single peer. It is based on the position of query object, the peer, and objects cached in that peer. Multiple Validation takes advantage the collaboration of peers to approximately validate the object of interest.

By this way, we eliminate the limitation of centralised approaches such as single point of failure and the bottleneck problem. The simulation result shows our query processing system saves more than 6 times communication cost via the short-range network compared to both centralised and hybrid system. In addition the mean latency of query processing in our proposed system is up to 36% less than the other benchmarking systems. Also the performance of P2P system is better in an environment with high density of moving objects. Details of results from simulation will be discussed in Chapter 6.

"Share your knowledge. It is a way to achieve immortality."

Dalai Lama XIV

4

Reverse Nearest Neighbours

4.1 Overview

The growing importance of mobile communication systems has highlighted the need for solutions to many problems of geographic searching. One of these needs is the problem of Reverse Nearest Neighbour (RNN), in which a query returns all objects that consider the query object as their nearest neighbour. Reverse Nearest Neighbour (RNN) queries were first introduced in 2000 by Korn and Muthukrishnan (Korn and Muthukrishnan, 2000). They have since attracted a growing number of studies in a wide range of applications, such as decision support systems, mobile navigation systems and resource allocation. The problem is raised from the objects' point of

view. Instead of finding the nearest objects from the query point q, it asks which objects consider q as their nearest neighbour.

There are two types of RNNs. Firstly *monochromatic* RNN (MRNN), in which query objects and objects of interest are of the same category. A typical example of MRNN is that in mixed reality games such as BotFighters, players need to shoot only other players who are the closest to them. Therefore, the strategy is finding her own reverse nearest neighbours to avoid their shootings. Secondly, *bichromatic* RNN (BRNN), in which they are of different categories. Specifically, in MRNN, we have all objects are of the same type and the answer of a MRNN from a query object $q \in 0$ is $MRNN(q) = \{o_i \in O | \forall o_j \in O, dis_E(q, o_i) \leq dis_E(o_j, o_i)\}$, where $dis_E(,)$ is the Euclidean distance function. The problem becomes more challenging in BRNN since there are two distinct types of objects: P and O in the network. The return of BRNN are objects in O that consider the query object in P as the nearest neighbour. Formally, for a query node $q \in P$, $BRNN(q) = \{o_i \in O | \forall p_j \in P, dis_E(q, o_i) \leq dis_E(p_j, o_i)\}$. For instance, in everyday applications, police patrols can communicate with each other to distribute their team members to locations that needs them most, for example, sites of car accidents or intersections congested with traffic. In disaster management, to reduce redundancy, optimise human resources and maximise the support, a rescuer would rather save a victim who considers him as the closest support.

In the literature, there is a wide variety of research on RNN queries; however, their query processing is based on a centralised base station (Tao et al., 2007). Scalability, bottlenecks and low fault-tolerance are critical issues of those centralised approaches, especially in large-scale systems. In particular, centralised systems contain only a central point of failure, which is likely to be corrupt in several scenarios. For example, in a natural disaster, the headquarters is vulnerable to unavailability or traffic congestion (Nghiem, Waluyo and Taniar, 2013).

In response to the limitations of centralised query processing systems and the advance of mobile technologies, the emergence of mobile peer-to-peer (P2P) query

processing systems is a promising solution. A typical mobile P2P network consists of a collection of moving objects that are equipped with a Global Positioning System (GPS) and able to communicate with each other in a P2P manner to share common interests via short-range wireless technology standards such as IEEE 802.11 and IEEE 802.15, etc. (Luo et al., 2010; Tao, Papadias and Lian, 2004; Xu et al., 2009). When a moving object q invokes an RNN query, it sends a query message to surrounding peers which are in its communication range instead of performing wide-range communication to the base station. In order to reply to the query from q, neighbour peers return their cached data, for example, their kNNs retrieved two minutes ago. Using data received from its peers about objects of interest, q makes a selection to answer its RNN queries.

While most recent research has proposed index structure or query processing algorithms in centralised database systems (Korn and Muthukrishnan, 2000; Tao, Papadias and Lian, 2004; Kang et al., 2007; Wu et al., 2008; Safar et al., 2009; Lian and Chen, 2009; Tran et al., 2010; Cheema et al., 2011; Kang et al., 2010), there are very few studies based on P2P approaches. A distributed multi-dimensional index structure, called P2PRdNN, was introduced in (Chen et al., 2006) to solve RNN queries. However, this research focused only on static MRNN queries and it was based on super-peer-based overlay. In addition, the work in (Chow et al., 2011; Gu and A., 2011; Nghiem, Waluyo and Taniar, 2013) proposed a framework to find answers for spatial queries in mobile P2P environments, but it dealt only with range and kNN queries, respectively. A peer-tree was also presented in (Demirbas and Ferhatosmanoglu, 2003) to work with nearest neighbour queries, but in sensor networks. Speaking from our best knowledge, there is no previous work in the context of P2P-bichromatic RNNs (P2P-BRNNs) for Mobile Ad-hoc Networks even though there are many potential practical applications in this context with high rate of location update and under no central supervision.

Therefore, we have developed three distinct P2P algorithms focusing on bichromatic RNN queries based on a boundary polygon around a mobile query object.

The Brute-Force Search Algorithm (BFA) makes use of available information from the peers while two Boundary Search Algorithms reduce the number of queried peers. Tight Boundary Search Algorithm (TBA) is an enhancement of Regular Boundary Search Algorithm (RBA) in which a significant number of peers are filtered in processing an RNN query. Specifically, I make the following overall contributions:

1. We introduce a new direction in mobile P2P query processing in solving bichromatic RNN queries.

2. We propose and evaluate three different algorithms to search cached spatial data of objects of interest from mobile peers.

3. From our experimental study involving a real-data set, we found that our proposed system is substantially more efficient in saving time and energy compared with the centralised system.

4. Our simulation of three proposed algorithms also shows that the accuracy rate of BFA is higher than that in RBA in most scenarios. Vice versa, the RBA reduces query response time as a number of queried peers are pruned. Nevertheless, TBA is the optimal algorithm as it outperforms in saving processing and communication cost by filter unnecessary peers and maintain high accuracy rate.

The organisation of the rest of this chapter will be as the following. Section 4.2 will investigate the model and assumptions for our system. Subsequently, Definitions and Notations used in this chapter are presented in Section 4.4. Query models and message types will be discussed in Section 4.3. This is followed by detailed description of the proposed RNN query processing system including 3 different algorithms (BFA, RBA and TBA) in Section 4.5. Examples of the system will be given in Section 4.6. Lastly, Section 4.7 will conclude the chapter.

4.2 System Model and Assumptions

We consider a mobile network without any central supervision or base station where query objects and their peers are dynamic. It is a symmetric system where each moving object such as a smart mobile phone or a PDA plays as both a query node and a peer of other nodes. Moving objects are also able to be self-aware of their location through an equipped GPS. The location of moving objects and objects of interest mentioned in this paper are actual physical location.

Moving objects are equipped to conduct ad-hoc communication with other neighbour moving objects via Wireless Local Networks (WLANs), Wireless Local Personal Networks (WPANs), or WiFi Direct, an emerging form of P2P communication. In addition, objects of interest are randomly distributed in the network.

To enhance P2P query processing, a cache memory is assigned to store spatial data of objects of interest from its previous queries. The nature of the cache is a priory queue based on the distance from the cached objects of interest to the moving object. When the cache is full and the new objects of interest will be cached if the distance to the moving object is less than any that of another object in the cache. This deletion strategy assures that cached data are the most useful to answer future queries; therefore, it increases the accuracy of the query results.

4.3 Query Models and Message Types

Basically the proposed system is designed to answer RNN queries based on the available information from peers. The query point is always at the same location as the moving object issuing the query. The information from peers here is the results from previous queries such as RNN, kNN or range queries stored in cache memory of peers.

There are four distinct message types between the query node q and its peer p_j:

1. *Beacon message* ($beacon\_msg$) is broadcast from the query node to detect peers in its communication range.

Table 4.1: Notations used in Chapter 4

Notation	Meaning
q	the query node
p_i	a peer node with ID i
$P = \{p_1, ..., p_N\}$	a priority queue of peers of q. $N = \|P\|$.
IO_{p_i}	is a set of sorted objects of interest cached in p_i.
r	the communication range of a moving object.

2. *Acknowledgement message* (*ack_msg*) from peers to the query node is attached location of peers.

3. *Query message* (*query_msg*) from the query node to the selected peers to ask for objects of interest cached in those peers.

4. *Query reply* (*reply_msg*) from peers to the query node attaches a possible answer from the cache of the peer. The answer consists of the location and type of objects of interest.

4.4 Notations and Definitions

Table 4.1 lists notations that are going to be used in this chapter.

Definition 4.4.1. Let q be the query node and p_i be one of its peers. A *boundary line* (b_i) is defined as the perpendicular bisector line of line segment$< q, p_i >$

Definition 4.4.2. Provided (b_i) is the boundary line corresponding to p_i, the *positive half plane*, denoted as $H^+(p_i)$, is defined as

$$H^+(p_i) = \{x \in \mathbb{R}x\mathbb{R}, dis_E(x, q) \leq dis_E(x, p_i)\} \tag{4.1}$$

Similarly, the *negative half plane*, denoted as $H^-(p_i)$, is defined as:

$$H^-(p_i) = \{x \in \mathbb{R}x\mathbb{R}, dis_E(x, q) \geq dis_E(x, p_i)\} \tag{4.2}$$

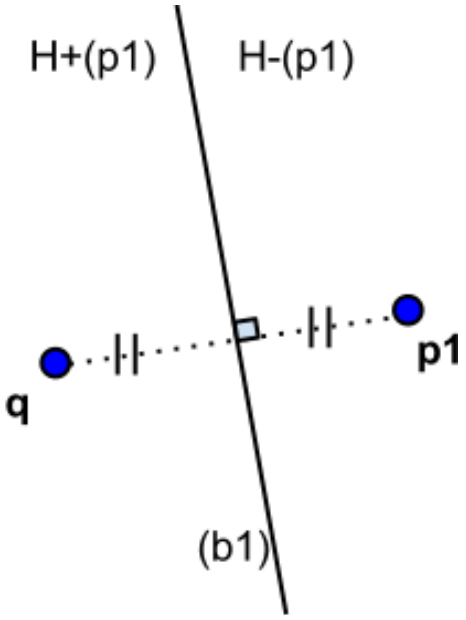


Figure 4.1: The boundary line (b_1) between query object q and its peer p_1 separates the positive half plane $H^+(p_1)$ and the negative half plane $H^-(p_1$.

It is easy to see that the boundary line (b_i) separates $H^-(p_i)$ and $H^+(p_i)$ because $\forall$ point $x \in (b_i), dis_E(x,q) = dis_E(x,p_i)$.

Fig. 4.1 illustrates the boundary line (b_1) between q and p_1. All objects of interest in the $H^+(p_1)$ are closer to q than to p_1.

Definition 4.4.3. Let P be the set of peers around query node q. A *boundary region* is the intersection of positive half spaces from peers or

$$B = \bigcap_{p_i \in P} H^+(p_i) \tag{4.3}$$

Definition 4.4.4. If B is closed, B is called a *boundary polygon*, $B =< V, E >$ where V is a set of its vertices and E is a set of its edges.

It is easy to derive the following *properties* of $B =< V, E >$:

$\forall v_i \in V, \exists m, n$ such that $v_i \in (b_m) \cap (b_n)$.

$\forall e_i \in E, \exists j$ such that e_i is a line segment on (b_j).

The boundary polygon B of q in Fig. 4.2 is the polygon bounded by $< v_1, v_2, v_3, v_4 >$. $v_1 \in b_2 \cap b_3$, $v_2 \in b_1 \cap b_3$, and so on. Line segment $< v_1, v_2 >$ is on b_3, $< v_2, v_3 >$ on b_1, and so on.

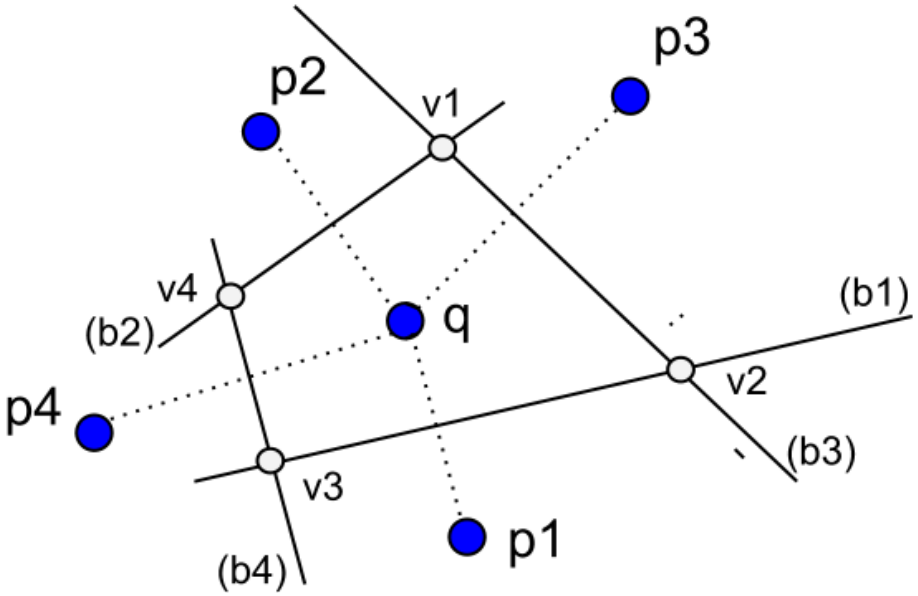


Figure 4.2: Boundary polygon B of q is bounded by $< v_1, v_2, v_3, v_4 >$

Definition 4.4.5. The boundary polygon B is called a *tight polygon* iff any object of interest o_i inside B regards q as the closest moving object.

4.5 System Details

Our ultimate aim is to harness the collaborative power of peers in a pure P2P RNN query processing on mobile environments in order to answer RNN queries. The core of our system is eliminating the base station by collecting results from selected peers by time and energy effective boundary-polygon-based algorithms. Overall, the proposed system is divided into 3 primary phases: 1)Initialisation and Peer Discovery Phase, 2)Constructing a Boundary Polygon and Sending Queries and lastly 3)Pruning Objects of Interest. Each phase is described in detail as below.

4.5.1 Initialisation and Peer Discovery

Each moving object maintains a default map of the associated objects in its cache. Since objects move frequently, their peers also change. As a result, before starting to send queries, query node q needs to discover which moving objects are in its

communication range by simply sending a one-hop broadcast message. Moving objects receiving the broadcast message send an acknowledgement message which is attached their ID and location information.

The query node q collects all acknowledgement messages from the surrounding nodes. From the attached location of each acknowledgement message, q calculates the distance between q and the peer to put them into a priority queue which is ascendingly ordered by that distance.

More specifically, this phase is described in Algorithm 4. It is a note that, all algorithms is distributed. If the subject that performs methods is not specified, they are done by q. Otherwise, the subject will be indicated explicitly. A priority peer queue is used in Algorithm 4 because the query node q does not always send its query to all of its peers. Instead, to *reduce communication cost and processing time* it sequentially sends the query to the peer in its priority queue and processes obtained results. Accordingly, q does not need to communicate to all of its peers to build the boundary polygon.

Algorithm 4: Initialisation and Peer Discovery

Data: query node q, transmission range R
Result: at q: priority queue of peers P

```
1 begin
2     P = ∅;
3     q broadcasts a one-hop beacon_msg to every peer p_i in range R;
4     if a peer p_i receives beacon_msg then
5         p_i sends ack_msg with its location and ID to q;
6     if q receives an ack_msg from p_i then
7         q updates P = P ∪ {p_i};
8 end
```

Each query node is assigned an acknowledgement time-out period. The query node waits to receive acknowledgement messages from its peers during that period. It is noted that the acknowledgement messages include the location of the peers. After that, it gets the first peer on the P list to start query processing.

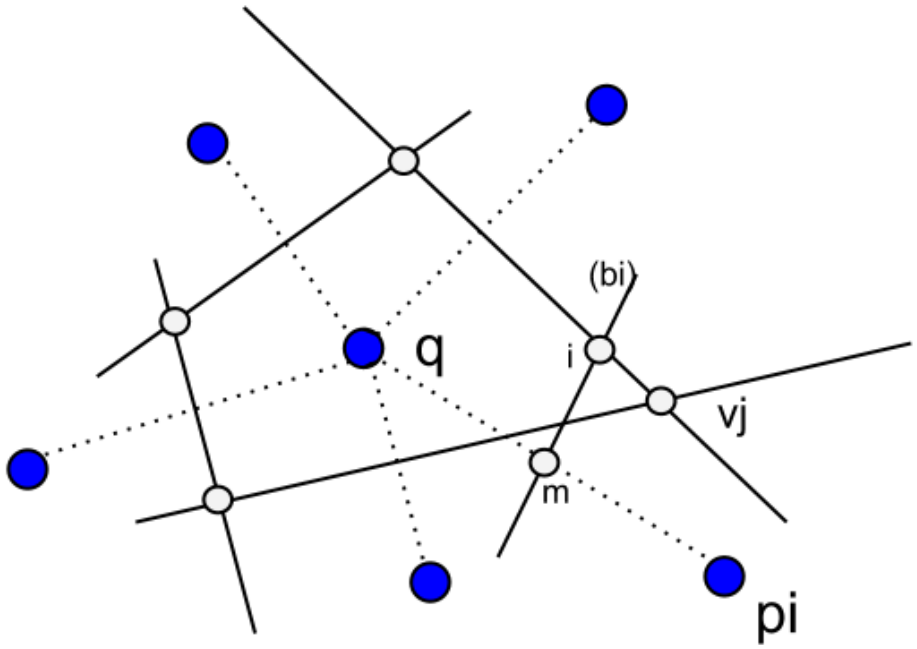


Figure 4.3: The illustration of Lemma 2's Proof.

4.5.2 Constructing a Boundary Polygon and Sending Queries

The query node starts query processing by popping out the information of the first peer in the P queue and starts building the boundary polygon. The polygon is created by drawing perpendicular bisector lines b_i for peers p_i from queue P. As Definition 4.4.3, the boundary polygon is the close region bounded by the intersection of positive half spaces $H^+(p_i)$, or $B = \bigcap_{p_i \in P} H^+(p_i)$. The constructing process is based on the Lemma 2 below.

It is assumed that there is a polygon $B =< V, E >$ around q and v_j is the furthermost vertex of B from q, i.e. $\forall v_m \in V, dis_E(q, v_m) \leq dis_E(q, v_j)$.

$P = \{p_1, ..., p_H\}$ is a priority queue of peers of query node q and it is sorted in ascending order of distance to q. $|P| = H$.

We create the polygon by drawing perpendicular bisector lines for each peer p_i from queue P. The lemma is stated as follows.

Lemma 2. If $\exists p_i \in P$ at the first position such that $dis_E(q, p_i) \geq 2dis_E(q, v_j)$, then B is a tight polygon. Put another way, we do not need to consider remaining peers left in the queue P and stop creating the polygon.

Proof. Let $B =< V, E >$ be a tight polygon. p_i is a peer of q such that

$$dis_E(q, p_i) \geq 2dis_E(q, v_{max}) \tag{4.4}$$

(b_i) is the boundary line of p_i where m is the intersection point of line segment (q, p_i) and (b_i) as in Fig. 4.3.

It is assumed that:

$$\exists e_i \in E, \exists i \notin V, (b_i) \cap e_i = i \tag{4.5}$$

We are going to show that the assumption is a contradiction.

As the assumption, v_{max} is the vertex of B farthest from q, i.e. $\forall v_m \in V, dis_E(q, v_m) \leq dis_E(q, v_{max})$.

Hence, any point on e_i is inside the circle $C(q, dis_E(q, v_{max}))$ or

$$dis_E(q, i) < dis_E(q, v_{max}) \tag{4.6}$$

as $i \notin V$.

Since segment $(q, m) \perp (b_i)$,

$$dis_E(q, m) \leq dis_E(q, i) \tag{4.7}$$

From 4.6 and 4.7,

$$dis_E(q, m) \leq dis_E(q, i) < dis_E(q, v_{max}) \tag{4.8}$$

Hence,

$$2dis_E(q, m) < 2dis_E(q, v_{max}) \tag{4.9}$$

Since m is the midpoint of line segment $< q, p_i >$, finally we obtain $dis_E(q, p_i) < 2dis_E(q, v_{max})$, which contradicts to 4.5. □

As long as the stop condition is not satisfied, q continues the boundary constructing process and sending queries to the next peer in the P queue. Here we develop three different algorithms which are Algorithm 5 - Brute-Force Search Algorithm (BFA), Algorithm 6 - Regular Boundary Search Algorithm (RBA) and Algorithm 7 - Regular Boundary Search Algorithm (TBA).

Algorithm 5: Brute-Force Search Algorithm

```
Data: query object q;
at q: peer list P, boundary polygon B, list V of intersection nodes or vertices
in B, pre-defined time-out;
Result: at q: priority queue IOq
1  begin
2      V = ∅; B = ∅;
3      boolean done = false ;
4      foreach p_i ∈ P do
5          if time-out is expired then
6              break;
7          if !done then
8              q calculates a boundary line (b_i);
9              q constructs/updates polygon B;
10             q finds intersection nodes and updates V;
11             if B is a polygon then
12                 q finds the furthest vertex v_j of B;
13                 if dis_E(q, p_i) >= 2dis_E(q, v_j) then
14                     done = true;
15         q sends query_msg to p_i;
16         p_i receives query_msg and sends reply_msg;
17         q receives reply_msg;
18         IO_q = IO_q ∪ IO_{p_i};
19 end
```

In BFA, which is illustrated in Algorithm 5, query peer q sends a query message ($query_{msg}$) to all of the peers in P, which were discovered in the previous phase until the predefined time-out expires. Before sending $query_{msg}$ to p_i in P, q calculates the boundary line b_i and construct the boundary polygon B. This polygon B will be used to prune objects of interest in the next phase.

Although BFA can make the most of information from peers to answer RNN queries, the query node does not select which peers to communicate, but broadcasts queries to all peers in its communication range. Therefore, this algorithm is not

Algorithm 6: Regular Boundary Search Algorithm (RBA)

Data: query node q;
at q: peer list P, boundary polygon B, list V of intersection nodes or vertices in B, pre-defined time-out
Result: at q: set of sorted objects of interest IO_q

```
1  begin
2      V = ∅; B = ∅;
3      boolean stopHit = false;
4      while time-out is not expired && !stopHit && P ≠ ∅) do
5          P pops the first element p_i;
6          if B is a polygon then
7              q finds the furthest vertex v_max of B;
8              if dis_E(q, p_i) >= 2dis_E(q, v_max) then
9                  q sets stopHit = true;
10         if !stopHit then
11             q calculates a boundary line (b_i);
12             q constructs and updates polygon B;
13             q finds intersection nodes;
14             q updates V;
15         q sends query_msg to p_i;
16         p_i receives query_msg and send reply_msg;
17         q receives reply_msg;
18         IO_q = IO_q ∪ IO_{p_i};
19 end
```

effective due to communication overhead if the density of moving objects is high. To limit the number of peers communicated by q, we propose the RBA (Algorithm 6). Accordingly, only the peers which contribute in building the tight boundary polygon B are sent queries. The decision to stop sending the query to the remaining peers in the list P is based on Lemma 2. The rest of RBA is the same as BFA.

Less peer communication in RBA leads to less query reply from peers to process. As a a result, BFA can reduce the communication and computation overheads. However the trade-off is that RBA can lower the accuracy of RNN answer since the answer of RNN query may be cached in other peers that q has not asked.

TBA is proposed as the optimal solution in which a tighter boundary is presented to filter more redundant peers while maintaining a higher accuracy rate. Apart from applying the decision to stop sending the query message to peers when the stop condition is satisfied as RBA, TBA also adopt the filter condition described in Lemma 3. It is a note that the time-out mechanism is also adopted in TBA as it does in BFA and RBA.

Lemma 3. Let $B =< V, E >$ is a polygon that consists of a set of vertices V and a set of edges E. $\forall p_i \in B, \forall v_j \in V, if dis_E(p_i, v_j) > dis_E(v_j, q)$, then p_i does not contribute in construction of the tight polygon or p_i is filtered from further processing.

Fig. 4.4 is an illustration of Lemma 3.

Proof. Let p_i be a peer of q, such that its boundary line b_i intersects with the boundary polygon at A and B. It is assumed that p_i is outside the circle $C(F, dis_E(F, q))$.

We are going to show that the assumption leads to a contradiction.

Let C be the intersection point of line segment $< q, p_i >$ and (b_i) as illustrated in Fig. 4.5 whereas $\triangle DFG$ is the boundary polygon. Then there must be a vertex F that

$$dis_E(F', q) > dis_E(C, q) \tag{4.10}$$

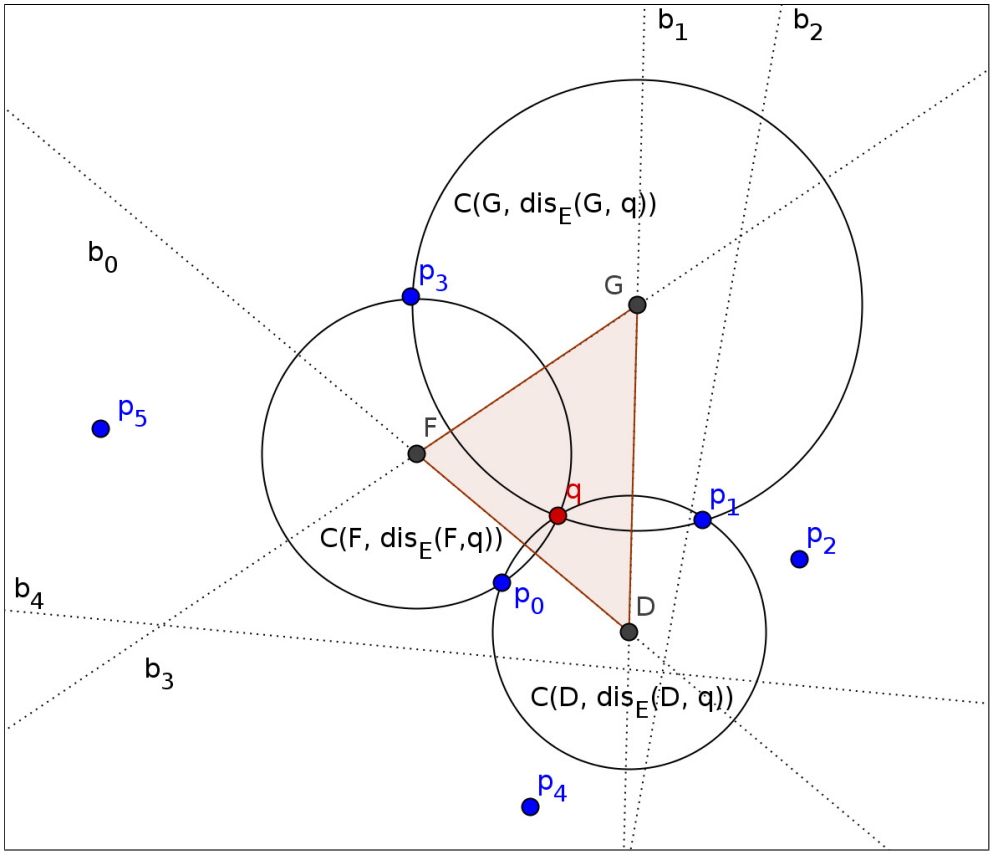


Figure 4.4: The illustration of Lemma 3. Any peers that are outside of 3 circles $C(F, dis_E(F, q))$, $C(D, dis_E(D, q))$, $C(G, dis_E(G, q))$ are filtered. In this example, p_1, p_4 and p_5 are filtered. Their boundary lines (b_1, b_4 and b_5) do not intersect with the boundary polygon $\triangle GFD$.

where F' is a projection point of F on $< q, p_i >$.

On the other hand, let I be the intersection of the line $< p_i, q >$ and $C(F, dis_E(Fq))$. Due to the assumption that p_i is outside the circle $C(F, dis_E(F, q))$ and all q, p_i and I are on a straight line, we obtain:

$$dis_E(p_i, q) > dis_E(I, q) \tag{4.11}$$

As C is the intersection point of line segment $< q, p_i >$ and (b_i), C is also the midpoint of line segment $< q, p_i >$ whereas F' is the midpoint of line segment $< I, q >$. Thus, combining with the inequation 4.11, we derive:

$$dis_E(C, q) > dis_E(F', q) \tag{4.12}$$

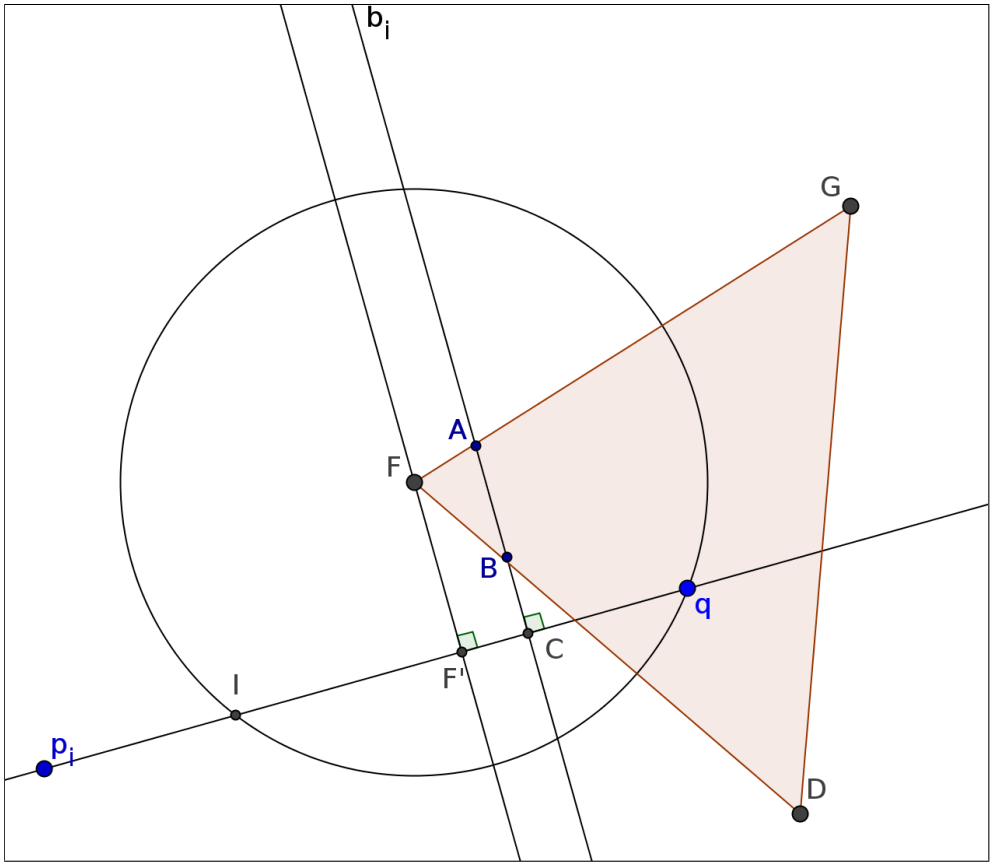


Figure 4.5: The illustration of Lemma 3's proof.

which contradicts to(4.10).

Similarly, if p_i is outside of $C(F, dis_E(F, q)), C(D, dis_E(D, q))andC(G, dis_E(G, q)$, the boudary line b_i will not intersect with any angle of the boundary triangle $\triangle$GFD.

□

Based on Lemma 3, TBA is stated as Algorithm 7 where a number of unnecessary peers are expected to be filtered.

4.5.3 Pruning Objects of Interest

After collecting cached objects of interest from peers, in this phase the query node q prunes objects of interest which do not consider q as the nearest neighbour or put another way, those objects of interest are not part of RNN's answer to the query. Based on the definition of tight polygon, all of the objects which are outside B are pruned.

Algorithm 7: Tight Boundary Search Algorithm (TBA)

Data: query node q;
at q: peer list P, boundary polygon B, list V of intersection nodes or vertices in B, pre-defined time-out
Result: at q: set of sorted objects of interest IO_q

```
1  begin
2      V = ∅; B = ∅;
3      stopHit = false;
4      while time-out is not expired && !stopHit &&P ≠ ∅ do
5          P pops the first element p_i;
6          if B is a polygon then
7              q finds the furthest vertex v_max of B;
8              if dis_E(q, p_i) >= 2dis_E(q, v_max) then
9                  stopHit = true;
10         if !stopHit then
11             q calculates a boundary line (b_i);
12             q constructs and updates polygon B;
13             q finds intersection nodes;
14             q updates V;
15         foreach p_m ∈ P do
16             foreach v_j ∈ V do
17                 if dis_E(p_m, v_j) > dis_E(v_j, q) then
18                     q removes p_m from P;
19         q sends query_msg to p_i;
20         p_i receives query_msg and send reply_msg;
21         q receives reply_msg;
22         IO_q = IO_q ∪ IO_{p_i};
23 end
```

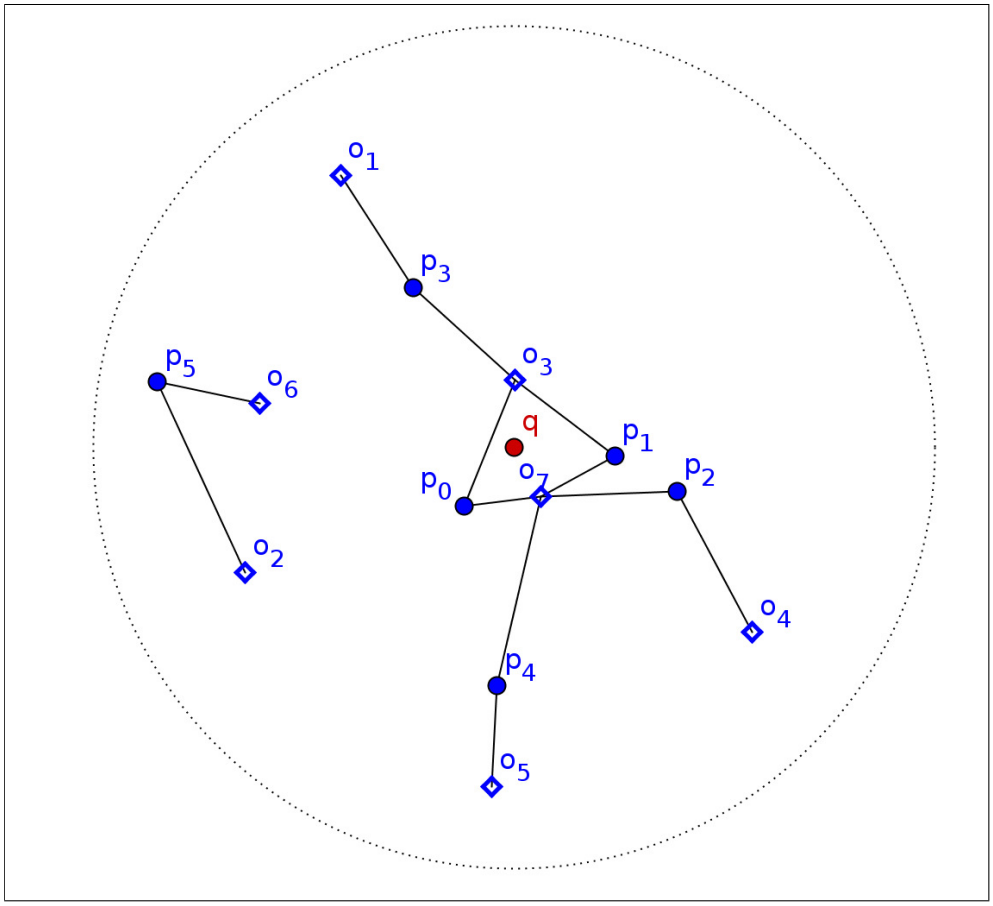


Figure 4.6: An example of RNN in Mobile P2P Networks. q and its five peers, $p_1, p_2, ..., p_5$. Each peers cache its two nearest objects of interest, which is represented by a continuous line. In general, there are 7 objects of interest, $\{o_1, o_2, .., o_7\}$. The circle represents the communication range of q.

Finally, the overall RNN query processing is concluded in Algorithm 8.

Algorithm 8: Main_Algorithm

Data: query node q; at q: boundary polygon B;
Result: at q: set of sorted objects of interest IO_q at q

```
1 begin
2     q calls Init() (Algorithm 1);
3     q calls BFA (Algorithm 2) or RBA (Algorithm 3) or TBA (Algorithm 4);
4     foreach o_i ∈ IO_q do
5         if o_i is outside B then
6             q updates IO_q = IO_q \ {o_i};
7 end
```

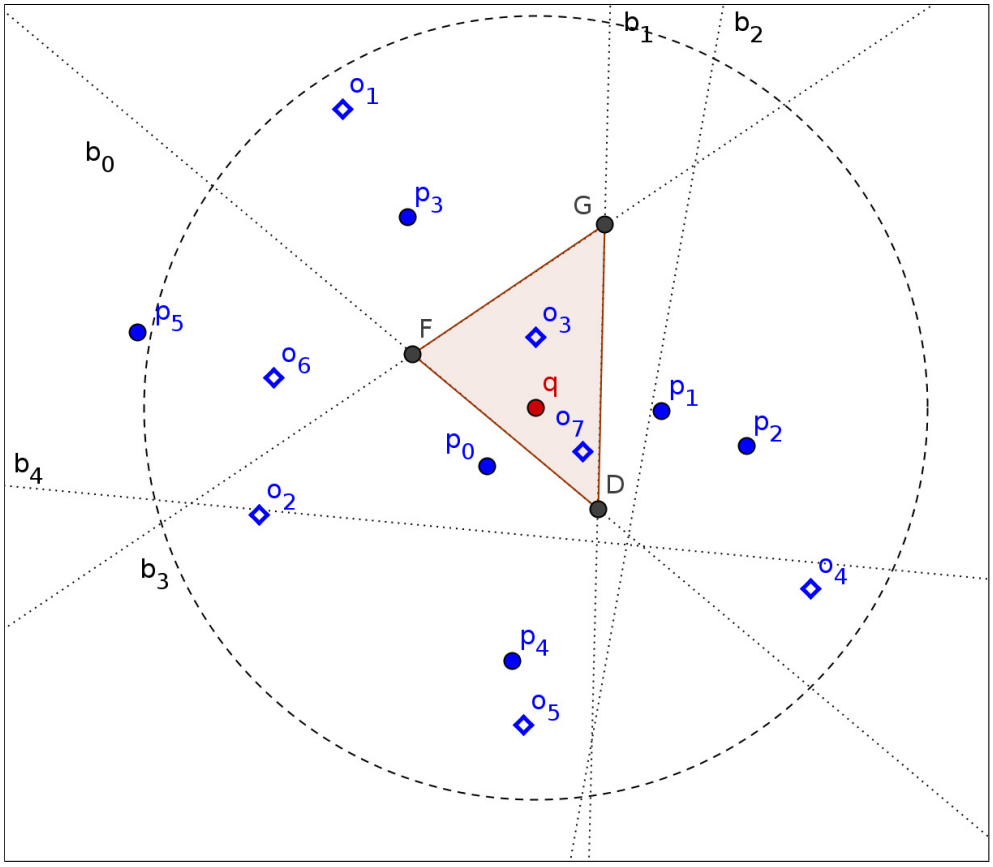


Figure 4.7: An example for BFA. The boundary polygon $\triangle DFG$ is built based on the boundary line b_1, b_2 and b_3.

4.6 Examples of BFA, RBA and TBA

This section is going to give an example that illustrate how the system works with 3 different search algorithms (BFA, TBA and RBA).

It is assumed that after the first stage, query object q identifies 5 peers and puts them into the priority queue $P =< p_0, p_2, ..., p_5 >$ within its communication range as Fig. 4.6. Each peers cache its two nearest objects of interest, which is represented by a continuous line. For example, o_3 and o_7 are two nearest neighbours cached by p_0, o_2 and o_6 by p_5 and so on. Many peers can cache one or more objects of interest in common such as p_0 and p_1 both cache o_3.

4.6.1 An Example of BFA

If BFA is applied, q performs brute-force search as Algorithm 5, broadcast queries all peers in its communication. It is assumed that all of the peers in P are connected and returned the response with their cached objects of interest before time-out threshold.

Then the boundary polygon $\triangle DFG$ is built based on the boundary line b_0, b_1 and b_3 corresponding to p_0, p_1 and p_3 in P. Apparently, boundary lines of b_2 and b_4 do not intersect with $\triangle DFG$. As p_5 is outside the circle $C(q, 2dis_E(q, G)$, where G is the farthest vertex in $\triangle DFG$ from q, p_5 does not join the construction of the boundary polygon. Therefore $\triangle DFG$ is the tight boundary polygon.

The next step is pruning objects of interest. Only the objects of interest in the boundary polygon, o_3 and o_7 are kept as the answer of RNN query.

4.6.2 An Example of RBA

Unlike in BFA, q does not broadcast queries to all peers in RBA. Instead, to limit the number of peers communicated by q, q stops sending the queries to its peers when the tight boundary polygon is constructed. In Fig 4.7, p_5 is outside the circle $C(q, 2dis_E(q, G)$; therefore $\triangle DFG$ is the tight boundary polygon as said in the previous example. As a result, q does not need to send the query to p_5. If q has more peers farther than p_5, all of them will be filtered from further query processing. The state of pruning objects of interest is exactly the same to BFA.

4.6.3 An Example of TBA

TBA provides a better boundary to filter peers from query processing. Accordingly, after processing p_3 and forming the boundary polygon $\triangle DFG$, any peers outside of circle $C(F, dis_E(F, q)), C(D, dis_E(D, q)), C(G, dis_E(G, q))$ are filtered as Fig. 4.8. In this example, p_4 and p_5 are filtered. Hence, communication cost and query response time are optimised. Like BFA and RBA, objects of interest that are outside the boundary polygon $\triangle DFG$ are pruned. Vice versa, the rest of objects of interest are returned as the answer of the query.

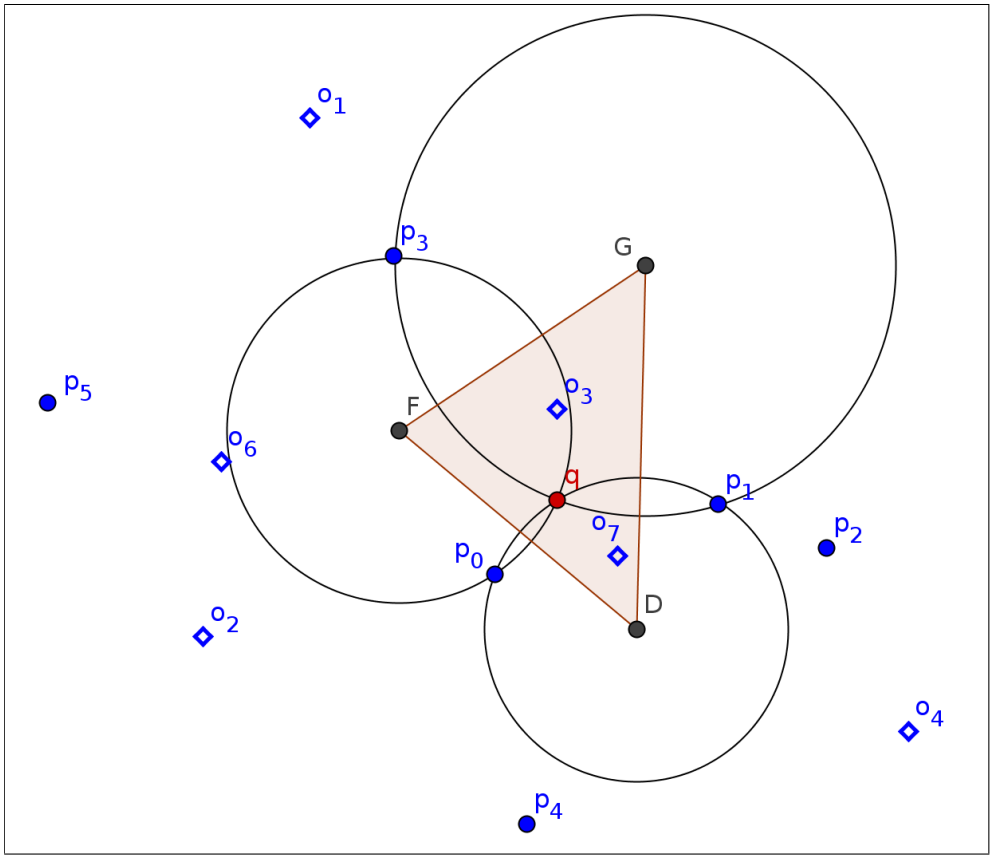


Figure 4.8: An example for TBA. Only peers inside $C(F, dis_E(F, q)), C(D, dis_E(D, q)), C(G, dis_E(G, q))$ are queried.

4.7 Chapter Summary

To sum up, this chapter investigated the potential of a pure P2P query processing system in Mobile Networks to solve RNN queries based on 3 distinct novel search algorithms. In particular, the Brute-Force Search Algorithm (BFA) makes use of available information from the peers while two Boundary Search Algorithms reduce the number of queried peers. Tight Boundary Search Algorithm (TBA) is an enhancement of Regular Boundary Search Algorithm (RBA) in which a significant number of peers are filtered in processing an RNN query. The system harnesses collaboration of peers to answer queries without any support from server side. As a consequence, we eliminate the limitation of centralised approaches, such as single point of failure and bottleneck problems.

The simulation design and result will be discussed in detail in Chapter 6. Overall, our P2P Search Algorithms significantly save communication cost via the short-range

network compared to the centralised system, regardless of the change to network parameters. In addition the mean latency of query processing in our proposed system is up to 43% less than the centralised system. The BFA provides the naive approach to make use of available cached data from the peers with time-out mechanism. The RBA and TBA reduce the number of peers involved in query processing and shorten response time as a result. Overall, the TBA is the most optimal option which achieve high accuracy with time and energy efficiency for the mobile network. In general, our P2P Search Algorithms is a practically feasible option for a large-scale and busy network.

"The network is opening up some amazing possibilities
for us to reinvent content, reinvent collaboration."

Tim O'Reilly

5

Group k Nearest Neighbour

5.1 Overview

The increasing use of location-based services has led to a new range of real-time services, such as location-based social networking (e.g Friend Finder (Strassman and Collier, 2004)), that enable a group of users to be involved in a single location-based query. For example, a group of users may want to meet at a place that minimises the total travel distance for them. The problem is called Group Nearest Neighbour (GNN), in which a selected set of objects $Q = \{q_1, .., q_M\}$ search for one target o such that the sum of distance from o to every q_i is minimum. If there is a set of objects $O = \{o_1, o_2, ..., o_k, k \geq 1\}$ returned, it is called GkNN where O is sorted by

total travel distance to every q_i. Therefore, GNN is a special case of GkNN when k equals 1.

Since GNN queries were first introduced in 2004 by Papadias, et al. (Papadias et al., 2004), it has attracted a growing number of studies in a wide range of applications, such as decision support systems, mobile navigation systems and resource allocation. However, their query processing is based on a centralised base station as in (Papadias et al., 2004; Lian and Chen, 2008; Hashem et al., 2010; Wu and Cao, 2012). Scalability, bottlenecks and low fault-tolerance are critical issues of those centralised approaches, especially in large-scale systems. In particular, those systems contain only a central point of failure, which is likely to be corrupt in several scenarios. For example, in a natural disaster, the headquarters is vulnerable to unavailability or traffic congestion (Nghiem, Waluyo and Taniar, 2013).

In response to the limitations of centralised query processing systems and the advance of mobile technologies, the emergence of mobile peer-to-peer (P2P) query processing systems is a promising solution. A typical mobile P2P network consists of a collection of moving objects. Each moving object is equipped with a Global Positioning System (GPS) and able to communicate with each other in a P2P manner to share common interests via short-range wireless technology standards (Xu et al., 2009; Luo et al., 2010). The overall idea is to harness the collaborative power of peers by using cached data to answer location-based queries such as GkNN.

While most recent research has proposed using index structure or query processing in centralised database systems, there are few studies based on P2P approaches. In addition, the work in (Nghiem, Waluyo and Taniar, 2013; Chow et al., 2011; Nghiem, Maulana, Waluyo, Green and Taniar, 2013) proposed a framework to find approximate answers for spatial queries in mobile P2P environments, but it dealt only with range, k-Nearest Neighbour queries and Reverse Nearest Neighbour. A peer-tree was also presented in (Demirbas and Ferhatosmanoglu, 2003) to work with nearest neighbour queries in sensor networks. Speaking from my best knowledge, there is no previous work in the context of P2P-Group k Nearest Neighbour (P2P-GkNN)

for Mobile Ad-hoc Networks. Therefore, I have developed a novel P2P algorithm focusing on GkNN queries. Our P2P-GkNN algorithm makes use of all cached nearest neighbours from the peers to answer GkNN in the most efficient way in terms of overhead cost, processing time and accuracy rate.

In this chapter, I will present a novel mobile P2P system with the following overall contribution:

1. I introduce a new framework for mobile P2P query processing in solving GkNN queries.

2. I propose and evaluate a novel algorithm to search cached spatial data of objects of interest from mobile peers.

3. From our experimental study involving both real and synthetic data set, I found that the proposed algorithm is substantially more energy-efficient and save at least 32% latency time compared with the centralised GkNN algorithm while maintaining high accuracy rate more than 82%.

The organisation of the rest of this chapter will be as the following. First I will investigate the model and assumptions for our system (Section 5.2). Query models and message types will be discussed in Section 5.3. Subsequently, Notations used in this chapter are presented in Section 5.4. This is followed by detailed description of the proposed GkNN query processing system in Section 5.5. Examples of the system will be given in Section 5.6. Lastly, Section 5.8 will conclude the chapter.

5.2 System Model and Assumptions

We consider a mobile network without any central supervision or base station where query objects and their peers are dynamic. It is a symmetric system where each moving object, such as a smart mobile phone or a tablet, plays as both a query node and a peer of other nodes. Moving objects are also able to be self-aware of their location through an equipped GPS. The location of moving objects and objects of

Table 5.1: Notations used in Chapter 5

Notation	Meaning
$Q = \{q_1, ..., q_M\}$	a group of query nodes
q_{head}	the query head of Q
$P = \{p_1, ..., p_N\}$	a priority queue of peers of q_{head}
$IO = \{o_1, ..., o_S\}$	a universal set of objects of interest
IO_{p_i}	is a set of sorted objects of interest cached in p_i.
$IO_{q_{head}}$	is a set of sorted objects of interest cached in q_{head}.
r	the communication range of a moving object.

interest mentioned in this paper are actual physical location. In addition, objects of interest are distributed randomly in the network.

To enhance P2P query processing, a cache memory is assigned to store spatial data of objects of interest from its previous queries. Also, moving objects are equipped to conduct ad-hoc communication with other neighbour moving objects via via Wi-Fi, Bluetooth or ZigBee.

5.3 Query Models and Message Types

Basically the proposed system is designed to answer GkNN queries based on cached data of peers. The cached data could be the results from previous kNN or even GkNN queries stored in cache memory of peers. Unlike kNN queries, there are more than one query points in GkNN queries. We assume that the group of moving objects generating the query are peers of each other. In another way, the group of moving query points can communicate with each other to exchange data and collaboratively answer their queries.

5.4 Notations

Table 5.1 lists notations that are going to be used in this Chapter.

5.5 System Details

Our ultimate aim is to harness the collaborative power of peers in a pure P2P GkNN query processing on mobile environments in order to answer GkNN queries. The core of our system is eliminating the base station by collecting results from selected peers by time and energy effective boundary-polygon-based algorithms. Overall, the proposed system is divided into two primary phases: 1)Initialisation and Peer Discovery Phase, 2)P2P-GNN Query Processing. Each phase is described in detail as below.

5.5.1 Initialisation and Peer Discovery

Each moving object maintains a cache that holds a default map and the associated objects. In GkNN, there is more than a query point which generates a query. Let $Q = \{q_1, ..., q_M\}$ be the set of M query points. Instead of all M nodes processing the query, to save energy and communication cost, a query head is be elected to conduct the query processing. Let $RE(q_i)$ be the remaining energy of q_i. The definition of a query head is as below

Definition 5.5.1. Let q_i be an arbitrary query node in Q, q_i is called a query head, denoted as q_{head} iff $\forall q_j \in Q$, $RE(q_i) \geq RE(q_j)$.

As the criterion for electing a query head is based on the remaining energy of the moving objects, the network lifetime is maximised.

Since objects move frequently, their peers also change. As a result, before starting to send queries, the query head q_{head} needs to discover which moving objects are in its communication range by simply sending a one-hop broadcast message. After receiving the broadcast messages, each peer sends an acknowledgement message.

The query head q_{head} collects all acknowledgement messages from the surrounding nodes to make a list of its peers. More specifically, this phase is described in Algorithm 9. It is a note that, all algorithms is distributed. If the subject that

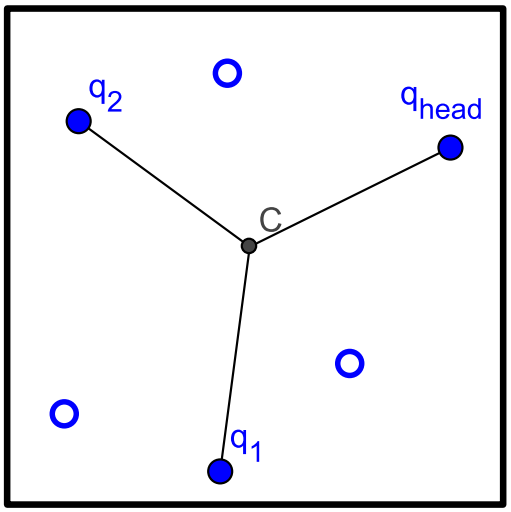


Figure 5.1: C is the centroid points of the query group $Q = \{q_1, q_2, q_{head}\}$. Other empty round symbol are also moving objects, but they do not belong to the query group.

performs methods is not specified, they are done by q_{head}. Otherwise, the subject will be indicated explicitly.

Algorithm 9: Initialisation and Peer Discovery

Data: query group Q, transmission range r
Result: head query node q_{head}, set of peers P

```
1 begin
2     Q elects q_head as the query head;
3     P = ∅;
4     q_head broadcast a one-hop beacon_msg from q_head to every neighbour MO_j
      in range r;
5     if a peer p_i receives beacon_msg then
6         p_i sends ack_msg to q_head;
7     if q_head receives an ack_msg from p_i then
8         P = P ∪ {p_i};
9 end
```

It is noted that q_{head} is assigned an acknowledgement time-out period. Therefore, q_{head} waits to receive acknowledgement messages from its peers during that period.

5.5.2 P2P-GNN Query Processing

Definition 5.5.2. In consideration of energy constraint of moving objects, the centroid point $C(x_C, y_C)$ of the set $Q = \{q_i(x_i, y_i), q_i \text{ is a query node}\}$ is approximated

by the q_{head} (see Fig. 5.1). The centroid point C is defined by the calculation of its coordinates as below:

$$x_C = \sum_{i=0}^{M} x_{q_i}/M \tag{5.1}$$

$$y_C = \sum_{i=0}^{M} y_{q_i}/M \tag{5.2}$$

Proposition 3. Let o_i be an arbitrary object of interest in IO. $\forall o_j \in IO$, if $distance(o_i, C) \leq distance(o_j, C)$ then $\sum_{i=1}^{M} distance(o_i, q_i) \leq \sum_{i=1}^{M} distance(o_j, q_i)$ or o_i is one of the GkNN of Q.

The algorithm is based on a hybrid approach inspired by multiple query method and single point method in (Papadias et al., 2004). However, instead of performing incremental NN queries for each point of Q, q_{head} takes advantage of cached data collected from its peers. The cached data actually consist of nearest neighbour answers of previous kNN queries or other GkNN query; therefore, they are the potential k nearest objects of q_{head}.

When collecting objects of interest of peers, q_{head} put them into a priority queue $IO_{q_{head}}$ so that only the good candidates are kept while the cache space is limited. Based on Proposition 3, the top-k objects of interest are the answer of the outer GkNN query. Finally, the result is sent out from q_{head} to other query nodes in Q. This stage is summarised as Algorithm 10.

Algorithm 10: P2P-GNN

Data: query group Q, query head q_{head}; set of peers P
Result: priority queue $IO_{q_{head}}$

```
1 begin
2     q_head calculates centroid point C;
3     q_head sets priority queue IO_{q_head} sorting based on distance to C;
4     q_head sends queries to set of peers P;
5     foreach IO_{p_i} received from p_i ∈ P by q_head do
6         IO_{q_head} = IO_{q_head} ⋃ IO_{p_i};
7     foreach q_i ∈ Q \ {q_head} do
8         q_head sends out the first k objects in IO_{q_head} to q_i;
9 end
```

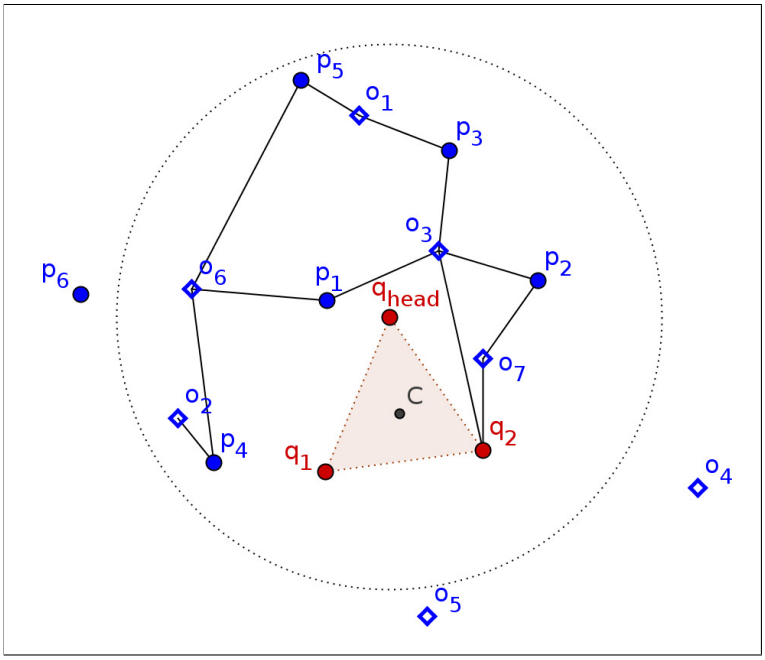


Figure 5.2: An example of GkNN query. The query group consists of q_1, q_2 and q_3. Here q_1 is elected and represented as q_{head}. p_1,...p_6 are mobile peers. The circle is the communication range of q_{head}. The links show which peers cache which objects of interest.

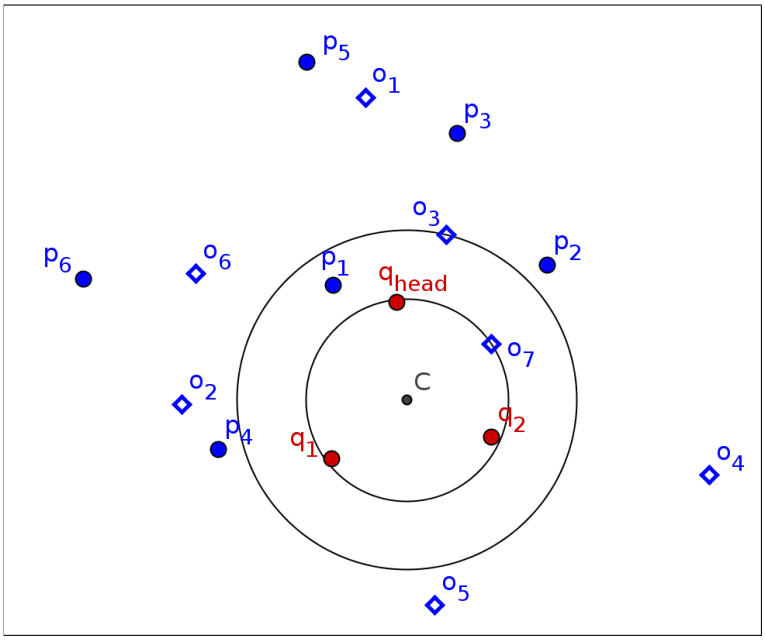


Figure 5.3: An example of GkNN query. o_3 and o_7 are two nearest neighbours of the group of q_{head}, q_1 and q_2.

5.6 An Example

Let us look at an example that illustrates how our algorithms process a GkNN query in Fig. 5.2. A group of mobile peers, $Q = \{q_1, q_2, q_3\}$, are represented a triangle.

q_1, q_2 and q_3 are interested in two common meeting points so that the total travel distances to each peer of the group is the least ($k = 2$).

According to Algorithm 9, q_1 is elected and represented as q_{head}. After that q_{head} broadcasts a beacon message to discover its peers. In this example, except p_6, all other mobile peers (q_1, q_2, p_1, p_2,...,p_5)) in Fig. 5.2 join the query processing and each sends an acknowledgement message to q_{head}. Their location data will be put into a priority queue at q_{head}.

According to Algorithm 10, the centroid point C is calculated based on the location of the query peers (q_1, q_2 and q_{head}). The calculation of coordinates of C is based on Equation 5.1 and Equation 5.2. The links represent caching relationships between mobile peers and objects of interest. For example, q_2 caches location of o_3 and o_7; p_1 caches location of o_3 and o_6; q_1 does not have any cache data.

Then q_{head} sends the G2NN query to its peers (q_1, q_2, p_1, p_2,...,p_5) and collects objects of interest cached in peers and put them into a priority queue that is sorted by the distance to the centroid point C. Finally, the first 2 objects, o_7 and o_3, in the queue will be the answer of this example (Fig 5.3). The smallest circle goes through o_7, the nearest objects to the query group Q. Similarly, the second circle passes o_3, the second nearest objects to Q.

5.7 Discussion and Extension

If we look at GkNN from a different perspective, GkNN can be considered to kNN problem after selecting the centroid point and query head q_{head}. Here the query point is not at the query object's location, but at the centroid point. Then q_{head} can perform single validation and mutual validation as Algorithm 2 and Algorithm 3 in Chapter 3. More specifically, q_{head} identifies which peers to contact and puts them in a priority queue, q_{head} collects objects of interest from its nearest peers, and performs single validation as as Algorithm 2. If the number of verified objects is less than k, it continues forwarding the query to the next peers and performs Algorithm 2 as well as Algorithm 3. Just a note that the validation will be processed against the centroid

Algorithm 11: GkNN_single_validate

Data: query node q; int k; peer p_i; priority queue IO_{p_i}, Centroid point C;
Result: priority queue VO, IVO; boolean isAnswered

```
1  begin
2      answered = false;
3      foreach o_i ∈ IO_{p_i} do
4          if dis_E(C, o_i) + dis_E(C, p_i) ≤ dis_E(p_i, o_k) and o_i ∉ VO then
5              Add o_i into VO;
6              if |VO| = k then
7                  isAnswered = true;
8                  return;
9          else
10             if |IVO| < k and o_i ∉ IVO then
11                 Add o_i into IVO;
12             else if IOV ∋ o_k : dis_E(o_k, C) > dis_E(o_i, C) then
13                 Replace o_k with o_i in IOV;
14 end
```

points other than the location of q_{head}. Therefore, Algorithm 2 and Algorithm 3 can be rewritten as Algorithm 11 and Algorithm 12, respectively. The process repeats until it gets enough k nearest neighbours.

5.8 Chapter Summary

In this chapter, the pure P2P query processing system in Mobile Networks to solve GkNN queries is presented. In summary, there are two stages involved in query processing, which are 1) Initialisation and Peer Discovery and 2) P2P-GNN Query Processing. In Initialisation and Peer Discovery, the group of query objects elects a query head q_{head} and identifies which peers to query. The P2P-GNN Query Processing stage harnesses collaboration of peers answer queries without any support from server side. As a consequence, we eliminate limitations of centralised approaches, such as single point of failure and bottleneck problems.

The simulation design and result will be discussed in detail in Chapter 6. Overall, the simulation result shows that our P2P-GkNN algorithm significantly saves communication cost via the short-range network compared to the centralised algorithm

Algorithm 12: GkNN_mutual_validate

Data: query node q; int k; peer p_i; priority queue P, UVO Centroid point C;
Result: priority queue VO, IVO; boolean isAnswered

```
1  begin
2      isAnswered = false;
3      foreach o_i ∈ UVO do
4          isFulFilled = true;
5          isChecked = false;
6          Find set N of 8 even distributed points on circle C(C, dis_E(C, o_i));
7          foreach n_h ∈ N do
8              foreach p_j ∈ P do
9                  r = dis_E(p_j, o_{k,j}), o_{k,j} is the k^th object of interest in IO_{p_j};
10                 if dis_E(p_j, n_h) <= r then
11                     isChecked = true;
12             if !isChecked then
13                 isFulFilled = false;
14                 break;
15             else
16                 isChecked = false;
17         if isFulFilled and o_i ∉ VO then
18             Add o_i into VO;
19             Remove o_i from IVO;
20             if |VO| = k then
21                 isAnswered = true;
22                 return;
23 end
```

regardless to the change of network parameters. In addition the mean latency of query processing in our proposed algorithm is approximately 32% less than the other benchmarking system while maintaining a high accuracy rate of more than 82%. In general, the P2P system provides a practically feasible option to solve GkNN queries in a large-scale and busy network.

"Science replaces private prejudice with public, verifiable evidence."

Richard Dawkins

6

Performance Evaluation

6.1 Overview

This chapter will first present simulation tools which are used to evaluate this research in Section 6.2. Second, Section 6.3 will show how the simulation model is built up to evaluate the performance. Then, the cost analysis and simulation results of all of the proposed algorithms (kNNs, RNNs and GkNNs) will carefully discussed in Section 6.4, Section 6.5 and Section 6.6, respectively. Finally, Section 6.7 will conclude the chapter.

6.2 Simulation Tools

The simulation tools ns-2 and OMNet++ are the most popular used in wireless networks (ns2 Community, n.d.; OMNeT++, n.d.). In term of performance comparison, OMNeT++ is preferred although it requires more effort to create a simulation. OMNeT++, running on both Unix and Windows, supplies modular based C++ library and framework to build our systems. Plus, it provides open programming interface and a user-friendly graphical interface while ns-2 has some restriction(Albeseder, n.d.). Therefore, following the growth of simulations using OMNeT++, we decided to choose OMNeT++ as our simulation environment.

Even though OMNeT++ is often regarded as a network simulator, it is actually not. Only basic machinery and tools to write simulation are provided whereas direct support and a concise modelling chain for wireless communication are not supported. Therefore, we need MiXiM, an extended simulation framework for wireless and mobile networks based on OMNeT++ simulation engine.

As the simulation is built on top of OMNeT++ and MiXiM, a brief description of them is presented in the following sections.

6.2.1 OMNeT++

OMNet++ (Objective Modular Network Test-bed in C++), originally developed by András Varga in 1997, is a powerful discrete event simulation environment (Varga, 2001). This free and open source simulation tool provides a C++ simulation library and framework for a variety of networks, including wired and wireless communication networks, to name just a few. Based on the core framework of OMNeT++, there are many extended model frameworks for wireless ad-hoc networks, sensor networks and Internet protocols. The Integrated Development Environment (IDE) of OMNeT++ is based on the Eclipse platform, which serves as a host for many different plug-ins. There are also many extensions to support alternative programming languages such as Java or C#, real-time simulation, etc,...

OMNet++ can run in multiple platforms such as Windows, Mac OS X and Linux. Thanks to its extensible, flexible architecture, OMNet++ has been building up a large user community. The latest version, OMNeT++ 4.3.1, was released on 17th, September, 2013.

OMNeT++ is an modular, component-based, simulation framework. In other words, an OMNeT++ model is composed of two type of modules: simple modules and compound modules. The simple modules are active modules written in C++, using the simulation class library, whereas compound modules are groups of simple modules. Below are some of the modules in OMNeT++ (Varga and Hornig, 2008):

- simulation kernel library;
- NED topology description language;
- OMNeT++ IDE based on the Eclipse platform;
- GUI for simulation execution, links into simulation executable (Tkenv);
- command-line user interface for simulation execution (Cmdenv);
- utilities (makefile creation tool, etc.);
- documentation, sample simulations, etc.

6.2.2 MiXiM

MiXiM (mixed simulator) is a model framework that integrates and extends existing simulation frameworks developed for wireless and mobile simulations in OMNeT++ (Köpke et al., 2008). In particular, MiXim combines the mobility support, connection management, and general structure from the Mobility Framework (MF, n.d.), the radio propagation models from the Channel Simulator (Valentin, 2006). In addition, the protocol library in MiXiM comes from the MAC simulator (MAC, n.d.), the Positif framework (Positif, n.d.), and from the Mobility Framework. In addition, it provides a user-friendly graphical design of wireless and mobile networks in OMNeT++, supporting debugging and defining complex scenarios.

MiXiM logically consists of 2 main parts: the base framework and the protocol library. General functionality are defined in the former part. They are including: Connection management, Mobility and Wireless Channel.

The protocol library is a set of standard protocols, such as mobility models, that can adapt on top of the base framework.

The next section will show in details how my research employs these tools to design and implement the simulation model used in this research.

6.3 Simulation Model

As mentioned in the previous section, MiXiM is used in my simulation environment as a powerful OMNeT++-based framework to model and analyse our Mobile P2P Query Processing System. First, we go through assumptions used in the simulation model before going into details of each implemented modules.

6.3.1 Assumptions

At the initial stage, each moving peer is assigned kNN objects of interest in its cache. The nature of the cache is a priory queue based on the distance from the cached objects of interest to the moving peer. When the cache is full, then new objects of interest will be cached if the distance to the moving peer is less than any that of another object in the cache.

6.3.2 Benchmarks

In order to assess the performance of our system against other benchmarking systems, we model the centralised system (Benetis et al., 2002) and the hybrid system (Ku and Zimmermann, 2008). In those systems, a BS with an R*-Tree (Beckmann et al., 1990) is implemented at the server side. Communication to the BS is conducted via 3G(WCDMA), band I-2100 which is used by Vodaphone and Optus in Australia.

Table 6.1: Range of parameter value

Parameters	Value
No. of peers	3800, 7600, 11400, 15200, 76000
Expected number of queries, λ	1, 2, 3, 4, 5
Speed (mps)	1, 10, 20, 30, 40

Data rate for high speed moving peers in this network is 128kbps (Silicon-Press, 2011) and current consumption in connected state is 365.6mW (Option®, 2008).

6.3.3 Data sets and Parameters

The simulation is run with a real dataset for a large-scale network as Table 6.2 and Fig. 6.1. More specifically, I collected statical data of inner Melbourne city, Australia in 2009 (ABS, 2009). The number of objects of interest is the number of tourist accommodation establishments. The number of moving peers is based on the number of registered vehicles. We expect that at least 10% of people who registered their vehicle joins query processing in vehicle networks (Delot, Ilarri, Cenerario and Hien, 2011; Delot, Mitton, S. and Hien, 2011). Each moving peer is equipped with a cache memory whose capacity is enough to store location and type of 50 objects of interest.

The effect of each parameter is investigated by varying its value in a range as Table 6.1. Each moving object is equipped with a cache memory whose capacity is enough to store location and type of 50 objects of interest.

For comparison purpose, the Centralised search algorithm is built based on an R*-tree BS (Korn and Muthukrishnan, 2000). *libspatialindex* framework is used support spatial indexing methods and basic RNN and kNN queries. (Hadjieleftheriou, 2013).

6.3.4 Performance Metrics

Several metrics need to be measured in order to compare the performance of the proposed algorithms with the benchmarking such as: mean latency of query processing time, power consumption, computation cost, false hits (or the accuracy rate of the system), etc. Here the communication cost and mean latency are the average energy

Table 6.2: Simulation parameters

Parameters	Value
Playground	87.2kmx87.2km
No. of objects of interest	550
No. of moving peers	15200
k	10
Expected number of queries generated by each moving peer	5
Cache Size	50
Simulation time	600s

and time to process a query in the network, respectively. Power consumption is measured by Battery and BatteryStats modules which are implemented in each moving peer. In addition, considering that the central approach has the full picture of the map and know the exact location of moving peers, its accuracy rate is 100%. Therefore, we focus on examining the accuracy rate of P2P search algorithms by measuring how many percent their answer matches with the expected answer from the centralised search algorithm. We can vary the value of parameters in Table 6.2 such as the number of objects of interest, the number of moving peers, k, communication range, etc to observe how the performance metrics change.

6.3.5 Network Definitions

P2PQueryNet

This section will describe my simulation design and implementation in detail. As OMNeT++ is a modular based simulation tool, the whole network of our simulation, called P2PQueryNet, is composed of a number of modules. The simulation is designed on a limited area, a playground, on which peers and objects of interest are placed. Fig. 6.2 shows the design view of our network framework. The source file is attached in Appendix A.1.

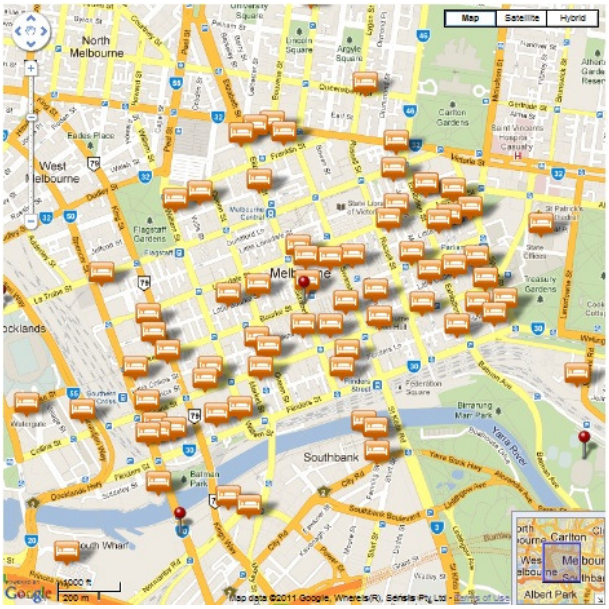


Figure 6.1: A Part of Tourist Accommodation Map in Melbourne (Melbourne-Hotel-Map, 2011)

In general, there are five modules in the network as below.

1. *World utility module (world):* provide utility methods and information to share among modules in the whole network such as the dimensions of the network (playground), the number of moving peers and the number of objects of interest. Details of code are presented in Appendix A.2.

2. *Connection Manager module (connectionManager):* manages connection between interfering peers. The connection is established when one peer is within the maximal interference distance of another and tearing down the connection once they exceed this distance. Please refer to Appendix A.3 for more detail.

3. *Objects of interest*: the network consists of a collection of static objects of interest. The number of objects is based on the number of tourist accommodation in Melbourne as said in Section 6.3.3. The detail of the implementation will be described in the section below.

4. *Moving peers*: Apart from objects of interest, moving peers are the core of the whole network. They are stored in a list of moving peers. In the simulation

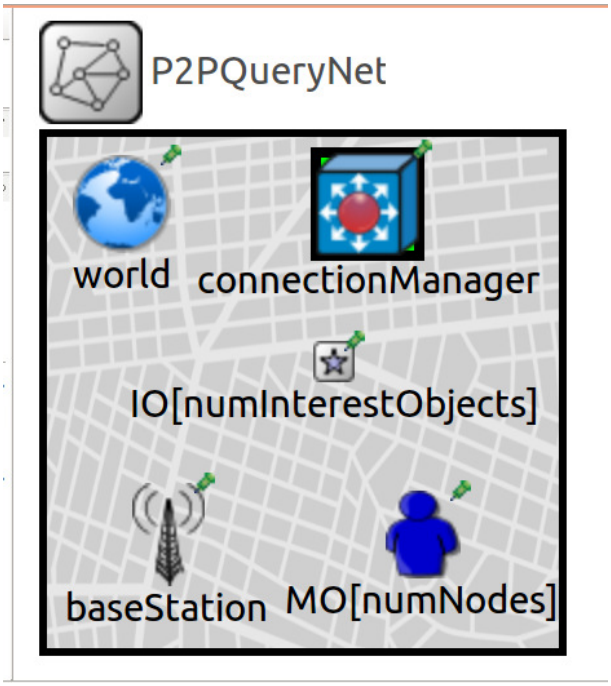


Figure 6.2: Five modules in the network in the design view of P2P Query Networks. The base station is implemented only for the purpose of comparison in Centralised and Hybrid Systems.

the connection manager will provide the information which peers of a moving object can exchange data.

5. *Base station*: The base station is actually implemented only for the purpose of comparison. It is activated and used only in Centralised or Hybrid System.

Objects of Interest

The list of objects of interest are represented as blue stars in Fig. 6.2. The implementation of these objects is a *simple module* in Appendix A.4. A simple module is an active element in the network model. It does not consist any sub-modules. In OMNet++, each module is implemented by a C++ class using OMNeT++ simulation class library. The simulation kernel regards any events as messages. The corresponding class for Objects of Interest is inherited from *cSimpleModule* as in Appendix A.4.

Moving Peers

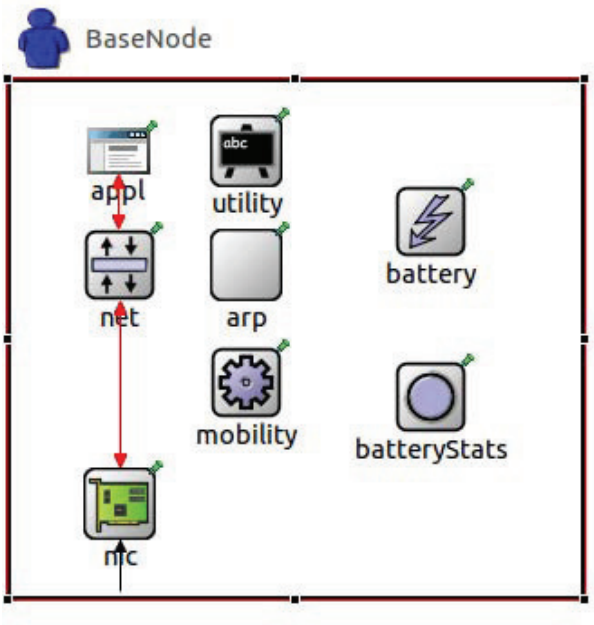


Figure 6.3: Moving objects' modules

Moving peers are implemented by a *Compound Module*, called *BaseNode*. Each peer contains 8 sub-modules as Fig. 6.3. Functions of each modules is shown as Table 6.3.

Moving pattern peers and speed of are implemented by *mobility* module. In this simulation model, moving peers move linearly in random directions. The speed of peers is passed to the *mobility* module from *omnet.ini* to evaluate its impact on the query processing performance.

As can be seen in Fig. 6.3, there are connections between *appl* module and *net* module, *net* layer and *nic* layer, *nic* module and other modules that do not belong to the *BaseNode* module. Basically, an input is linked to an output gate with a connection. Connections can be within a compound module, corresponding gates of two sub-modules such as *appl* and *net*; *net* and *nic*, or a gate of one sub-module and a gate of the compound module like from *appl* to *radioIn*. The implementation of the connections in *BaseNode.ned* is as shown in the snippet below:

```
1 //...
2         nic.upperGateOut --> net.lowerGateIn;
3         nic.upperGateIn <-- net.lowerGateOut;
```

Techniques

Table 6.3: Modules and their function in MiXiM

Module name	Functions
appl	application layer
net	network layer
nic	network interface card consists of MAC layer and physical layer
utility	collecting statistical data and maintaining parameters among modules
arp	address resolution protocol
mobility	responsible for the movements of a node or an object
battery	measuring battery issues (Feeney and Willkomm, 2010)
batteryStats	collecting statistical data from the battery module

```
4        nic.upperControlOut --> {@display("ls=red;m=m,70,0,70,0");}
5                             --> net.lowerControlIn;
6        nic.upperControlIn <-- {@display("ls=red;m=m,70,0,70,0");}
7                            <-- net.lowerControlOut;
8
9        net.upperGateOut --> appl.lowerGateIn;
10       net.upperGateIn <-- appl.lowerGateOut;
11       net.upperControlOut --> {@display("ls=red;m=m,70,0,70,0");}
12                            --> appl.lowerControlIn;
13       net.upperControlIn <-- {@display("ls=red;m=m,70,0,70,0");}
14                           <-- appl.lowerControlOut;
15       radioIn --> appl.radioIn;
```

The further detail of network definition of Moving Peers, or *BaseNode.ned* is presented in Appendix A.5. The next section, the implementation of sub-modules in Moving Peers will be elaborated.

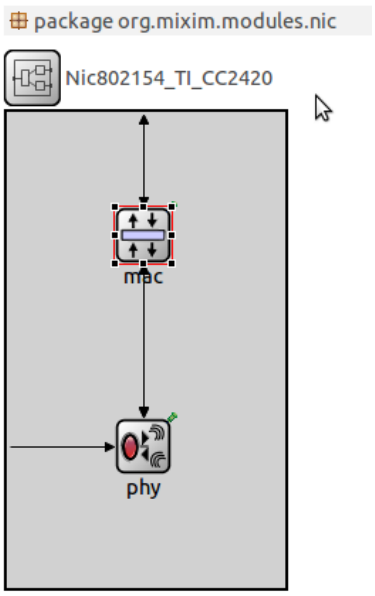


Figure 6.4: Nic module includes two submodules: mac and phy.

Nic (Physical layer and MAC layer)

In this simulation model, simulation uses Nic802154_TI_CC2420, a network interface card which follows IEEE 802.15.4 standard for Low-Rate Wireless Local and Metropolitan Networks (LR-WPANs) such as ZigBee, Wireless HART and MiWi P2P. According to the configuration for the network interface cards in MiXiM, transmission current txCurrent=17.4mW and receiving current rxCurrent=11mW.

6.3.6 Message Handling

All modules in OMNeT++ communicate with each other by messages. Messages are implemented as objects which may hold any data structures and other objects. An example of a query, including $< k$, type of queried objects, network address and position of the source peer is defined as below:

```
1 cplusplus {{
2 #include "ApplPkt_m.h"
3 #include "Coord.h"
4 }}
5 packet ApplPkt;
```

```
6  class Coord;
7  //
8  // TODO generated message class
9  //
10 packet Query extends ApplPkt {
11     int k; //number of objects of interest needed querying
12     char objectType;
13     long srcNetworkAddress;
14     Coord srcPosition;
15 }
```

In this simulation models, 4 different types of messages below are defined to be generated in *appl* of moving peers. The detail of implementation in Appendix B

1. *Beacon message* (*beacon_msg*) is broadcast from the query node to detect peers in its communication range. The implementation is presented in Appendix B.1.

2. *Acknowledgement message* (*ack_msg*) from peers to the query node is attached location of peers. The implementation is presented in Appendix B.2.

3. *Query message* (*query_msg*) from the query node to the selected peers to ask for objects of interest cached in those peers. The implementation is presented in Appendix B.3.

4. *Query reply* (*reply_msg*) from peers to the query node attaches a possible answer from the cache of the peer. The answer consists of the location and type of objects of interest. The implementation is presented in Appendix B.4.

Generating queries is based on Poisson arrival model with parameter λ. The universal λ is assigned to all moving peers as an average number of queries coming per unit time; or expected number of queries generated by each moving peer is $E(N) = \lambda T$ where T is the simulation time.

Here we will discuss about two types of message handling that are used in OMNet++ in general and in *appl* my simulation in particular.

Handling messages between modules

When one peer (represented by *BaseNode* module) wants to send a query to another, all 4 types of messages will be generated in application layer, then go through lower layers including *net* and *nic* before reaching another peer. When the other peer receives the message, the message will be transfer from *nic* to *net* and then *appl*. The framework of message handling from another sub-module through lower gate of *appl* module is as below:

```
1  void MyApp::handleLowerMsg( cMessage* msg )
2  {
3      switch( msg->getKind() ){
4      case BEACON_MESSAGE: //for peer nodes
5      //... beacon message processing
6          break;
7      case BEACON_REPLY:
8      //... acknowledgement message processing
9          break;
10     case QUERY_MESSAGE: //for peer nodes
11     //... query message processing
12         break
13     case QUERY_REPLY: // for query nodes
14     //... query reply processing
15         break;
16
17     default:
18             EV <<"Error! got packet with unknown kind: "
19              << msg->getKind()<<endl;
```

```
20          delete msg;
21      }
22  }
```

Handling Self-messages

Apart from messages from other modules, simple modules such as *appl* also handle its own messages. For example, when *appl* receive a QUERY_TIMER message to revoke a beacon message to discover peers. After the waiting time is expired, an EXPIRATION_MESSAGE will be generated. When a peer receives its own EXPIRATION_MESSAGE , it can send a query. The framework of self-message handling is as below.

```
1  void MyApp::handleSelfMsg(cMessage *msg) {
2      switch( msg->getKind() ){
3      case QUERY_TIMER:
4        //...
5                  break;
6
7      case EXPIRATION_TIMER:
8        //...
9              break;
10     default:
11             EV << "Unknown selfmessage! -> delete, kind: "
12             <<msg->getKind() <<endl;
13             delete msg;
14     }
15 }
```

After presenting how I build up the simulation model, the next sections will be the discussion for the simulation results for each proposed algorithm.

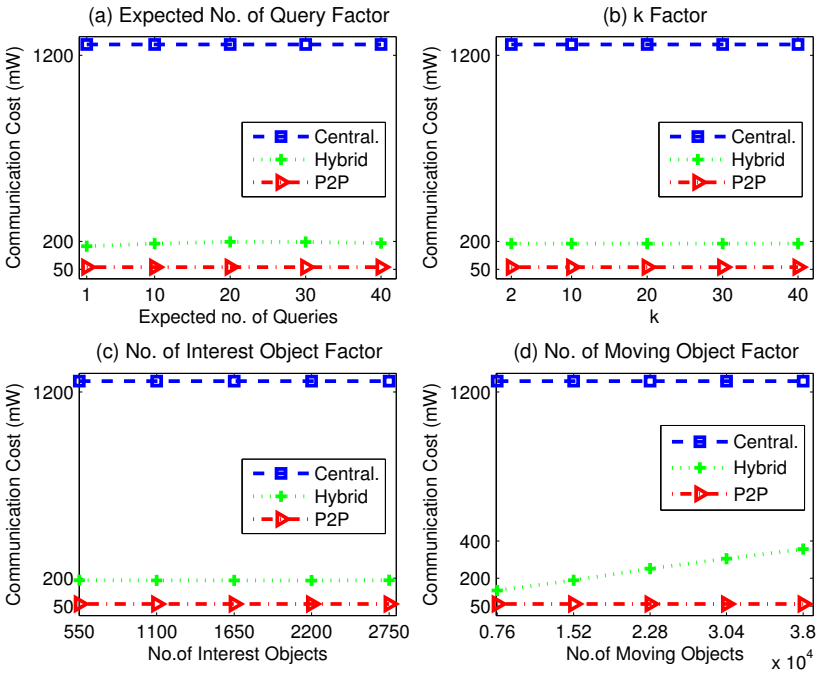


Figure 6.5: Communication Cost among Centralized, Hybrid and P2P Systems

6.4 Simulation Results for kNN

In this section we discuss differences between the simulation results for centralised, hybrid and P2P systems in two contexts: communication cost and mean latency of query processing time. Finally, accuracy rate of the proposed system are also taken into evaluation with both high and low speed of moving peers. Finally, we measure the percentage of query resolved with high and low density of moving peers. In each case, the number of expected queries, k, number of objects of interest and moving peers are varied to evaluate the importance of each factor.

6.4.1 Communication cost

Fig. 6.5 compares the communication cost among centralised, hybrid and P2P systems.This cost is calculated as the average power consumption that a moving peer needs to answer a query. In general, the result shows P2P system outperforms roughly as at least 6 times as the rest. In addition, the communication cost of P2P is stable no matter how the factors change. Although the communication cost of

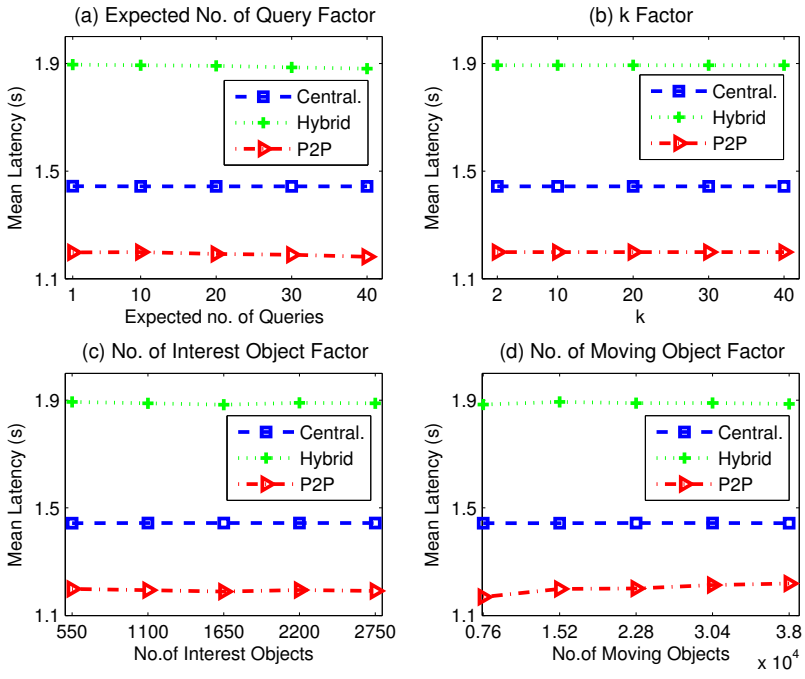


Figure 6.6: Mean latency of query processing among Centralized, Hybrid and P2P Systems

hybrid system is less than that of centralised system, it is higher as six times as that of P2P system since it still needs to exchange messages with the BS. Also it increases when the number of moving object increases as an accumulation of spending energy for communicating to both peers and the BS.

Mean latency

The comparison of mean latency of query processing among centralised, hybrid and P2P systems is further investigated and shown in Fig. 6.6. Mean latency is measured by the average time required for a moving peer to solve a kNN query. In all cases P2P approach can save approximately 17% and 36% query processing time compared with hybrid and centralised approaches respectively. The mean latency of the P2P approach slightly increases when the number of moving peers rises due to the need to communicate with more peers to answer queries.

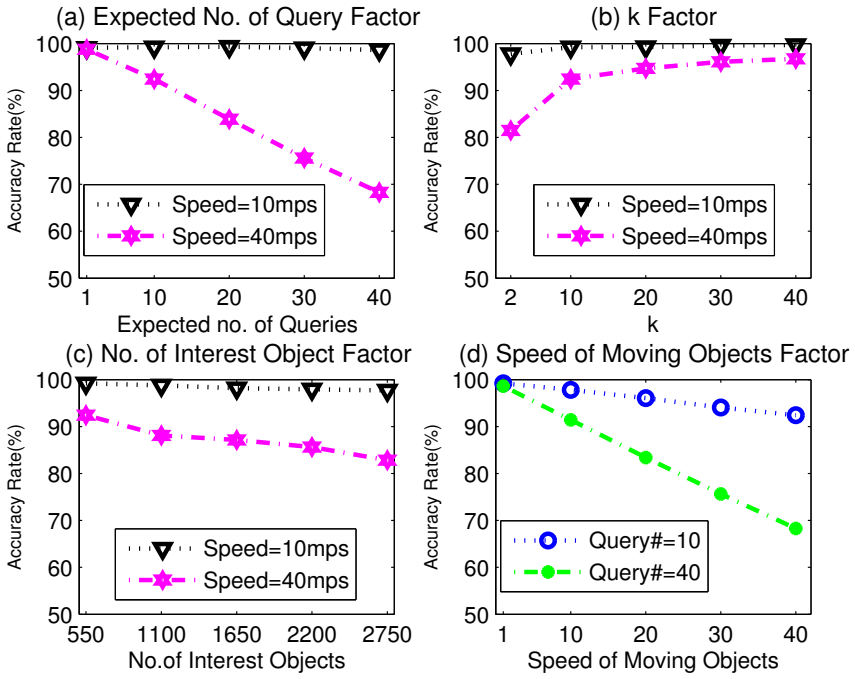


Figure 6.7: Accuracy rate of P2P System

6.4.2 Accuracy Rate of P2P System

We also evaluated the accuracy rate of the proposed system by measuring the matching rate of kNNs results from our P2P systems with the centralised methods. The results are shown in Fig. 6.7. Accordingly, in the environment that the mobile objects move too fast at the speed of 40 meters per second(mps), the accuracy rate falls gradually when expected number of queries generated from each moving peer increases or the density of objects of interest is higher. However, when the number of k goes up, the matching number of kNN improves remarkably. More importantly, the accuracy rate maintains quite good (approximately 98%) in the environment of average speed moving peers (10mps) no matter how λ, k or the number of objects of interest is. From a closer look at Fig. 6.7(d), the decrease is only gradual when the expected number of queries to each moving peers (λ) is low (10 queries generated by each moving peer).

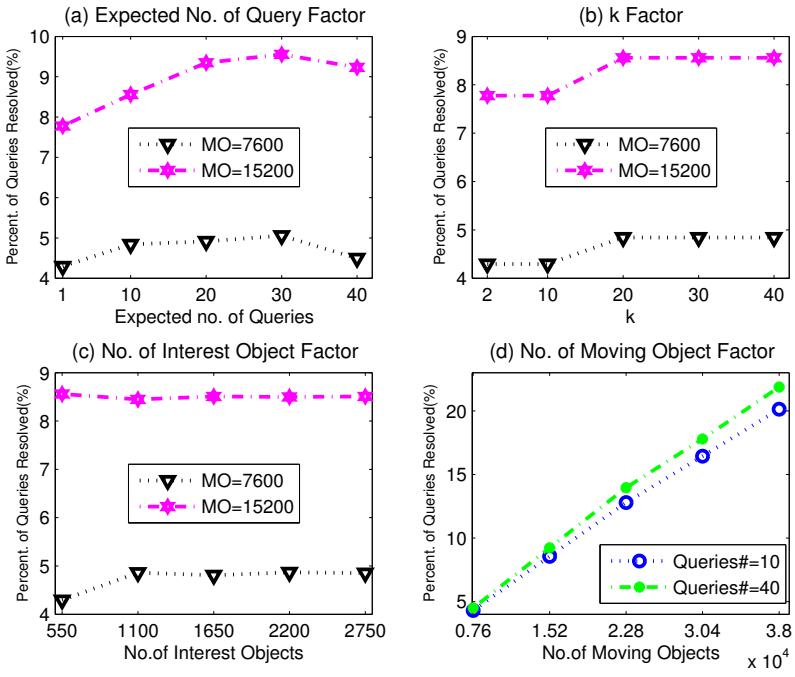


Figure 6.8: The Percentage of Queries Resolved of P2P System

6.4.3 The Percentage of Queries Resolved of P2P System

Fig. 6.8 shows the percentage of queries resolved in low and high density of moving peers by the proposed P2P system. It is measured by the number of queries that are fully answered only by the collaboration of peers out of the total number of queries sent by moving peers. When the number of generated queries goes up, the resolving percentage also increases until this number reaching 30 in 600s as Fig. 6.8(a). Fig. 6.8(b)(c) reflects that when k is larger than 20 or the number of objects of interest is more than 1100, the resolving percentage becomes stable providing that k is less than cache size of moving peers. The results from all sub-graphs clearly show that the higher density of moving peers is, the higher the percentage of queries to be answered by peers is. More specifically, this rising up is a linear line as shown in Fig. 6.8(d).

6.4.4 Discussion and Future Work

In this chapter, we presented a pure P2P query processing system in Mobile Ad-hoc Network. Based on novel gathering and validating algorithms, the system

harnesses collaboration of peers to answer queries without any support from server side. In this way, we eliminate the limitation of centralised approaches such as single point of failure and the bottleneck problem. The simulation result shows our query processing system saves more than 6 times the communication cost via the short-range network compared to both centralised and hybrid system. In addition the mean latency of query processing in our proposed system is up to 36% less than the other benchmarking systems. Also the P2P is dependent on the density of moving peers so it performs much better in an environment with high density of moving peers.

The research opens several new directions in our future work. One of the potential directions is improving the number of queries resolved by accepting approximate results or storing local map at each moving peer. We also plan to expand our work to deal with a wide variety of kNN queries.

6.5 Cost Analysis and Simulation Results for Reverse Nearest Neighbours

6.5.1 Overhead Cost Analysis

This section first analyses the overhead cost including computation cost and communication cost; and compares the performance of the P2P search algorithms with the Centralised Search Algorithm through simulation results with respect to a variety of parameters as Table 6.1. Subsequently, the effect of each factor in Table 6.1 is carefully evaluated for three proposed P2P search algorithms (BFA, RBA, and TBA) by 3 evaluation metrics: mean latency, accuracy rate and finally the number of peers pruned.

Here the communication cost and mean latency are the average energy and time to process a query in the network, respectively. In addition, considering that the central approach has the full picture of the map and know the exact location of

moving peers, its accuracy rate is 100%. Therefore, we focus on examining the accuracy of BFA, RBA and TBA by measuring how many percent of RNN answer match with the expected answer from Centralised Search Algorithm.

Computation Cost

Centralised Search Algorithm: As the complexity of Centralised search algorithm is similar to kNN search, the node accesses for kNN is estimated as follow (Tao, Zhang, Papadias and Mamoulis, 2004):

$$NA(k) = \sum_{i=0}^{\log_f \frac{N_{IO}}{f}} \left\lceil \frac{N_{IO}}{f_{i+1}} \right\rceil . \left(\frac{L_i - (L_i/2 + s_i/2)^2}{1 - s_i} \right)^2 \tag{6.1}$$

where N_{IO} is the number of objects of interests, f is the average node fanout, L_i is the length of each dimension of rectangle $R(q, L_i)$ such that the volume of $R(q, L_i)$ equals to that of the centroid $\Theta(q, r)$ and s_i the extent of a level-i node. Both s_i and L_i are estimated by a function of N_{IO}, f and k as in (Tao, Zhang, Papadias and Mamoulis, 2004).

BFA: Since P2P search algorithms do not maintain a tree to process queries, the efficiency in term of computation cost is evaluated by number peers involved solving a query (NP). As peers are assumed to be uniformly distributed, the calculation of NP is summarised as below:

$$NP = 2\pi R^2 (\frac{N_{MO}}{A}) \tag{6.2}$$

where R is transmission range of each moving object, A is the network area and N_{MO} is the number of peers in the network.

RBA: According to Algorithm 6, peers engaging in query processing are restricted in the area $C(q, R) \cap C(q, 2 * dis(q, v_{max})$ where v_{max} is the farthest vertex in the tight polygon B. Hence,

$$NP = 2\pi r^2 (\frac{N_{MO}}{A}) \tag{6.3}$$

where $r = min(R, 2 * dis(q, v_{max}))$

TBA: The peers selected in this algorithm are located in a tighter boundary: $TB = C(q, r) \cap (\bigcup_{v_i \in V} C(v_i, dis(v_i, q)))$ where V is the set of vertices in polygon B. It is derived to:

$$NP = A'(\frac{N_{MO}}{A}) \tag{6.4}$$

where A' is the area of TB.

According to the Equation 6.2, 6.3 and 6.3, the computation cost of TBA is the least due to the area $A' \leq 2\pi r^2 \leq 2\pi R^2$. As TBA provides each query a tight boundary polygon that can filter more peers that join query processing. Therefore, it reduces waiting time for responses from peers and processing time for the results from peers. Vice versa, BFA does not maintain a boundary to filter peers except its communication range, which results in the least efficient in computation cost.

In addition, in all P2P search algorithms, not all peers of the query peer q process the query. Instead, there are a significant number of peers filtered from query processing thanks to time-out mechanism. After a threshold of time, q will stop sending and receiving messages from peer and start processing received data in combination with its own cached data. The number of peers pruned by each algorithms is presented in simulation results below.

Communication Cost

Centralised Search Algorithm: Communication to the BS is conducted via 3G(WCDMA), band I-2100 which is used by Vodaphone and Optus in Australia. Data rate for high speed moving objects in this network is 128kbps(Silicon-Press, 2011) and current consumption in connected state is 365.6mW(Option®, 2008).

P2P Search Algorithms(BFA, RBA and TBA): As mentioned in simulation model, each moving object is implemented a network interface card, Nic802154_TI_CC2420, which follows IEEE 802.15.4 standard. According to the configuration for the network interface cards in MiXiM, transmission current txCurrent=17.4mW and receiving current rxCurrent=18.8mW.

6.5.2 Simulation Results

P2P Search Algorithms versus Centralised Search Algorithm

Communication Cost: Fig. 6.9a shows the average communication cost for each query processing by Centralised and P2P search algorithms when the expected number of queries vary (λ). Apparently, P2P Search Algorithms (BFA, RBA and TBA) are consistently and substantially more energy-efficient than Centralised Search Algorithm in a large-scale network. Even though in P2P networks, a peer needs to communicate with many peers to gather data, it only requires is just short-range communication while Centralised Search Algorithm needs to be connected to wide-range communication such as 3G, which consumes more than 20 times of energy compared to short range communication. In addition, there are significant number of peers pruned by time-out and pruning mechanism in BFA, RBA and TBA as in Fig. 6.9c. Here we do not measure the number of peers pruned by the Centralised Search Algorithm since there is no peer of query object involved in the query processing. In BFA, RBA and TBA, the number of peers pruned actually increase when there are more queries generated. It reflects the feasibility of P2P search algorithms in busy networks.

Mean Latency: Fig. 6.9b compares mean latency that requires P2P Search Algorithms and Centralised Search Algorithm to process a query. In general, P2P Search Algorithms can save up to 43% time in processing a query compare to Centralised Search Algorithm. The main reason is in P2P Search Algorithms, query objects do not need to wait for the far BS to answer. Instead, close-by peers can provide quick reply to the query. Also, the time-out mechanism (in all BFA, RBA and TBA); the stop condition (in RBA) and filter conditions in RBA and TBA) reduce the number of peers involved in query processing. As a result, mean latency of query processing time has remarkably been decreased. Another observation from Fig. 6.9b is the mean latency of P2P Search Algorithms slightly go up when more queries are generated. This can be caused by the fact that more traffic in the networks leads to more waiting time for the reply from peers.

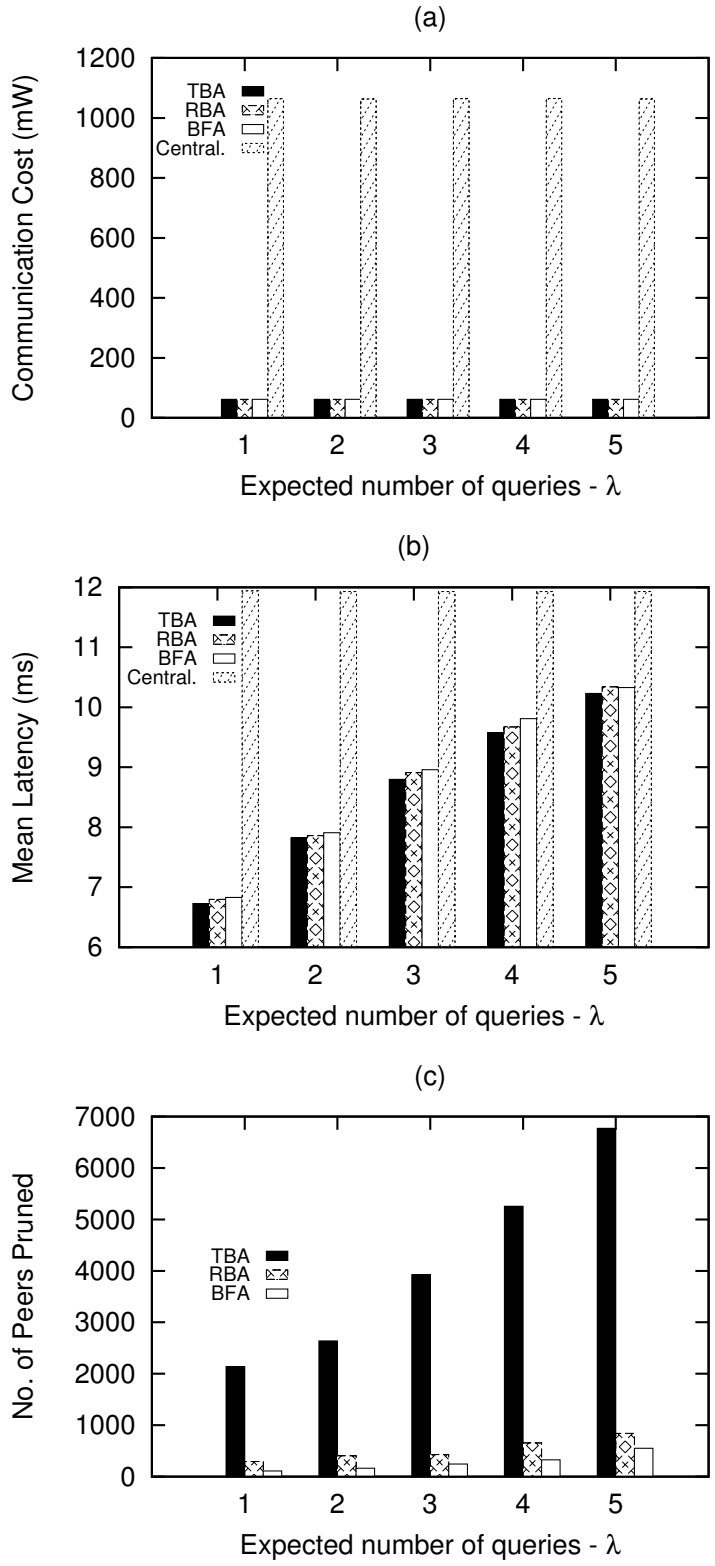


Figure 6.9: Communication Cost and Mean Latency of P2P Search Algorithms (BFA, RBA, BFA) versus Centralised Search Algorithm. (a)Communication cost. (b)Mean Latency. (c) Number of Peers Pruned by the Algorithms.

BFA, RFA and TBA

In this section, the effects of all of the input factors, which are the expected number of queries from each peers λ, the number of objects of interest, the speed of peers and the number of moving objects, are taken into consideration to evaluate the performance of three P2P search algorithms: BFA, RFA and TBA.

Effect of expected number of queries from each peers (λ). Fig. 6.10 compares performance of P2P Search Algorithms with respect to the expected number of queries generated by each peer, λ. In general, mean latency gradually increases when the number of queries increase even though it still far below that of compare to Centralised Search Algorithm. Moreover, TBA is clearly proved to be the most time-efficient while BFA spends the most time to process a query in Fig. 6.10a. The main reason is a significant number of unnecessary peers are filtered by the stop condition (Lemma 2) and the filter condition (Lemma 3) in RBA and TBA as shown in Fig. 6.10c, which results in shortening query processing time. Moreover, the number of peers pruned goes up when the number of queries is increased, which means this algorithm can apply for busy network with a huge number of communications.

Regarding to accuracy rate, all P2P algorithms show the percentage of accuracy fluctuate at high rate of 96%. Sometimes the accuracy rate of BFA is better the other since it can reach more peers. As a result, it may retrieve more data to process queries. However, all algorithms are bounded by a time-out mechanism. Therefore, within the same period of time, TBA shows that it can select better peers to ask. This leads to the fact that TBA is the most steadily high compared to the rest no mater how the value of λ changes. Even if RBA shows its accuracy rate is greater than 95.8%, it cannot compete with the other P2P search algorithms. The main reason is its stop condition is not enough to effectively filter the peers. Hence a combination between the stop condition (Lemma 2) and filter condition (Lemma 3) in TBA is the better solution. Above all, TBA outperforms in filtering peers to answer queries while maintaining an approximate accuracy to BFA and RBA.

Effect of Speed of Peers Fig. 6.11 plots the performance of P2P Search

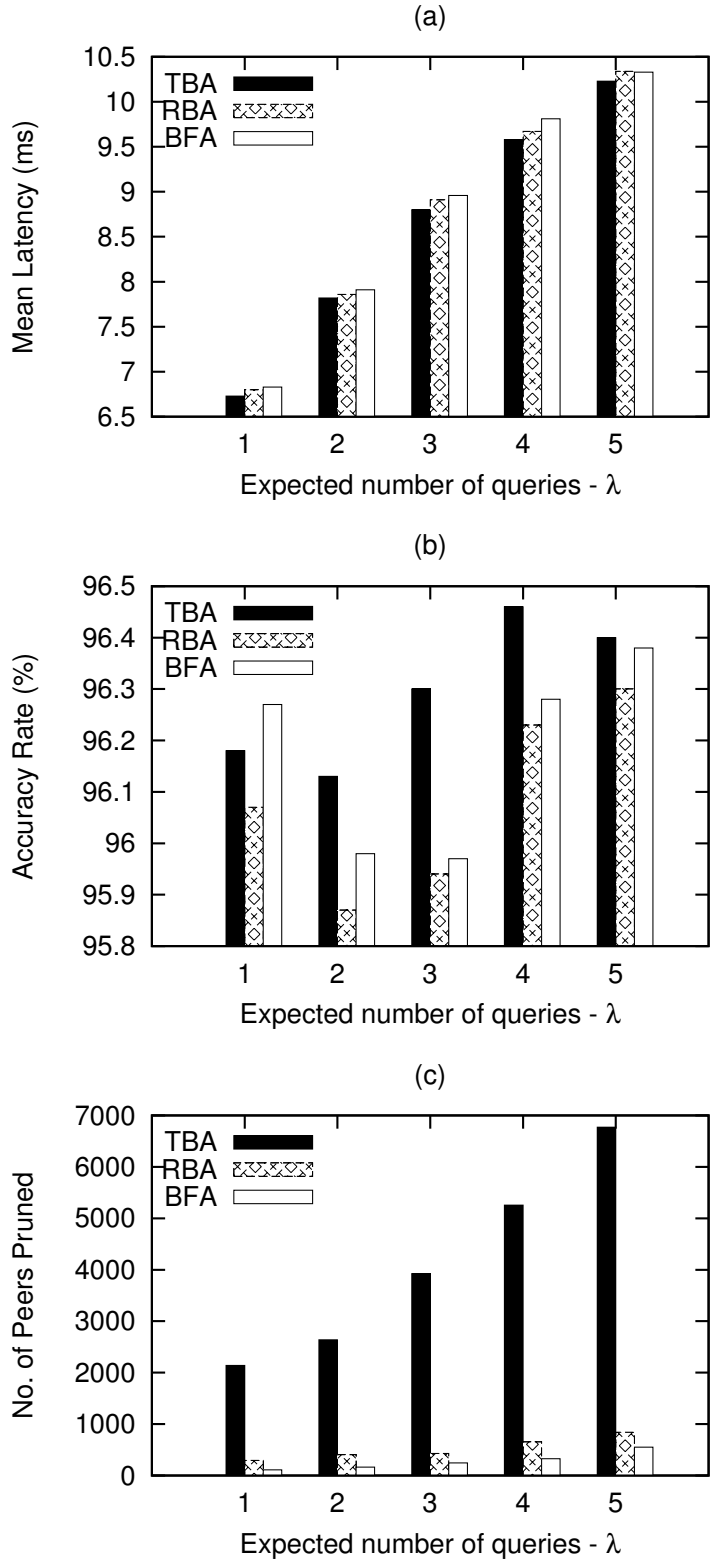


Figure 6.10: Effect of expected number of queries from each peer on P2P Search Algorithms. (a)Mean Latency. (b) Accuracy Rate. (c) Number of Peers Pruned by the Algorithms.

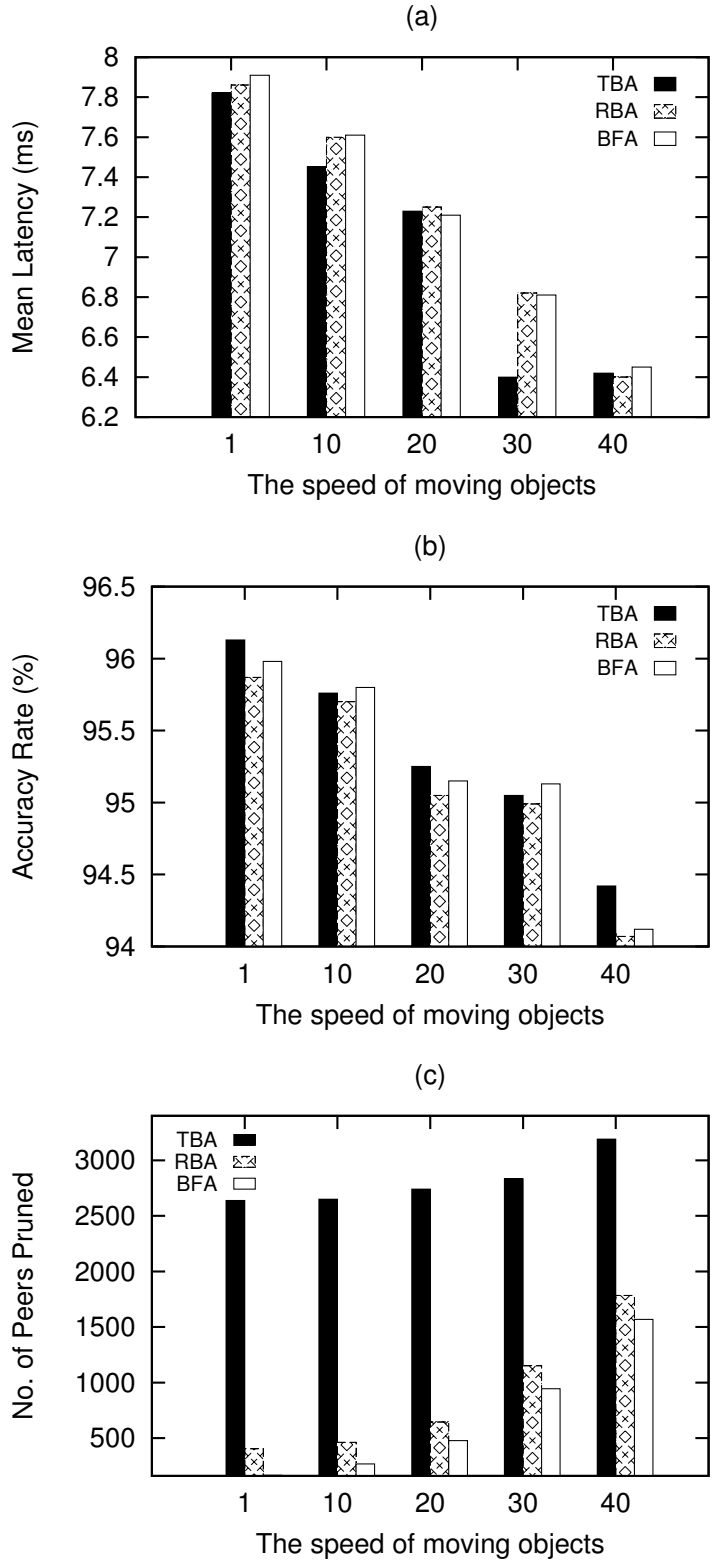


Figure 6.11: Effect of speed of peers on P2P Search Algorithms. (a)Mean Latency. (b) Accuracy Rate. (c) Number of Peers Pruned by the Algorithms.

Algorithms by varying the speed of peers. Although the accuracy rate of P2P falls as the more cached data become invalid, the P2P Search Algorithms is more time-efficient with the increase of peers' speed. BFA shows high rate of accuracy in a few cases as when history data are outdated, the more nodes are communicated, the better the accuracy rate is. Nevertheless, TBA is still proved as the best algorithm in most cases as it can select best peers to communicate in order to save query processing time and maintain high accuracy rate. In addition, it is easy to see that the faster the peers move, the more peers are pruned. One of the possible reason is from the time the query sends a beacon message to discover its peer to the time the query object sends the query to its peers, the location of query peer and other peers change. The increase of mobility leads to the increase of the location change. As a consequence, a number of the peers, discovered from Peer Discovery phase, are pruned by Lemma 2 and Lemma 3 in Section 4.5.2. Moreover, Figure 6.12(a) and (b) also show the trade-off between mean latency and accuracy rate. When the query processing time is longer, more data can be retrieved and processed. As a result, the query response can be more accurate.

Effect of Number of Peers Fig. 6.12 shows the performance of P2P Search Algorithms with respect to the number of peers. In general, P2P Search Algorithms is durable for large-scale network. The mean latency of all of three algorithms go up slightly when there are more peers due to more communication involved. However, in general, the advantages of TBA is clearer compared to BFA and RBA in term of mean latency when the number of moving objects increases. When the number of peers rise up, the filter condition (Lemma 3) becomes increasingly effective. By pruning a significant number of unnecessary peers as shown in Fig.6.12(c), TBA beats BFA and RFA in term of mean latency as Fig.6.12(a). Vice versa, time-out mechanism is not enough to keep the query processing time of BFA less than the other P2P Search Algorithms. However, compared to RBA, BFA is outperformed in term of accuracy than RBA as it can query more peers and access more data.

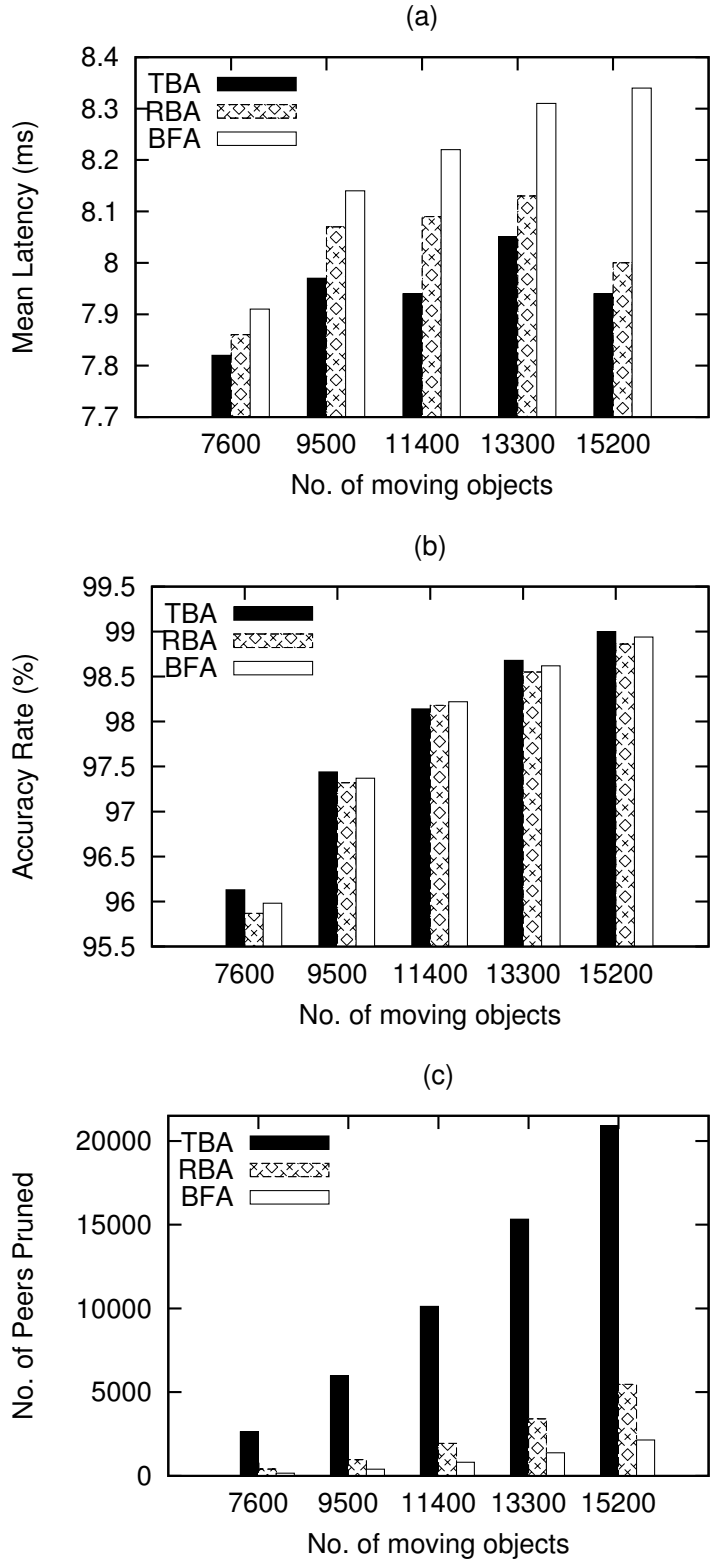


Figure 6.12: Effect of number of peers on P2P Search Algorithms. (a)Mean Latency. (b) Accuracy Rate. (c) Number of Peers Pruned by the Algorithms.

Nevertheless, TBA has been the best P2P Search Algorithm so far by selecting the best peers to ask within the time constraint.

6.6 Cost Analysis and Simulation for Group k Nearest Neighbours

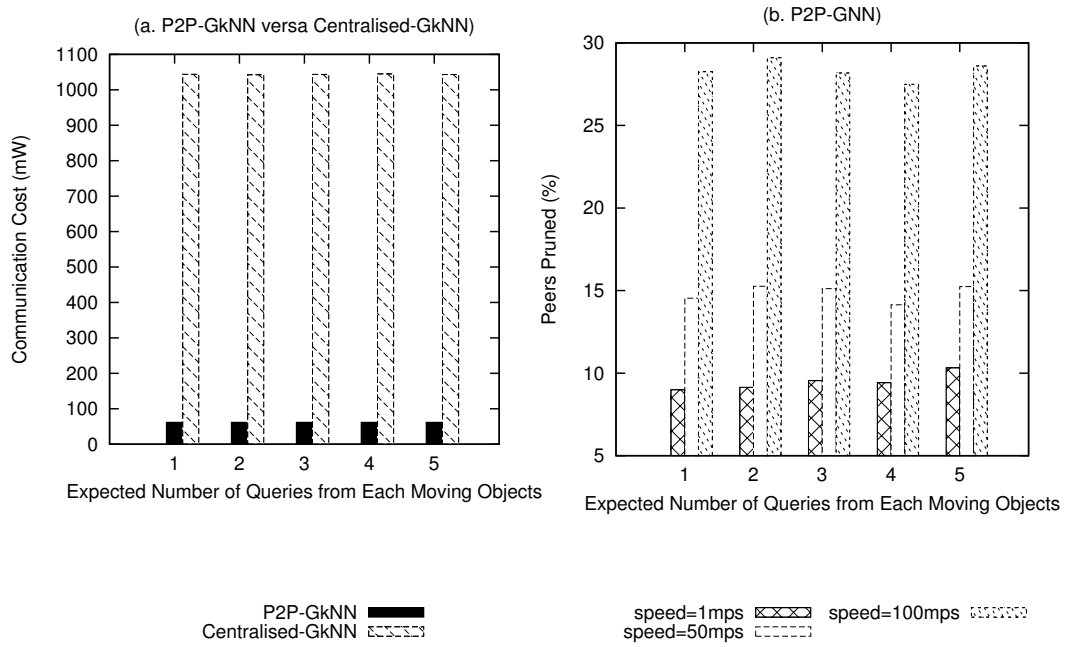


Figure 6.13: Communication cost and the percentage of peers pruned

6.6.1 Communication Cost

Centralised-GkNN: Communication to the BS is conducted via 3G(WCDMA), band I-2100 which is used by Vodaphone and Optus in Australia. Data rate for high speed moving peers in this network is 128kbps(Silicon-Press, 2011) and current consumption in connected state is 365.6mW(Option®, 2008).

As mentioned in simulation model, each moving peer is implemented a network interface card, Nic802154_TI_CC2420, which follows IEEE 802.15.4 standard for Low-Rate Wireless Local and Metropolitan Networks (LR-WPANs) such as ZigBee,

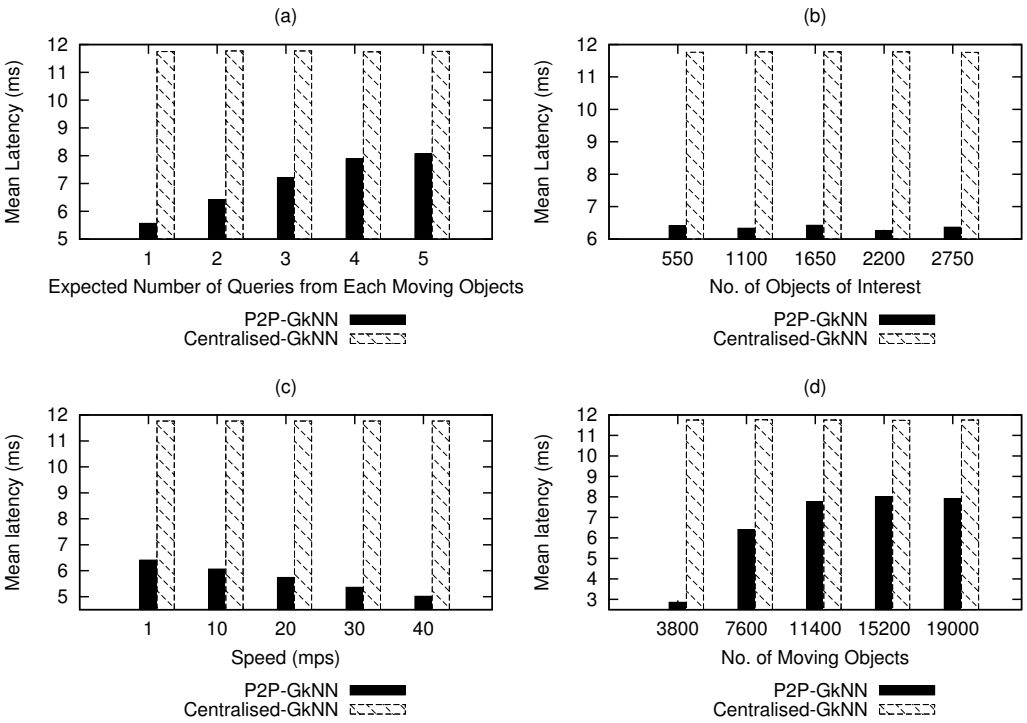


Figure 6.14: Mean Latency of P2P-GkNN versus Centralised-GkNN

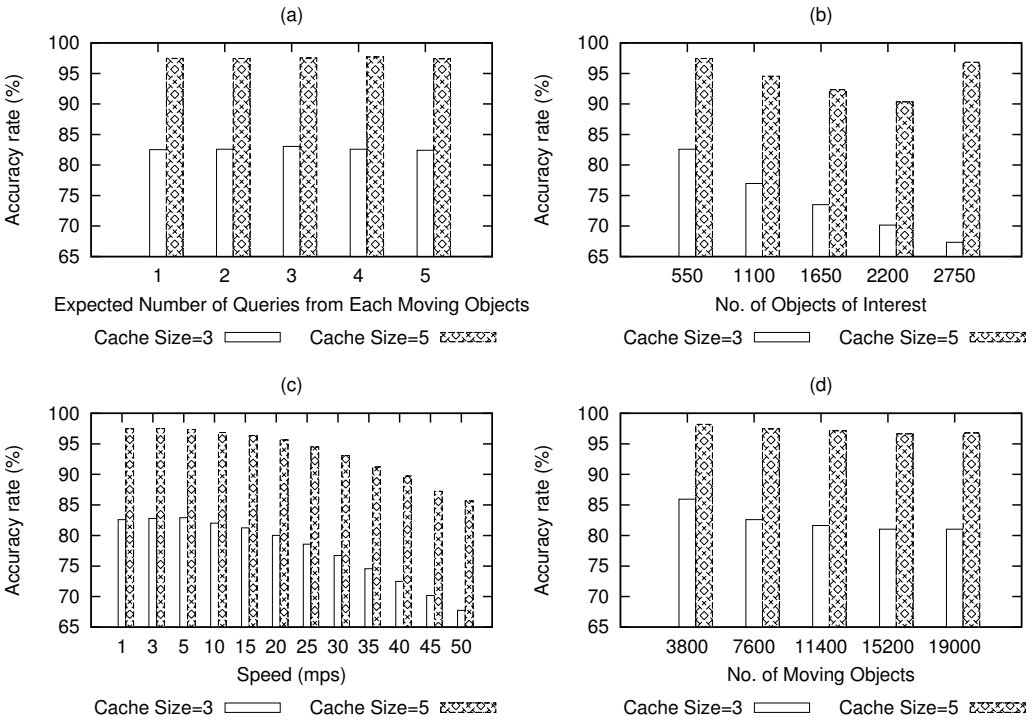


Figure 6.15: Accuracy rate of P2P-GkNN

Wireless HART and MiWi P2P. According to the configuration for the network

interface cards in MiXiM, transmission current txCurrent=17.4mW and receiving current rxCurrent=18.8mW.

Fig. 6.13a shows the average communication cost for each query processing by Centralised-GkNN and P2P-GkNN while the expected number of queries vary (λ). Apparently, P2P-GkNN is consistently and substantially more energy-efficient than Centralised-GkNN in a large-scale network since Centralised-GkNN needs to exchange messages with the BS via a wide-range communication like 3G.

6.6.2 Computation Cost

Centralised-GkNN: As the complexity of Centralised-GkNN search algorithm is similar to kNN search, the node accesses for kNN is estimated as follow (Tao, Zhang, Papadias and Mamoulis, 2004):

$$NA(k) = \sum_{i=0}^{\log_f \frac{N_{IO}}{f}} \left\lceil \frac{N_{IO}}{f_{i+1}} \right\rceil . \left(\frac{L_i - (L_i/2 + s_i/2)^2}{1 - s_i} \right)^2 \tag{6.5}$$

where N_{IO} is the number of objects of interests, f is the average node fanout, L_i is the length of each dimension of rectangle $R(q, L_i)$ such that the volume of $R(q, L_i)$ equals to that of the centroid $\Theta(q, r)$ and s_i the extent of a level-i node. Both s_i and L_i are estimated by a function of N_{IO},f and k as in (Tao, Zhang, Papadias and Mamoulis, 2004).

P2P-GkNN: Since P2P-GkNN does not maintain a tree to process queries, the efficiency in term of computation cost is evaluated by number peers involved solving a query. As moving peers are uniformly distributed, the number of objects of interest accesses is summarised as below:

$$NA = 2\pi R^2 (\frac{N_{MO}}{A}) \tag{6.6}$$

where R is transmission range of each moving peer, A is the network area and N_{MO} is the number of moving peers in the network.

In addition, in P2P-GkNN, not all peers of the query head q_{head} process the query. Instead, there are a significant number of peers filtered from query processing thanks to time-out mechanism. After a threshold of time, q_{head} will stop sending and receiving messages from peer and start processing received data in combination with its own cached data. Fig. 6.13b shows the percentage of peers pruned in respect to different number of queries and speed of moving peers. Accordingly, the greater number of queries, the more peers are pruned by P2P-GkNN algorithm. In addition, the speed of moving objects also affects the performance of P2P-GkNN. The algorithm can filter more peers in more dynamic environments.

6.6.3 Mean latency of query processing

Fig. 6.14 compares mean latency that requires P2P-GkNN and Centralised-GkNN to process a query. In general, GkNN-P2P can save at least 32% time in processing a query compare to Centralised-GkNN. Although the mean latency of P2P-GkNN slightly goes up when more queries are generated or the number of moving peers increases, it just fluctuates around 6.3ms no matter how the number of objects of interest varies. More interestingly, when the moving peers are faster, the query processing time reduces gradually.

6.6.4 Accuracy rate

Fig.6.15 illustrates the accuracy rate of P2P-GkNN. Generally, cache size at each moving peers gives an impact on this metric. While it is quite stable as the number of queries change, the trend of accuracy is decreasing gradually when there is an increase in the number of moving peers or objects of interest or the velocity of moving peers. Nonetheless, P2P-GkNN always maintains high accuracy rate of over 82% and this number can be significant improved by the increase of cache size at moving peers.

6.6.5 Discussion

In this paper, we confirmed the potential of a pure P2P query processing system in Mobile Networks to solve GkNN queries.The system harnesses collaboration of peers to answer queries without any support from server side. As a consequence, we eliminate the limitation of centralised approaches such as single point of failure and bottleneck problems. The simulation result shows that our P2P-GkNN algorithm significantly saves communication cost via the short-range network compared to the centralised algorithm regardless of changes to the network parameters. In addition the mean latency of query processing in our proposed algorithm is approximately 32% less than the other benchmarking system while maintaining a high accuracy rate of more than 82%. In general, the P2P system provides a practically feasible option to solve GkNN queries in a large-scale and busy network.

6.7 Chapter Summary

This chapter has been discussed several aspects of the performance evaluation of the research in this thesis. First, simulation tools are presented. Second, the simulation model are described in detail. Based on the simulation models and cost analysis, each proposed algorithm has been evaluated carefully. The results have also been compared with the centralised system (Benetis et al., 2002) and the hybrid system (Ku and Zimmermann, 2008). The experimental study uses both real and synthetic data sets show how the practical feasibility of the P2P approach in solving Nearest Neighbour queries for mobile networks. Indeed, the algorithms that will be proposed in this thesis are proved to harness the collaborative power of mobile devices, substantially increase energy efficiency and reduce latency time for ad-hoc networks with high density of moving objects.

"In literature and in life we ultimately pursue, not conclusions, but beginnings."

Sam Tanenhaus, *Literature Unbound*

7

Conclusion

7.1 Overview

In this chapter, the research presented in this thesis and its original contribution to the field will be summarised in Section 7.2. Then discussion and gaps in this research, and potential areas for future research will be shown in Section 7.3. Finally, Section 7.4 will discuss the wider implication of the research.

7.2 Main Findings

The thesis has revolved around the central research question: "Can the P2P approach solve Spatial Nearest Neighbour Queries in Mobile Environments?". The new novel, pure P2P, search algorithms first avoid the limitations of centralised search algorithms

such as scalability, bottlenecks, and low fault-tolerant problems. Second, the proposed algorithms can harness the computing power, intelligence and functionality of mobile devices.

The main findings of this thesis carefully addressed three cases arising from nearest neighbour queries. These were: kNN P2P query processing; RNN P2P query processing and GkNN P2P query processing.

7.2.1 kNN P2P Query Processing

As described in Chapter 3, a pure mobile P2P query processing system was proposed. The system harnesses collaboration of peers answer kNN queries without any central supervision. Two lightweight validation algorithms including: *Single Validation* and *Multiple Validation* were designed to assist the querying process. When a peer sends its cached objects to the query peer, first the query peer will perform *Single Validation*. It is based on the positions of the query object, the peer, and cached objects to verify whether the cached object can be used to answer the kNN query. If the spatial information from the peer is insufficient, then *Multiple Validation* will be called. Unlike *Single Validation*, *Multiple Validation* takes advantage of the collaboration of multiple peers to validate the cached objects.

Compared with centralised and hybrid systems, kNN P2P Query Processing System can significantly reduce energy consumption in a reasonable mean latency of processing time for networks with high density of moving objects, as shown in Section 6.4.

7.2.2 RNN P2P Query Processing

Based on the framework in the kNN P2P Query Processing research, RNN P2P Query Processing System was developed and proven to be a practically feasible solution to solve RNN queries in a large-scale and busy network in Chapter 4 and Section 6.5, respectively. The proposed system is based on three distinct, novel search algorithms including *Brute-Force Search Algorithm (BFA)*, *Regular Boundary*

Search Algorithm (RBA) and *Tight Boundary Search Algorithm (TBA)* without any support from the server side. In particular, BFA makes use of available information from the peers while two Boundary Search Algorithms reduce the number of queried peers. TBA is an enhancement of RBA in which a significant number of peers are filtered in processing an RNN query.

The simulation design and results showed that our P2P Search Algorithms significantly save communication cost via the short-range network compared to the centralised system, regardless of the change to network parameters. In addition the mean latency of query processing in our proposed system is up to 43% less than that of the centralised system. The BFA provides the brute-force approach to make use of available cached data from the peers with time-out mechanism. The RBA and TBA reduce the number of peers involved in query processing and shorten response time as a result. Overall, the TBA is the optimal option. It achieves high accuracy with time and energy efficiency for the mobile network.

7.2.3 GkNN P2P Query Processing

The final research subject in this thesis is GkNN queries. We developed a novel P2P algorithm focusing on GkNN queries in Chapter 5. P2P-GkNN makes use of all cached nearest neighbours from the peers to answer GkNN in the most efficient way in terms of overhead cost, processing time and accuracy rate. From our experimental study involving both real and synthetic data set, we found that our proposed algorithm is substantially more energy-efficient and save at least 32% latency time compared with the centralised GkNN algorithm while maintaining high accuracy rate more than 82%.

7.3 Discussion, Gaps and Further Research

The P2P search algorithms are confirmed to meet the three major challenges of P2P approaches in Section 1.2. How each of the challenges has been addressed and what remains to be done will be briefly discussed as below:

7.3.1 Challenge 1

The first challenge is how to design a light P2P query processing framework that is suitable for processing complex spatial queries in Mobile P2P Network. It is also required that the proposed framework be efficient in memory space, query processing time and power consumption.

How the challenge has been addressed in the thesis

From the work on kNN (Chapter 3), we developed a P2P Query Processing Framework to tackle the most common location-based queries, kNN queries. This framework eliminates the role of a base station and focuses on the power of collaboration between peers. The framework does not require a large memory for each moving object to store the full picture of the network. Instead, only cache data from previous queries by the peers are stored. In addition, considering that the volume of spatial data to be processed is not substantial, an R-tree or another index tree becomes unnecessary; therefore, they are excluded in the framework to avoid the overhead cost of generating, updating and maintaining the tree.

In addition, all communication between peers are based on short-range wireless technology. Besides, the validation algorithms in Chapter 3; RBA and TBA in Chapter 4, as well as time-out mechanism can prune unnecessary peer communication to decrease query processing time and energy consumption.

Gaps and Future Work

First, the proposed framework can actually be applied for more than only these 3 types of queries in this thesis (kNN, RNN and GkNN). It is possible to extend the

application of the proposed framework to solve other types of nearest neighbour queries that were discussed in Section 2.5 in Chapter 2 Literature Review, such as kNNs in spatial road network space, indoor space and obstacle space; All-kNNs and Reflexive kNNs. Although the framework does not require significant modification to adapt it to new types of nearest neighbour queries, suitable query processing algorithms for the specific types of queries will need to be investigated.

Second, the current framework is restricted to stationary objects of interest. The future direction of this research is to target queries that involve moving objects of interest. A promising application of this research can be monitoring moving objects of interest such as wild animals in a sanctuary. For example, visitors (as mobile peers) can share location data about wild tigers or lions around to alert other visitors.

7.3.2 Challenge 2

Without a centralised base station, the answer of queries in P2P systems largely depends on the availability of data cached in peers. Therefore, providing a system with a high accuracy rate is a real challenge in our research. Remember that the accuracy rate is measured by the percentage of answers from P2P systems that match the expected answers from the centralised search algorithm.

How the challenge has been addressed in the thesis

First, to improve the accuracy rate of query response, a priority queue is implemented by the peer initiating a query to store the location of its peers (Section 3.4.1, Chapter 3). The nearer peers will be communicated earlier as there is a higher probability that they share a common interest. For example, if the query peer q asks one of its peers p which is only 100 metres away from p about the nearest restaurant. It is most likely that the nearest restaurant whose location is cached in p is also the closest one to q. Remember that a peer can initialise and refresh its cache either when it connects to a terminal or when it receives and processes data from other peers.

Second, the proposed system is designed for urban areas where there is the high density of moving peers. The more peers, the more cached data. Therefore, the accuracy rate can be improved. Moreover, there is a trade-off between mean latency and accuracy rate as seen in Figure 6.12(a) and (b). When the query processing time is longer, more data can be retrieved and processed. As a result, the query response can be more accurate. In addition, if the size of cache memory of mobile peers increases, the accuracy rate will significantly improves as shown in Fig 6.15.

Gaps and Future Work

This research focuses on one-hop communication between peers. To improve the accuracy rate of query result, multi-hop communication could be considered in the future work. We can consider that each query peer is a tree root. Its direct peers are the first lever nodes which would become branches of the tree by recursively appending its peers. The more peers are involved in query processing, the more input is provided. As a result, the query results are likely to be more accurate. If multi-hop communication is adopted, the domain of the proposed system is no longer bound to urban areas, but expanded to areas with the low to medium density of moving peers.

However, more peer communication also means higher overhead costs of computation and data transmission. Therefore, a traversal algorithm along that tree to select the peer to communicate needs to satisfy two criteria. The first one is a high probability of answering. The second one is no duplication in term of visiting a node more than once since a moving object is possibly a peer of many other moving objects. A related issue is to avoid loops: i.e., the source of a query becomes a peer that is asked to respond. In addition, which routing path the query answer should follow to reach the query peer should also be researched if multi-hop communication is adopted. Another issue is that we need to investigate what happens if there are too many peers making queries; whether it causes escalating response time or deadlock.

In addition, we plan to investigate and implement an incentive model. It assigns economic rewards to mobile peers that share useful data in solving queries and penalises the ones that refuse to share data or share unrelated data.

7.3.3 Challenge 3

As mentioned in the previous section, the proposed system is currently designed to adopt in urban areas, which means each moving peer is often surrounded by many other peers. As a consequence, a query can be forwarded to many peers, which may lead inefficiencies in term of energy consumption and processing time. Therefore, another challenge is how to filter peers that contributes insignificantly to query processing.

How the challenge has been addressed in the thesis

First, in the validation algorithms of kNN work (Chapter 3), a query is not broadcast to all peers but subsequently sent to the peers in the priority queue. Remember that for the query peer, the priority queue stores the locations of its peers. The queue is sorted by the distance from the data item to the query peer. Once the query is answered or the pre-set time-out expires, the remainder in the priority queue will not receive or process the query.

Second, RBA and TBA in RNN work (Section 4.5.2, Chapter 4) limit the number of peers communicated by q. Accordingly, only the peers which contribute in building the boundary polygon B are sent queries.

In the proposed systems for all types of queries addressed in this thesis (kNN, RNN and GkNN), query peer q (or q_{head} in GkNN) is assigned an acknowledgement time-out period. If the waiting time exceeds time-out period, q or q_head stops waiting and proceeds query processing with the received acknowledgement messages from the peers that it has been reached.

Gaps and Future Work

First, at the moment, a time-out mechanism is implemented in GkNN only to limit redundant peer communication. However, as discussed in Section 5.7, Chapter 5, a GkNN problem can be regarded as a kNN problem, in which the query point is not at the query object's location but at the centroid point. Therefore it is possible to implement the validation algorithms as we did for kNNs. Then, the validation algorithms can prune unnecessary peer communication to reduce overhead costs.

Second, when many peers are involved in replying to a query, an arising question is whether all of them are reliable or not. In addition, any peer can join or leave the network, which causes many security threats since there is no base station to control the peer community. Each moving object needs to face and manage malicious behaviours by itself. Therefore, developing a new trust and reputation model should be considered in future work. For example, before a moving peer starts communicating with another, it could ask for the reputation stored by a close peer based on the peer relationship in the past, reputable rate and decide which peers are reliable. In this way it could avoid interacting with false nodes.

7.4 Wider Implications

The research in this thesis has introduced a pure P2P query processing for location-based queries. The system not only harnesses the collaborative power of peers but also it is specially useful when the base station is unavailable. Therefore, many practical location-based applications, ranging from everyday applications to disaster management, can be raised from this research. Below are some potential applications for each type of query.

7.4.1 kNNs

A traveller needs to find two nearest hotels by asking his friends around. In disaster management, in case of an earthquake, local people are disconnected from the

centralised base station. A victim in that area can take advantage of his smart phone to contact other people within his communication range and find out the nearest shelters.

7.4.2 RNNs

In everyday applications, police patrols can communicate with each other to distribute their team members to locations that need them most, for example, sites of car accidents or intersections congested with traffic. In public health applications, users use their iPhone to see all current outbreaks new strain of A(H1N1) virus or swine flu in their neighbourhood. In relation to this study, these applications can run RNN queries to find the set of sites taking the victims' location as the nearest neighbours. As a result, this information is helpful not only for international travellers but also for governments to detect and prevent infectious diseases in time.

7.4.3 GkNNs

A group of users may want to find a meeting venue that minimises the total travel distance for them. Or in disaster management, the rescue team can find the nearest shelters to a group of victims and bring food, medicine and other facilities there.

To sum up, when the research in this thesis can become products, businesses such as social networks, traffic management, search and rescue, etc can stand to benefit from the products.

All of the research in this thesis concerns the methods of dealing with spatial queries in a working P2P network. As such it presupposes that certain technical features have been implemented to make such a system possible. These features would include the following: a Global Positioning System (GPS), short-range communication technologies such as: Wi-Fi and Bluetooth, a cache memory and the mobile applications based on the P2P search algorithms proposed in this thesis.

A

Network Definition Source Code

A.1 P2P Query Network Definition

Below is the content of *P2PQueryNet.ned*, which defines the main network in the simulation.

```
1  package p2pquerynet;
2
3  import org.mixim.base.connectionManager.ConnectionManager;
4  import org.mixim.base.modules.BaseWorldUtility;
5
6  network P2PQueryNet
7  {
8      parameters:
9          double playgroundSizeX @unit(m);
10         double playgroundSizeY @unit(m);
11         double playgroundSizeZ @unit(m);
12
13         int numNodes; // total number of mobile peers in the network
14         int k; // type of query - kNN
15         int numInterestObjects; //number of objects of interest
16         @display("bgb=200,200;bgi=background/streetmap,s;bgl=4");
```

```
    submodules:
        connectionManager: ConnectionManager {
            parameters:
                @display("p=125,27;b=42,42,rect,green;
                    i=abstract/multicast");
        }
        world: MyWorldUtility {
            parameters:
                interestObjectNo = numInterestObjects;
                movingObjectNo = numNodes;
                k = k;
                playgroundSizeX = playgroundSizeX;
                playgroundSizeY = playgroundSizeY;
                playgroundSizeZ = playgroundSizeZ;

                @display("p=30,27;i=misc/globe");
        }
        MO[numNodes]: BaseNode {
            parameters:
                @display("p=146,155");
        }
        IO[numInterestObjects]: InterestObject {
            parameters:

                @display("p=112,88");
        }
        baseStation: BaseStation {
            parameters:
                @display("p=48,155;i=device/antennatower;is=n");
        }

    connections allowunconnected:
}
```

A.2 World Utility

```
package p2pquerynet;

import org.mixim.base.modules.BaseWorldUtility;

// Basic utility module for the whole network.
// Provides utility methods and information used by
// the whole network as well as simulation wide
// black board functionality.

simple MyWorldUtility extends BaseWorldUtility {
    parameters:
        @class(MyWorldUtility);
        @display("i=misc/globe");
        int interestObjectNo;
        int movingObjectNo;
        int k;
        int cacheSize;

        bool recordVectors = default(false);
        double bitrate;
        bool bcTraffic = default(true);
}
```

A.3 Connection Manager

```
//*************************************************************
// * file:       ConnectionManager.ned
// *
// * author:     Steffen Sroka, Daniel Willkomm, Karl Wessel,
// *                    Michael Swigulski
// *
// * copyright: (C) 2004 Telecommunication Networks Group (TKN) at
// *            Technische Universitaet Berlin, Germany.
// *
// *            This program is free software; you can
// *            redistribute it and/or modify it under the terms
// *            of the GNU General Public License as published by
// *            the Free Software Foundation
// *************************************************************
// * part of:     framework implementation developed by tkn
//*************************************************************


package org.mixim.base.connectionManager;

// Module to control all connection related stuff
//
// The central module that coordinates the connections between all
// nodes, and handles dynamic gate creation.
// ConnectionManager therefore
// periodically communicates with the mobility module and
// ChannelAccess.
//
// The four parameters pMax, sat, alpha, and carrierFrequency are
// used to calculate the interference distance between nodes.
// The values used here in ConnectionManager are used to calculate
// the upper bound, i.e. they can be redifined in the analogue
// models, but never such that the maximal interference
// distance is exceeded.
//
//
// @author Steffen Sroka, Daniel Willkomm, Karl Wessel
// @see BaseMobility
//
simple ConnectionManager
{
    parameters:
        // debug switch for core framework
        bool coreDebug;
        // send directly to the node or create separate gates for
        //every connection
        bool sendDirect;
        // maximum sending power used for this network [mW]
        double pMax @unit(mW);
        // minimum signal attenuation threshold [dBm]
        double sat @unit(dBm);
        // minimum path loss coefficient
        double alpha;
        // minimum carrier frequency of the channel [Hz]
        double carrierFrequency @unit(Hz);
        bool drawMaxIntfDist = default(false);

```

```
        @display("i=abstract/multicast");
}
```

A.4 Objects of Interest

```
 package p2pquerynet;

simple InterestObject
{
    parameters:
        @class(InterestObject);
        double xpos = uniform(0, 9300); //uniformly generated
        double ypos = uniform(0, 9300);
        int type;
        @display("p=$xpos,$ypos;i=block/star;is=vs");
}
```

Below is the implementation class of the previous ned file.

```
#ifndef INTERESTOBJECT_H_
#define INTERESTOBJECT_H_

#include <omnetpp.h>
#include <iostream>
#include "ObjectData.h"

using namespace std;

class InterestObject : public cSimpleModule {
public:
        void initialize(int stage);
        //InterestObject(int xpos, int ypos, short objectType);
        virtual ~InterestObject();
        void setProperties(int xpos, int ypos, short objectType, int
            objectID);
        void print();
        InterestObject& operator=(const InterestObject& other);
        ObjectData* getObjectData();

private:
        int x;
        int y;
        short type;
        int ID;
};

#endif /* INTERESTOBJECT_H_ */
```

A.5 Moving Peers

```
 package p2pquerynet;

import org.mixim.modules.nic.Nic802154_TI_CC2420;
import org.mixim.base.modules.BaseUtility;
import org.mixim.base.modules.BaseArp;
import org.mixim.base.modules.IBaseMobility;
import org.mixim.base.modules.IBaseApplLayer;
import org.mixim.base.modules.IBaseNetwLayer;
```

```
9  import org.mixim.modules.power.battery.SimpleBattery;
10 import org.mixim.modules.power.battery.BatteryStats;
11
12 module BaseNode
13 {
14     parameters:
15         string applType; //type of the application layer
16         string netwType; //type of the network layer
17         string mobType; //type of the mobility module
18         @display("bgb=,,white;i=abstract/person");
19     gates:
20         input radioIn; // gate for sendDirect
21     submodules:
22         utility: BaseUtility {
23             parameters:
24
25 @display("p=130,38;b=24,24,rect,black;i=block/blackboard");
26         }
27         arp: BaseArp {
28             parameters:
29
30 @display("p=130,101;b=24,24,rect,blue;i=block/process");
31         }
32         mobility: <mobType> like IBaseMobility {
33             parameters:
34                 @display("p=130,166;i=block/cogwheel");
35         }
36         appl: MyApp{
37             parameters:
38                 @display("p=59,38;i=app");
39         }
40         net: <netwType> like IBaseNetwLayer {
41             parameters:
42                 @display("p=60,101;i=block/layer");
43         }
44         nic: Nic802154_TI_CC2420 {
45             parameters:
46                 @display("b=32,30;p=60,233;i=block/ifcard");
47         }
48
49         battery: MyBattery{
50             @display("p=232,77;i=block/control");
51         }
52     connections allowunconnected:
53         nic.upperGateOut --> net.lowerGateIn;
54         nic.upperGateIn <-- net.lowerGateOut;
55         nic.upperControlOut --> {@display("ls=red;m=m,70,0,70,0");}
56                             --> net.lowerControlIn;
57         nic.upperControlIn <-- {@display("ls=red;m=m,70,0,70,0");}
58                            <-- net.lowerControlOut;
59
60         net.upperGateOut --> appl.lowerGateIn;
61         net.upperGateIn <-- appl.lowerGateOut;
62         net.upperControlOut --> {@display("ls=red;m=m,70,0,70,0");}
63                             --> appl.lowerControlIn;
64         net.upperControlIn <-- {@display("ls=red;m=m,70,0,70,0");}
65                            <-- appl.lowerControlOut;
66
67         radioIn --> appl.radioIn;
68
69 }
```

A.6 Nic

```
//*************************************************************
// * file:        Nic802154_TI_CC2420.ned
// *
// * author:      Jerome Rousselot, Marc Loebbers
// *
// * copyright: (C) 2008-2010 CSEM SA, Neuchatel, Switzerland.
// *            (C) 2004 Telecommunication Networks Group (TKN) at
// *            Technische Universitaet Berlin, Germany.
// *            This program is free software; you can
// *            redistribute it and/or modify it under the terms
// *            of the GNU General Public License as published by
// *            the Free Software Foundation
//*************************************************************

package org.mixim.modules.nic;

import org.mixim.modules.phy.PhyLayerBattery;
import org.mixim.modules.mac.CSMA802154;


//
// This NIC implements a Texas Instruments CC 2420 802.15.4 network
// interface card using the CSMA protocol as specified
// in IEEE 802.15.4-2006.
//// @author Jerome Rousselot
//
module Nic802154_TI_CC2420 like INic
{
    parameters:
        string connectionManagerName = default("");
        //name of the ConnectionManager module

        // power consumption from boards (at 3.3V)
            double sleepCurrent \@unit(mA) = 0.000021mA;
                double rxCurrent \@unit(mA) = 18.8 mA;
                double decodingCurrentDelta @unit(mA) = 0 mA;
                double txCurrent \@unit(mA) = 17.4 mA;
                double setupRxCurrent \@unit(mA) = 0.6391mA;
                // 0.002109 W
=(ESetupPower+ESetupXtal+ETxCalibrate)/TSetupTx

                double setupTxCurrent @unit(mA) = 0.6845mA;
                // 0.002259 W
                // = (ESetupPower+ESetupXtal+ERxCalibrate)/TSetupRx

                double rxTxCurrent @unit(mA) = 18.8 mA;
                       // Upper bound
                double txRxCurrent @unit(mA) = 18.8 mA;
                      // idem gates:
        input upperGateIn; // from upper layers
        input radioIn; // to receive AirFrames
        output upperGateOut; // to upper layers
        output upperControlOut; // control connection
        input upperControlIn;

    submodules:
        phy: PhyLayerBattery {
            parameters:
```

```
                decider = xmldoc("Nic802154_TI_CC2420_Decider.xml");
                //publishRSSIAlways = false;
                headerLength = 48 bit; // ieee 802.15.4
                thermalNoise = -110 dBm;
                // From TI CC1100 datasheet rev. C
                timeSleepToRX = 0.001792 s;
                timeSleepToTX = 0.001792 s;
                timeRXToTX = 0.000192 s;
                timeTXToRX = 0.000192 s;
                timeRXToSleep = 0 s;
                timeTXToSleep = 0 s;
                @display("p=96,236;i=block/wrxtx");

        }
        mac: CSMA802154 {
            parameters:
                rxSetupTime = 0.001792 s;
                txPower = default(1 mW);

                @display("p=96,87;i=block/layer");

        }
    connections:

        radioIn --> phy.radioIn;

        phy.upperGateIn <-- {@display("m=m,25,0,25,0");}
                        <-- mac.lowerGateOut;
        phy.upperGateOut --> {@display("m=m,73,0,50,50");}
                         --> mac.lowerGateIn;
        phy.upperControlOut --> mac.lowerControlIn;
        phy.upperControlIn <-- mac.lowerControlOut;

        mac.upperGateOut --> upperGateOut;
        mac.upperControlOut --> upperControlOut;
        mac.upperGateIn <-- upperGateIn;
        mac.upperControlIn <-- upperControlIn;

}
```

MAC implementation:

```
 //***************************************************************
//* file:         CSMA802154.ned
//*
//* author:       Jerome Rousselot, Marc Loebbers
//*
// * copyright: (C) 2004 Telecommunication Networks Group (TKN) at
// *             Technische Universitaet Berlin, Germany.
// *
// *             This program is free software; you can
// *             redistribute it and/or modify it under the terms
// *             of the GNU General Public License as published by
// *             the Free Software Foundation
//***************************************************************
//* part of:    Modifications to the MF-2 framework by CSEM
//***************************************************************
package org.mixim.modules.mac;

//
// IEEE 802.15.4-2006 non-beacon enabled CSMA protocol
```

```
20 // This model was independently validated on a
21 // wireless sensor network testbed.
22 // For more information, see
23 // Accurate Timeliness Simulations for Real-Time Wireless Sensor
24 // Networks,
25 // J. Rousselot, J.-D. Decotignie, M. Aoun, P. van der Stok,
26 // R. Serna Oliver,
27 // G. Fohler. In Proceedings of the 2009 Third UKSim European
28 // Symposium on Computer Modeling and Simulation.
29 // http://dx.doi.org/10.1109/EMS.2009.34
30 //
31 simple CSMA802154 extends csma
32 {
33     parameters:
34         @class(CSMA802154);
35
36         sifs @unit(s) = 0.000192 s; // 12 symbols
37         headerLength @unit(bit) = 72 bit;
38         queueLength = default(100);
39         bitrate @unit(bps) = 250000 bps;
40         ackLength @unit(bit) = 40 bit;
41         macMaxCSMABackoffs = default(4);
42           // takes values between 0 and 5
43         macMaxFrameRetries = default(3);
44           // takes values between 0 and 7
45         macAckWaitDuration @unit(s) = 0.00056 s;
46           // 1+12+10+12 symbols
47         ccaDetectionTime @unit(s) = 0.000128 s; // 8 symbols
48         aTurnaroundTime @unit(s) = 0.000192 s;
49           // 12 symbols
50         //txPower @unit(mW);
51         useMACAcks = default(true);
52
53         backoffMethod = "exponential";
54         aUnitBackoffPeriod @unit(s) = 0.00032 s;
55         macMaxBE = default(5);
56           // takes values between 3 and 8
57         macMinBE = default(3);
58           // takes values between 0 and macMaxBE
59 }
```

B

Message Definitions

B.1 Beacon Messages

```
cplusplus {{
#include "ApplPkt_m.h"
}}
packet ApplPkt;

//
// TODO generated message class
//
packet BeaconMessage extends ApplPkt {
    long srcNetworkAddress;
}
```

B.2 Acknowledgement Message

```
cplusplus {{
#include "ApplPkt_m.h"
#include "Coord.h"
}}
```

```
5
6
7  packet ApplPkt;
8
9  class Coord;
10 //
11 // TODO generated message class
12 //
13 packet BeaconReply extends ApplPkt {
14     double distFromQ;
15     Coord srcPosition; // position of source peer node
16     long srcNetworkAddress; // network address of source peer node
17 }
```

B.3 Query Messages

```
1  cplusplus {{
2  #include "ApplPkt_m.h"
3  #include "Coord.h"
4  }}
5  packet ApplPkt;
6  class Coord;
7  //
8  // TODO generated message class
9  //
10 packet Query extends ApplPkt {
11     int k; //number of objects of interest needed querying
12     char objectType;
13     long srcNetworkAddress;
14     Coord srcPosition;
15 }
```

B.4 Query Reply Messages

```
1  cplusplus {{
2  #include "ApplPkt_m.h"
3  #include "Coord.h"
4  #include <cQueue.h>
5  }}
6
7  packet ApplPkt;
8
9  class cQueue;
10 class Coord;
11
12 packet QueryReply extends ApplPkt
13 {
14     Coord srcPosition;
15     cQueue objectList;
16 }
```

C

An Example of A Configuration File

Below is the content of *omnet.ini*, a configuration file for GNN simulation while number of objects of interest varies.

```
[General]

network = P2PQueryNet
cmdenv-event-banners= false
cmdenv-event-banner-details = false
cmdenv-express-mode = true
cmdenv-interactive = false
**.vector-recording = false

num-rngs = 1

###############################################################
#                          Simulation parameters                                #
###############################################################
cmdenv-message-trace = false
debug-on-errors = false
record-eventlog = false
#eventlog-file = omnetpp.log
#**.param-record-as-scalar = true

**.**.coreDebug = false
```

```
22
23  #**.playgroundSizeX = 600m
24  #**.playgroundSizeY = 600m
25  **.playgroundSizeZ = 10m #ignored when use2D
26
27
28  ###########################################################
29  #                          WorldUtility parameters                          #
30  ###########################################################
31  **.world.useTorus = false
32  **.world.use2D = true
33  **.world.bitrate = 250000
34
35  ###########################################################
36  #                                channel parameters                         #
37  ###########################################################
38  **.connectionManager.sendDirect = false
39  **.connectionManager.carrierFrequency = 2.4E+9Hz
40
41
42  ##CSMA##############
43  **.connectionManager.pMax = 100mW
44  **.connectionManager.sat = -84dBm
45  **.connectionManager.alpha = 3.0
46
47  ################ PhyLayer parameters ######################
48  **.MO[*].nic.phy.usePropagationDelay = false
49
50  **.MO[*].nic.phy.analogueModels = xmldoc("config.xml")
51  **.MO[*].nic.phy.sensitivity = -100dBm
52  **.MO[*].nic.phy.maxTXPower = 1.1mW
53  **.MO[*].nic.phy.useThermalNoise = true
54
55  ######CSMA#########################
56
57
58  ##CSMA###################
59  **.MO[*].nic.mac.stats = false
60
61  ################ NETW layer parameters ####################
62  **.MO[*].netwType = "BaseNetwLayer"
63  **.MO[*].net.stats = false
64  **.MO[*].net.headerLength = 32bit
65
66  ################ Mobility parameters ######################
67
68  **.MO[*].mobType = "MyMobility"
69  **.MO[*].mobility.debug = false
70  **.MO[*].mobility.updateInterval = 1s
71  **.MO[*].mobility.acceleration = 0
72  **.MO[*].mobility.angle = 10deg
73  **.MO[*].mobility.x = -1
74  **.MO[*].mobility.y = -1
75  **.MO[*].mobility.z = -1
76
77  ############## Application parameter ###############
78
79  **.MO[*].applType = "MyApp"
80  **.MO[*].appl.debug = false
81  **.MO[*].appl.result-recording-modes = +histogram
82
83  **.MO[*].appl.groupSize = ${1, 3, 50,100,200,500}
84
```

```
**.MO[*].appl.headerLength = 512bit
#divided by minimum bitrate


############ BaseStation##########

**.baseStation.nic.phy.usePropagationDelay = false

**.baseStation.nic.phy.analogueModels = xmldoc("config.xml")
**.baseStation.nic.phy.sensitivity = -100dBm
**.baseStation.nic.phy.maxTXPower = 1.1mW
**.baseStation.nic.phy.useThermalNoise = true

**.baseStation.netwType = "BaseNetwLayer"
**.baseStation.net.stats = false
**.baseStation.net.headerLength = 32bit

**.baseStation.mobType = "BaseMobility"
**.baseStation.mobility.x = 0
**.baseStation.mobility.y = 0
**.baseStation.mobility.z = 0

**.baseStation.applType = "BSApp"
**.baseStation.appl.debug = false
**.baseStation.appl.headerLength = 512bit


############## Objects of Interest ############
**.IO[*].type = 1
#1 is RESTAURANT
#2 is GAS_STATION
#3 is SHOPPING_CENTRE


############battery module############
**.battery.debug = false
**.battery.capacity = 99999999.8mAh
**.battery.nominal = 9999999.8mAh
**.battery.voltage = 3.3V
**.battery.resolution = 0.1s
**.battery.publishDelta = 0.01
**.battery.publishTime = 0.1s
**.battery.numDevices = 2

############battery stats############


############test configuration##########

[Config GNN_IO_080413]
seed-0-mt = 1
**.playgroundSizeX = 9.3km
**.playgroundSizeY = 9.3km
sim-time-limit = 1s
**.MO[*].appl.type = ${0,1}
**.MO[*].mobility.speed = 1mps
**.numInterestObjects = ${550,550*2,550*3,550*4,550*5}
**.numNodes = 7600
**.k = 3
**.cacheSize = 50
**.MO[*].appl.lamda = 2/10
```

Publications

Most of the work presented in this thesis has been published. Below is a list of my publications that present results from the thesis.

Nghiem, T. P., Maulana, K., Green, D., Waluyo, A. B., and Taniar, D. (2013) Peer- to-peer bichromatic reverse nearest neighbours in mobile ad-hoc networks. In Journal of Parallel and Distributed Computing. under review.

Nghiem, T. P., Green, D., and Taniar, D. (2013) Peer-to-peer group k-nearest neigh- bours in mobile ad-hoc networks. In Proc. of the 19th IEEE International Conference on Parallel and Distributed Systems. Seoul, South Korea.

Nghiem, T. P., Maulana, K., Waluyo, A. B., Green, D., and Taniar, D. (2013) Bichromatic reverse nearest neighbors in mobile peer-to-peer networks. In Proc. of IEEE International Conference on Pervasive Computing and Communications. San Diego, LA, USA.

Nghiem, T. P., Waluyo, A., and Taniar, D. (2013) A pure peer-to-peer approach for knn query processing in mobile ad hoc networks, In Personal and Ubiquitous Computing 17(5): 973-985.

Nghiem, T. P., and Waluyo, A. B. (2011) . A pure p2p paradigm for query processing in mobile ad-hoc networks. In Proc. of the 9th International Conference on Advances in Mobile Computing and Multimedia. Ho Chi Minh city, Vietnam.

Permanent Address: Clayton School of Information Technology

Monash University

Australia

This thesis was typeset with LaTeX 2$_\varepsilon$[1] by the author.

[1] LaTeX 2$_\varepsilon$ is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this thesis were written by Glenn Maughan and modified by Dean Thompson and David Squire of Monash University.

Bibliography

ABS (2009). National regional profile: Inner melbourne (statistical subdivision), http://www.abs.gov.au.

Alamri, S., Taniar, D. and Safar, M. (2012). Indexing moving objects in indoor cellular space, Proc. of 15th International Conference on Network-Based Information Systems.

Alamri, S., Taniar, D. and Safar, M. (2013). Indexing of spatiotemporal objects in indoor environments, Proc. of 2013 IEEE 27th International Conference on Advanced Information Networking and Applications.

Albeseder, D. (n.d.). Comparison of network-simulators, Website. http://ti.tuwien.ac.at/ecs/people/albeseder/simcomp/simcomp.

Ali, M., Tanin, E., Zhang, R. and Kulik, L. (2008). Load balancing for moving object management in a p2p network, Database Systems for Advanced Applications, Vol. 4947, pp. 251–266.

Alliance, Z. (n.d.). Zigbee alliance, Website. http://www.zigbee.org/.

Beckmann, N., Kriegel, H., Schneider, R. and Seeger, B. (1990). The r*-tree: an efficient and robust access method for points and rectangles, SIGMOD Record **19**: 322–331.

Benetis, R., Jensen, C. S., Karciauskas, G. and Saltenis, S. (2002). Nearest neighbor and reverse nearest neighbor queries for moving objects, Proc. of International Symposium on Database Engineering and Applications.

Bohm, C. and Krebs, F. (2003). Supporting kdd applications by the k-nearest neighbor join, in V. Marik, W. Retschitzegger and O. Stepankova (eds), Database and Expert Systems Applications, Vol. 2736 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp. 504–516.

Cao, J., Zhang, Y., Cao, G. and Xie, L. (2007). Data consistency for cooperative caching in mobile environments, Computer **40**(4): 60–66.

Cheema, M., Lin, X., Zhang, W. and Zhang, Y. (2011). Influence zone: Efficiently processing reverse k nearest neighbors queries, Proc. of the 27th IEEE Conference on Data Engineering, pp. 577 –588.

Chen, D., Zhou, J. and Le, J. (2006). Reverse nearest neighbor search in peer-to-peer systems, Flexible Query Answering Systems, Vol. 4027, pp. 87–96.

Chen, Y. and Patel, J. (2007). Efficient evaluation of all-nearest-neighbor queries, Proc. of 23rd IEEE International Conference on Data Engineering, Istanbul, Turkey.

Cheung, K. L. and Fu, A. W. (1998). Enhanced nearest neighbour search on the r-tree, SIGMOD Record **27**: 16–21.

Chow, C., Leong, H. V. and Chan, A. T. S. (2007). Grococa: group-based peer-to-peer cooperative caching in mobile environment, IEEE Journal on Selected Areas in Communications **25**(1): 179 –191.

Chow, C., Mokbel, M. and Leong, H. (2011). On efficient and scalable support of continuous queries in mobile peer-to-peer environments, IEEE Transactions on Mobile Computing **10**: 1473–1487.

Ciou, Y., Leu, F., Huang, Y. and Yim, K. (2011). A handover security mechanism employing the diffie-hellman key exchange approach for the ieee802.16e wireless networks, Mobile Information Systems **7**(3): 241–269.

Clarkson, K. (1983). Fast algorithms for the all nearest neighbors problem, Pro. of 24th Annual Symposium on Foundations of Computer Science, Washington, DC, USA.

Corral, A., Manolopoulos, Y., Theodoridis, Y. and Vassilakopoulos, M. (2000). Closest pair queries in spatial databases, Proc. of the 2000 ACM SIGMOD international conference on Management of data.

Corral, A., Manolopoulos, Y., Theodoridis, Y. and Vassilakopoulos, M. (2004). Algorithms for processing k-closest-pair queries in spatial databases, Data Knowledge Engineering **49**(1): 67–104.

Dahlin, M. D., Wang, R. Y., Anderson, T. E. and Patterson, D. A. (1994). Cooperative caching: using remote client memory to improve file system performance, Proc. of the 1st USENIX conference on Operating Systems Design and Implementation.

Dar, S., Franklin, M. J., Jónsson, B. T., Srivastava, D. and Tan, M. (1996). Semantic data caching and replacement, Proc. of the 22th International Conference on Very Large Data Bases.

Datta, A., Quarteroni, S. and Aberer, K. (2004). Autonomous gossiping: A self-organizing epidemic algorithm for selective information dissemination in wireless mobile ad-hoc networks, Semantics of a Networked World, Vol. 3226, pp. 126–143.

Delot, T., Ilarri, S., Cenerario, N. and Hien, T. (2011). Event sharing in vehicular networks using geographic vectors and maps, Mobile Information Systems **7**: 21–44.

Delot, T., Mitton, N., S., I. and Hien, T. (2011). Geovanet: A routing protocol for query processing in vehicular networks, Mobile Information Systems **7**(4): 329–359.

Demirbas, M. and Ferhatosmanoglu, H. (2003). Peer-to-peer spatial queries in sensor networks, Pro. of the 3rd International Conference on Peer-to-Peer Computing.

Denko, M. K. and Tian, J. (2006). Cooperative caching with adaptive prefetching in mobile ad hoc networks, Proc. of IEEE International Conference on Wireless and Mobile Computing, Networking and Communications.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs, Numerische Mathematik **1**: 269–271.

Dittrich, J., Blunschi, L. and Vaz Salles, M. (2009). Indexing moving objects using short-lived throwaway indexes, Advances in Spatial and Temporal Databases, Vol. 5644, pp. 189–207.

Du, Y., Gupta, S. K. and Varsamopoulos, G. (2009). Improving on-demand data access efficiency in manets with cooperative caching, Ad Hoc Networks **7**(3): 579–598.

Feeney, L. M. and Willkomm, D. (2010). Energy framework: an extensible framework for simulating battery consumption in wireless networks, Proc. of the 3rd International ICST Conference on Simulation Tools and Techniques, pp. 20:1–20:4.

Fitzek, F. H. P. and H., C. (2007). Springer.

Ghadiri, N., Baraani-Dastjerdi, A., Ghasem-Aghaee, N. and Nematbakhsh, M. A. (2011). Optimizing the performance and robustness of type-2 fuzzy group nearest-neighbor queries, Mobile Information Systems **7**: 123–145.

Goyal, M., Xie, W. and H., H. (2011). Ieee 802.15.4 modifications and their impact, Mobile Information Systems **7**: 69–92.

Gu, Y. and A., B. (2011). {HD} tree: A novel data structure to support multi-dimensional range query for {P2P} networks, Journal of Parallel and Distributed Computing **71**(8): 1111 – 1124.

Guttman, A. (1984). R-trees: a dynamic index structure for spatial searching, SIGMOD Record **14**: 47–57.

Hackmann, G. (2006). 802.15 personal area networks, Department of Computer Science and Engineering, Washington University .

Hadjieleftheriou, M. (2013). http://libspatialindex.github.io/#id1.

Hashem, T., Kulik, L. and Zhang, R. (2010). Privacy preserving group nearest neighbor queries, Proc. of the 13th International Conference on Extending Database Technology.

Hjaltason, G. R. and Samet, H. (1999). Distance browsing in spatial databases, ACM Transaction on Database Systems **24**: 265–318.

Hu, H., Xu, J., Wong, W., Zheng, B., Lee, D. and Lee, W. (2005). Proactive caching for spatial queries in mobile environments, Proceedings. 21st International Conference on Data Engineering.

Huang, Y. and Garcia-Molina, H. (2004). Publish/subscribe in a mobile environment, Wireless Networks **10**: 643–652.

IEEE802.15.1 (2002). Ieee 802.15 wpan task group 1 (tg1), Website. `http://www.ieee802.org/15/pub/TG1.html`.

IEEE802.15.4 (2003). Ieee 802.15 wpan task group 4 (tg4), Website. `http://www.ieee802.org/15/pub/TG3.html`.

Ilarri, S., Mena, E. and Illarramendi, A. (2006). Location-dependent queries in mobile contexts: distributed processing using mobile agents, IEEE Transactions on Mobile Computing **5**(8): 1029–1043.

Jain, A. K., Murty, M. N. and Flynn, P. J. (1999). Data clustering: a review, ACM Comput. Surv. **31**(3): 264–323.

Jarvis, R. A. and Patrick, E. A. (1973). Clustering using a similarity measure based on shared near neighbors, IEEE Transactions on Computers **C-22**(11): 1025–1034.

Johnson, S. (1967). Hierarchical clustering schemes, Psychometrika **32**(3): 241–254.

Kamel, I. and Faloutsos, C. (1994). Hilbert r-tree: An improved r-tree using fractals, Proc. of the 20th International Conference on Very Large Data Bases, VLDB '94, pp. 500–509.

Kang, J. M., Mohamed, F. M., Shekhar, S., Xia, T. and Zhang, D. (2007). Continuous evaluation of monochromatic and bichromatic reverse nearest neighbors, Proc. of the IEEE 29th International Conference on Data Engineering.

Kang, J. M., Mokbel, M. F., Shekhar, S., Xia, T. and Zhang, D. (2010). Incremental and general evaluation of reverse nearest neighbors, Knowledge and Data Engineering, IEEE Transactions on **22**(7): 983–999.

Karp, B. and Kung, H. T. (2000). Gpsr: greedy perimeter stateless routing for wireless networks, Proc. of the 6th Annual International Conference on Mobile Computing and Networking.

Kim, Y., Shim, Y. and Lee, K. (2011). A cluster-based web service discovery in manet environments, Mobile Information Systems **7**: 299–315.

Koenig, S. and Smirnov, Y. (1996). Graph learning with a nearest neighbor approach, Proc. of the ninth annual conference on Computational learning theory.

Kolahdouzan, M. and Shahabi, C. (2004). Voronoi-based k nearest neighbor search for spatial network databases, Proc. of the 13th International Conference on Very Large Data Bases, pp. 840–851.

Kollios, G., Gunopulos, D. and Tsotras, V. (1999). Nearest neighbor queries in a mobile environment, Spatio-Temporal Database Management, Vol. 1678, pp. 119–134.

Köpke, A., Swigulski, M., Wessel, K., Willkomm, D., Haneveld, P. T. K., Parker, T. E. V., Visser, O. W., Lichte, H. S. and Valentin, S. (2008). Simulating wireless and mobile networks in omnet++ the mixim vision, Proc. of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops.

Korn, F. and Muthukrishnan, S. (2000). Influence sets based on reverse nearest neighbor queries, Proc. of the 2000 ACM SIGMOD international conference on Management of data.

Ku, W. and Zimmermann, R. (2006). Location-based spatial queries with data sharing in mobile environments, Proc of the 22nd IEEE Conference on Data Engineering Workshops.

Ku, W. and Zimmermann, R. (2008). Nearest neighbor queries with peer-to-peer data sharing in mobile environments, Pervasive and Mobile Computing **4**(5): 775 – 788.

Kulik, J., Heinzelman, W. and Balakrishnan, H. (2002). Negotiation-based protocols for disseminating information in wireless sensor networks, Wireless Networks **8**: 169–185.

Lee, K., Lee, W., Zheng, B. and Xu, J. (2006). Caching complementary space for location-based services, Proc. of the 10th International Conference on Extending Database Technology.

Lehikoinen, J., Salminen, I., Aaltonen, A., Huuskonen, P. and Kaario, J. (2006). Meta-searches in peer-to-peer networks, Personal and Ubiquitous Computing **10**: 357–367.

Li, D. and Lee, D. (2008). A lattice-based semantic location model for indoor navigation, Proc. of the 9th International Conference on Mobile Data Management.

Li, G., Fan, P., Li, Y. and Du, J. (2010). An efficient technique for continuous k-nearest neighbor query processing on moving objects in a road network, Proc. of 2010 IEEE 10th International Conference on Computer and Information Technology.

Li, H., Lu, H., Huang, B. and Huang, Z. (2005). Two ellipse-based pruning methods for group nearest neighbor queries, Proc. of the 13th annual ACM international workshop on Geographic information systems.

Li, Y., Chen, H., Xie, R. and Wang, J. Z. (2011). Bgn: A novel scatternet formation algorithm for bluetooth-based sensor networks, Mobile Information Systems **7**: 93–106.

Lian, X. and Chen, L. (2008). Probabilistic group nearest neighbor queries in uncertain databases, IEEE Transactions on Knowledge and Data Engineering **20**(6): 809–824.

Lian, X. and Chen, L. (2009). Efficient processing of probabilistic reverse nearest neighbor queries over uncertain data, The VLDB Journal **18**(3): 787–808.

Lim, S., Lee, W., Cao, G. and Das, C. R. (2006). A novel caching scheme for improving internet-based mobile ad hoc networks performance, Ad Hoc Networks **4**(2): 225–239.

Loo, A. (2007). Springer.

Luo, Y., Chen, H., Furuse, K. and Ohbo, N. (2007). Efficient methods in finding aggregate nearest neighbor by projection-based filtering, Proc. of the 2007 international conference on Computational science and its applications.

Luo, Y., Wolfson, O. and Xu, B. (2008). Mobile local search via p2p databases, Proc. of 2nd IEEE International Interdisciplinary Conference on Portable Information Devices.

Luo, Y., Wolfson, O. and Xu, B. (2010). The mobi-dik approach to searching in mobile ad hoc network databases, Handbook of Peer-to-Peer Networking, Vol. 9, pp. 1105–1123.

M., S. (1986). Mutual and shared neighbor probabilities finite- and infinite-dimensional results,, Advances in Applied Probability **18**: 388–405.

MAC (n.d.). Mac simulator, Website. `http://www.consensus.tudelft.nl/software.html/`.

Mauve, M., Widmer, A. and Hartenstein, H. (2001). A survey on position-based routing in mobile ad hoc networks, IEEE Network **15**(6): 30–39.

Melbourne-Hotel-Map (2011). Interactive hotel map, `http://melbournehotelmap.com/`.

MF (n.d.). Mobility framework (mf) for simulating wireless and mobile networks using omnet++, Website. `http://mobility-fw.sourceforge.net/`.

Mouratidis, K., Papadias, D. and Hadjieleftheriou, M. (2005). Conceptual partitioning: an efficient method for continuous nearest neighbor monitoring, Proc. of the 2005 ACM SIGMOD international conference on Management of data.

Namnandorj, S., Chen, H., Furuse, K. and Ohbo, N. (2008). Efficient bounds in finding aggregate nearest neighbors, Proc. of the 19th international conference on Database and Expert Systems Applications.

NBCNews (2011). Japan phone service struggling post quake, `http://www.nbcnews.com/technology/japan-phone-service-struggling-post-quake-124578`.

Nekovee, M. and Bogason, B. B. (2007). Reliable and effcient information dissemination in intermittently connected vehicular adhoc networks, VTC Spring.

Nghiem, T. P., Green, D. and Taniar, D. (2013). Peer-to-peer group k-nearest neighbours in mobile ad-hoc networks, Proc. of the 19th IEEE International Conference on Parallel and Distributed Systems. under review.

Nghiem, T. P., Maulana, K., Green, D., Waluyo, A. B. and Taniar, D. (2013). Peer-to-peer bichromatic reverse nearest neighbors in mobile ad-hoc networks, Journal of Parallel and Distributed Computing . under review.

Nghiem, T. P., Maulana, K., Waluyo, A. B., Green, D. and Taniar, D. (2013). Bichromatic reverse nearest neighbors in mobile peer-to-peer networks, Proc. of 2013 IEEE International Conference on Pervasive Computing and Communications.

Nghiem, T. P. and Waluyo, A. B. (2011). A pure p2p paradigm for query processing in mobile ad-hoc networks, Proc. of the 9th International Conference on Advances in Mobile Computing and Multimedia.

Nghiem, T., Waluyo, A. and Taniar, D. (2013). A pure peer-to-peer approach for knn query processing in mobile ad hoc networks, Personal and Ubiquitous Computing **17**(5): 973–985.

Nock, R., Sebban, M. and Bernard, D. (2003). A simple locally adaptive nearest neighbor rule with application to pollution forecasting, Internal Journal of Pattern Recognition and Artificial Intelligence **17**: 2003.

ns2 Community (n.d.). Website. `http://www.isi.edu/nsnam/ns/`.

Nuggehalli, P., Srinivasan, V. and Chiasserini, C. (2003). Energy-efficient caching strategies in ad hoc wireless networks., Proc. of the 4th ACM international symposium on Mobile ad hoc networking & computing.

OMNeT++ (n.d.). Website. `http://www.omnetpp.org/`.

Option® (2008). Power considerations for 2g & 3g modules in mid designs, `http://www.option.com/en/newsroom/media-center/white-papers/`.

Option® (2011). Wi-fi certified wi-fi direct®:personal, portable wi-fi® that goes with you anywhere, anytime, `http://www.wi-fi.org/discover-and-learn/wi-fi-direct`.

Padhariya, N., Mondal, A., Goyal, V., Shankar, R. and Madria, S. K. (2011). Ecotop: An economic model for dynamic processing of top-k queries in mobile-p2p networks, Database Systems for Advanced Applications, Vol. 6588, pp. 251–265.

Pallavicini, M., Patrignani, C., Pontil, M. and Verri, A. (1998). The nearest-neighbor technique for particle identification, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **405**(1): 133–138.

Papadias, D., Shen, Q., Tao, Y. and Mouratidis, K. (2004). Group nearest neighbor queries, Proc. of the 21st IEEE Conference on Data Engineering.

Papadias, D., Tao, Y., Mouratidis, K. and Hui, C. K. (2005). Aggregate nearest neighbor queries in spatial databases, ACM Transactions on Database Systems **30**: 529–576.

Papadias, D., Zhang, J., Mamoulis, N. and Tao, Y. (2003). Query processing in spatial network databases, Proc. of the 29th International Conference on Very large data bases.

Park, K. and Valduriez, P. (2011). Energy efficient data access in mobile p2p networks, IEEE Transactions on Knowledge and Data Engineering **23**(11): 1619–1634.

Positif (n.d.). Positif localization simulation framework, Website. `http://www.consensus.tudelft.nl/software.html`.

Pucha, H., Das, S. M. and Hu, Y. C. (2004). Ekta: an efficient dht substrate for distributed applications in mobile ad hoc networks, Proc. of Sixth IEEE Workshop on Mobile Computing Systems and Applications.

Ren, Q., Dunham, M. and Kumar, V. (2003). Semantic caching and query processing, IEEE Transactions on Knowledge and Data Engineering **15**(1): 192–210.

Roussopoulos, N., Kelley, S. and Vincent, F. (1995). Nearest neighbor queries, SIGMOD Record **24**: 71–79.

Safar, M., El-Amin, D. and Taniar, D. (2011). Optimized skyline queries on road networks using nearest neighbors, Personal and Ubiquitous Computing **15**(8): 845–856.

Safar, M., Ibrahimi, D. and Taniar, D. (2009). Voronoi-based reverse nearest neighbor query processing on spatial networks, Multimedia Systems **15**: 295–308.

Sailhan, F. and Issarny, V. (2003). Cooperative caching in ad hoc networks, Mobile Data Management, Vol. 2574, pp. 13–28.

Saltenis, S., Jensen, C. S., Leutenegger, S. T. and Lopez, M. A. (2000). Indexing the positions of continuously moving objects, SIGMOD Record **29**: 331–342.

Sellis, T. K., Roussopoulos, N. and Faloutsos, C. (1987). The r+-tree: A dynamic index for multi-dimensional objects, Proc. of 13th International Conference on Very Large Data Bases.

Silicon-Press (2011). Third generation (3g) wireless technology brief, `http://www.silicon-press.com/briefs/brief.3g/index.html`.

Sistla, P. A., Wolfson, O., Chamberlain, S. and Dao, S. (1997). Modeling and querying moving objects, Proc. of the 13th IEEE Conference on Data Engineering.

Stanoi, I., Agrawal, D. and Abbadi, A. E. (2000). Reverse nearest neighbor queries for dynamic databases, Pro. of ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery.

Strassman, M. and Collier, C. (2004). Case study: The development of the find friends application, in J. H. Schiller and A. Voisard (eds), Location-Based Services, Morgan Kaufmann, pp. 27–40.

Sturek, D. (2005). Zigbee architecture and specifications overview, ZigBee Alliance presentation. `http://www.zigbee.org/en/events/documents/December2005_Open_House_Presentations/043120r008ZB_Architecture-ZigBeeV1-0Architecture.pdf`.

Taniar, D. and Rahayu, W. (2013). A taxonomy for nearest neighbour queries in spatial databases, Journal of Computer and System Sciences **79**(7): 1017 – 1039.

Tao, Y., Papadias, D. and Lian, X. (2004). Reverse knn search in arbitrary dimensionality, Proc. of the Thirtieth international conference on Very large data bases.

Tao, Y., Papadias, D., Lian, X. and Xiao, X. (2007). Multidimensional reverse knn search, The VLDB Journal **16**(3): 293–316.

Tao, Y., Zhang, J., Papadias, D. and Mamoulis, N. (2004). An efficient cost model for optimization of nearest neighbor search in low and medium dimensional spaces, IEEE Transactions on Knowledge and Data Engineering **16**: 1169–1184.

TeleGeography (2011). Communications hit after earthquake, `http://www.telegeography.com/products/commsupdate/articles/2011/03/14/communications-hit-after-earthquake/`.

timesofindia (2011). Tokyo phone lines jammed, trains stop, `http://articles.timesofindia.indiatimes.com/2011-03-12/rest-of-world/28683249_1_trains-phone-lines-tokyo`.

Tran, Q., Taniar, D. and Safar, M. (2010). Bichromatic reverse nearest-neighbor search in mobile systems, IEEE Systems Journal **4**(2): 230–242.

Vaidya, P. M. (1989). An o(n log n) algorithm for the all-nearest-neighbors problem, Discrete and Computational Geometry **4**(2): 101–115.

Valentin, S. (2006). Chsim, a wireless channel simulator for omnet++, Proc.p of TKN Simulation Workshop.

Varga, A. (2001). The omnet++ discrete event simulation system, Proc. of the European Simulation Multiconference (ESM'2001), Vol. 9, p. 185.

Varga, A. and Hornig, R. (2008). An overview of the omnet++ simulation environment, Proc. of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops.

Visvanathan, A., Youn, J. and Deogun, J. (2005). Hierarchical data dissemination scheme for large scale sensor networks, Proc. of 2005 IEEE International Conference on Communications.

Waluyo, A., Srinivasan, B. and Taniar, D. (2003). Global index for multi channel data dissemination in mobile databases, Computer and Information Sciences - ISCIS 2003, Vol. 2869, pp. 212–219.

Waluyo, A., Srinivasan, B., Taniar, D. and Rahayu, W. (2005). Incorporating global index with data placement scheme for multi channels mobile broadcast environment, Embedded and Ubiquitous Computing, Vol. 3824, pp. 755–764.

Waluyo, A., Taniar, D., Rahayu, W. and Srinivasan, B. (2009). Mobile service oriented architectures for nn-queries, Journal of Network and Computer Applications **32**(2): 434–447.

Wessels, D. and Claffy, K. (1998). Icp and the squid web cache, IEEE Journal on Selected Areas in Communications **16**(3): 345–357.

Wilkes, M. V. (1965). Slave memories and dynamic storage allocation, Electronic Computers, IEEE Transactions on **EC-14**(2): 270–271.

Wolfson, O., Xu, B., Yin, H. and Cao, H. (2006). Search-and-discover in mobile p2p network databases, Proc. of the 26th IEEE International Conference on Distributed Computing Systems.

Wu, W. and Cao, J. (2012). Efficient cache discovery for cooperative caching in wireless ad hoc networks, Proc. of 2012 IEEE 18th International Conference on Parallel and Distributed Systems.

Wu, W., Yang, F., Chan, C. and Tan, K. (2008). Finch: evaluating reverse k-nearest-neighbor queries on location data, Proc. of the VLDB Endowment, pp. 1056–1067.

Xia, C., Hsu, D. and Tung, A. (2004). A fast filter for obstructed nearest neighbor queries, in H. Williams and L. MacKinnon (eds), Key Technologies for Data Management, Vol. 3112 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp. 203–215.

Xia, T. and Zhang, D. (2006). Continuous reverse nearest neighbor monitoring, Proc. of 22nd International Conference on Data Engineering.

Xie, X., Lu, H. and Pedersen, T. (2013). Efficient distance-aware query evaluation on indoor moving objects, Proc. of IEEE 29th International Conference on Data Engineering (ICDE).

Xiong, X., Mokbel, M. F. and Aref, W. G. (2005). Sea-cnn: scalable processing of continuous k-nearest neighbor queries in spatio-temporal databases, Proc. of 21st International Conference on Data Engineering.

Xu, B., Vafaee, F. and Wolfson, O. (2009). In-network query processing in mobile p2p databases, Proc. of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems.

Xu, B. and Wolfson, O. (2005). Data management in mobile peer-to-peer networks, Databases, Information Systems, and Peer-to-Peer Computing, Vol. 3367, pp. 1–15.

Xu, Y., Heidemann, J. and Estrin, D. (2001). Geography-informed energy conservation for ad hoc routing, Proc. of the 7th annual international conference on Mobile computing and networking.

Yang, B., Lu, H. and Jensen, C. S. (2010). Probabilistic threshold k nearest neighbor queries over moving objects in symbolic indoor space, Proc. of the 13th International Conference on Extending Database Technology, pp. 335–346.

Yin, J. and Cao, G. (2006). Supporting cooperative caching in ad hoc networks, IEEE Transactions on Mobile Computing **5**(1): 77–89.

Yoo, J., Shekhar, S. and Celik, M. (2005). A join-less approach for co-location pattern mining: a summary of results, Proc. of the Fifth IEEE International Conference on Data Mining.

Yu, X., Pu, K. and Koudas, N. (2005). Monitoring k-nearest neighbor queries over moving objects, Proc. of the 21st International Conference on Data Engineering.

Zhang, J., Mamoulis, N., Papadias, D. and Tao, Y. (2004). All-nearest-neighbors queries in spatial databases, Proc. of the 16th International Conference on Scientific and Statistical Database Management, Washington, DC, USA.

Zhang, J., Papadias, D., Mouratidis, K. and Zhu, M. (2004). Spatial queries in the presence of obstacles, in E. Bertino, S. Christodoulakis, D. Plexousakis, V. Christophides, M. Koubarakis, K. Bohm and E. Ferrari (eds), Advances in Database Technology - EDBT 2004, Vol. 2992 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp. 366–384.

Zhang, J., Zhu, M., Papadias, D., Tao, Y. and Lee, D. (2003). Location-based spatial queries, Proc. of the 2003 ACM SIGMOD international conference on Management of data.

Zhao, G. (2012). K nearest neighbour search in spatial databases, PhD thesis. Monash University.

Zhao, G., Xuan, K., Rahayu, W., Taniar, D., Safar, M., Gavrilova, M. and Srinivasan, B. (2009). Voronoi-based continuous k nearest neighbor search in mobile navigation, IEEE Transactions on Industrial Electronics **58**(6): 2247–2257.

Zhao, G., Xuan, K., Taniar, D., Safar, M., Gavrilova, M. and Srinivasan, B. (2009). Multiple object types knn search using network voronoi diagram, Computational Science and Its Applications, ICCSA 2009, Vol. 5593, pp. 819–834.

Zheng, B. and Lee, D. L. (2001). Semantic caching in location-dependent query processing, Proc. of the 7th International Symposium on Advances in Spatial and Temporal Databases, pp. 97–116.

Zhu, Q., Lee, D. and Lee, W. (2011). Collaborative caching for spatial queries in mobile p2p networks, Proc. of the 27th IEEE International Conference on Data Engineering, pp. 279–290.

Last Thing

"Ends are not bad things, they just mean that something else is about to begin. And there are many things that don't really end, anyway, they just begin again in a new way."

– C. JoyBell C.