

**PRIVACY-AWARE BIG DATA SYSTEMS AND SERVICES IN THE INTERNET  
OF THINGS ERA**

A Dissertation  
Presented to  
The Academic Faculty

By

Emre Yigitoglu

In Partial Fulfillment  
of the Requirements for the Degree  
Doctor of Philosophy in the  
School of Computer Science

Georgia Institute of Technology

May 2018

Copyright © Emre Yigitoglu 2018

**PRIVACY-AWARE BIG DATA SYSTEMS AND SERVICES IN THE INTERNET  
OF THINGS ERA**

Approved by:

Dr. Ling Liu, Advisor  
School of Computer Science  
*Georgia Institute of Technology*

Dr. Margaret Loper, Co-Advisor  
Georgia Tech Research Institute  
*Georgia Institute of Technology*

Dr. Calton Pu  
School of Computer Science  
*Georgia Institute of Technology*

Dr. Shamkant Navathe  
School of Computer Science  
*Georgia Institute of Technology*

Dr. Greg Eisenhauer  
School of Computer Science  
*Georgia Institute of Technology*

Dr. Jinpeng Wei  
Department of Software and Information Systems  
*University of North Carolina at Charlotte*

Date Approved: March 26, 2018

*To my lovely wife, Askin, and my family.*

## ACKNOWLEDGEMENTS

I would like to extend my deepest gratitude to my advisor Professor Ling Liu for her guidance, instructive comments, continues support, and patience during my doctoral study. I am indebted for the countless hours she has spent on perfecting my work and the immense amount of advice she has given me on research, career, and life related matters. She will undoubtedly continue to light my way in my future career.

I would like to thank Dr. Margaret Loper for broadening my horizons through her remarkable comments as my co-advisor. I feel very fortunate to have her share her knowledge and experience with me during my graduate work.

I would also like to thank my committee members Dr. Calton Pu, Dr. Shamkant Navathe, Dr. Greg Eisenhauer, and Dr. Jinpeng Wei for being very supportive of my research and for their constructive critiques greatly contributed to my dissertation.

I am amazed at the dedication and quality of the work performed by many members of the DiSL research group. Completing this work would have been more difficult were without the support and friendship provided by Emre Gursoy, Abdurrahman Yasar, Alper Yildirim, Qi Zhang, Kisung Lee, and Balaji Palanisamy.

I would like to thank all my internship mentors, Peter Pesti, Nitin Agrawal, and Mohammed Mohammed. Their impact on my research perspective has definitely helped me in raising the quality of my work, and gaining valuable work experience.

This dissertation would not have been possible without the support and nurturing of my parents, who have always trusted my choices and understand I had to travel so far to find inspiration. I would also like to thank my brother, sister and friends back home.

This last word of acknowledgment I have saved for my dear wife Askin Guler Yigitoglu, who has been with me all these years and has made them the best years of my life.

## TABLE OF CONTENTS

<b>Acknowledgments</b> . . . . .	v
<b>List of Tables</b> . . . . .	x
<b>List of Figures</b> . . . . .	xi
<b>Chapter 1: Introduction</b> . . . . .	1
1.1 Motivation . . . . .	1
1.2 Dissertation Scope and Contributions . . . . .	5
1.2.1 Information Service Orchestration . . . . .	5
1.2.2 Application Service Deployment . . . . .	5
1.2.3 Privacy Protection . . . . .	6
1.3 Dissertation Organization . . . . .	7
<b>Chapter 2: Distributed Orchestration in Large-scale IoT Systems</b> . . . . .	8
2.1 Introduction . . . . .	8
2.2 System Model . . . . .	13
2.2.1 IoT Computing Reference Model . . . . .	13
2.2.2 System Assumptions . . . . .	14
2.3 Distributed Orchestration Architecture . . . . .	16

2.4	Optimizations . . . . .	18
2.5	Experiments . . . . .	21
2.5.1	Server Side Computation Cost . . . . .	22
2.5.2	Object Side Computation Cost . . . . .	22
2.5.3	Communication Cost . . . . .	24
2.6	Related Work . . . . .	24
<b>Chapter 3: Foggy: A Framework for Continuous Automated IoT Application Deployment in Fog Computing . . . . .</b>		
<b>25</b>		
3.1	Introduction . . . . .	25
3.2	Motivation and Background . . . . .	27
3.2.1	IoT Infrastructure . . . . .	27
3.2.2	Requirements . . . . .	29
3.3	Automated IoT Application Update and Deployment System . . . . .	31
3.3.1	IoT Applications . . . . .	31
3.3.2	System Components . . . . .	34
3.3.3	IoT Application Deployment and Update Workflow . . . . .	37
3.3.4	Planning and Provisioning Procedure . . . . .	37
3.3.5	Container Update Model . . . . .	39
3.4	Implementation . . . . .	40
3.5	Related Work . . . . .	42
<b>Chapter 4: Privacy-Preserving Location Queries for Mobile Travelers: A Utility- Aware and Attack-Resilient Approach . . . . .</b>		
<b>44</b>		
4.1	Introduction . . . . .	45

4.2	<b>STARCLOAK Overview</b>	47
4.2.1	Road Network Model	49
4.2.2	Utility-aware Location Privacy Model	50
4.2.3	Anonymous Location-based Service Architecture	55
4.2.4	Anonymous Query Cost Model	57
4.2.5	Inference Attack Models	61
4.2.6	Baseline Anonymization Models	65
4.2.7	Solution Approach	67
4.3	<b>STARCLOAK: Model and Algorithms</b>	70
4.3.1	Star Structure and Star Graph	70
4.3.2	Data Structures	72
4.3.3	Location Query Preprocessing	73
4.3.4	StarCloak in a Nutshell	74
4.3.5	Select Star	76
4.3.6	Cloaking Graph Update	78
4.3.7	Candidate Star-Set Selection	84
4.3.8	Star-Set Pruning	87
4.4	<b>Optimizations</b>	89
4.4.1	Spatially Bounded StarCloak	89
4.4.2	Hybrid StarCloak	92
4.5	<b>Experimental Evaluations</b>	92
4.5.1	Experimental Setup	93
4.5.2	Evaluation Metrics	94

4.5.3	Experimental Results . . . . .	95
<b>Chapter 5: PrivacyZone</b>	. . . . .	<b>106</b>
5.1	Introduction . . . . .	106
5.2	System Overview . . . . .	110
5.2.1	Concept . . . . .	110
5.2.2	Architecture . . . . .	112
5.2.3	Road Network Model . . . . .	113
5.2.4	Privacy Zones . . . . .	115
5.2.5	Privacy Model . . . . .	115
5.2.6	Attack Models . . . . .	117
5.3	Privacy Zone Processing . . . . .	120
5.3.1	Motivating Approaches . . . . .	120
5.3.2	Hibernation Period-Based Monitoring . . . . .	121
5.3.3	Hibernation Region-Based Monitoring . . . . .	125
5.4	Experimental Evaluation . . . . .	129
5.4.1	Evaluation Setup . . . . .	129
5.4.2	Experimental Results . . . . .	130
<b>Chapter 6: Conclusion</b>	. . . . .	<b>138</b>
6.1	Future Directions . . . . .	139
<b>References</b>	. . . . .	<b>146</b>

## LIST OF TABLES

4.1	Default parameter setting for query generation. . . . .	94
5.1	Road network properties. . . . .	129

## LIST OF FIGURES

2.1	IoT Computing Architecture . . . . .	10
2.2	Distributed Orchestration in ISYMPHONY (Self-Driving Cars) . . . . .	14
2.3	Dead Reckoning . . . . .	19
2.4	Safe period optimization for query evaluation . . . . .	20
2.5	Safe period optimization for partition evaluation . . . . .	21
2.6	Experimental results of ISYMPHONY . . . . .	23
3.1	IoT Infrastructure . . . . .	28
3.2	Foggy Framework . . . . .	31
3.3	Video Surveillance System . . . . .	32
4.1	A road network model. . . . .	48
4.2	Overall architecture of STARCLOAK. . . . .	55
4.3	Illustration of query processing on road networks. . . . .	59
4.4	Query Injection Example . . . . .	65
4.5	Baseline location-anonymization models. . . . .	66
4.6	Illustration of the STAR model. . . . .	71
4.7	StarCloak Example . . . . .	74
4.8	Cloaking Graph Update Example . . . . .	83

4.9	Pruning Example . . . . .	88
4.10	Spatially Bounded StarCloak Example . . . . .	90
4.11	Average traffic in downtown Atlanta. The color codes red, orange, yellow and green represents relative traffic speeds with stop and go, slow, moderate and free flow, respectively. . . . .	91
4.12	Query execution cost (in terms of the average query processing time per query) and communication cost (in terms of the average size fo candidate result per query) with respect to varying settings of parameters $\delta_k$ , $\delta_s$ , $\sigma_s$ and $k$ . . . . .	96
4.13	Success rate for California Map . . . . .	98
4.14	Success rate for Georgia Map . . . . .	98
4.15	Anonymization time for California Map . . . . .	100
4.16	Anonymization time for Georgia Map . . . . .	101
4.17	Successful throughput for California Map . . . . .	102
4.18	Successful throughput for Georgia Map . . . . .	103
4.19	Average entropy for California Map . . . . .	104
4.20	Average entropy for Georgia Map . . . . .	104
5.1	Demo: Create Privacy Zone . . . . .	111
5.2	User alert example . . . . .	112
5.3	System architecture . . . . .	112
5.4	Star and Rectangular Shape Privacy Zones . . . . .	114
5.5	Privacy Zones for all religious buildings . . . . .	116
5.6	Minimum Travel Time-based Attack . . . . .	118
5.7	Markov model-based transition attack . . . . .	120
5.8	Euclidean vs Road Network Shortest Path-based Hibernation Period . . . . .	122

5.9	Euclidean lower bound-based filtering . . . . .	123
5.10	Shortest Distance vs Rectangle-shaped Hibernation Region . . . . .	127
5.11	Calculating rectangle shape hibernation region . . . . .	128
5.12	Road networks used in experiments . . . . .	129
5.13	Hibernation time comparison . . . . .	131
5.14	Number of wake-up comparison . . . . .	131
5.15	Number of location update to the server comparison . . . . .	132
5.16	Server side processing time comparison . . . . .	133
5.17	Client side processing time comparison . . . . .	134
5.18	Privacy protection . . . . .	135
5.19	Effect of different road networks . . . . .	136
5.20	Deciding right alpha value . . . . .	137

## SUMMARY

The Internet of Things (IoT) has been considered to be the next industrial revolution. Connecting physical objects to the internet and equipping things with the powerful data analytic capabilities, the IoT will transform our everyday life from work to travel, leisure, to entertainment. It is expected that by 2025 the number of connected devices will reach 100 billion with more than \$11 trillion global economic impact. At the same time, the large scale IoT deployment will be comprised of a huge number of interconnected and heterogeneous wireless and mobile devices, enabling continuous monitoring of activities and events and happenings in our environments and our society. We argue that the large scale IoT deployment demands privacy-aware big data systems and services that can deliver real-time and life enhancing experiences while respecting both data privacy and user privacy. This dissertation research contributes original ideas and innovative techniques in the context of Internet of Things from three perspectives: information service orchestration, application service deployment, and privacy protection.

The first contribution of this dissertation research is the development of ISYMPHONY, a distributed IoT service orchestration architecture for scaling real-time and on-demand IoT services provisioning in large scale IoT systems while guaranteeing the quality of service with respect to performance, availability, and security. The main idea is to provide intelligent partition of a real time IoT computing task into an optimal coordination between the server side processing at IoT gateway and smart edge client side processing on edge devices. This computation partitioning and data partitioning architecture enable edge devices and edge clients running on top of edge devices to contribute to the IoT computing tasks based on their computing and communication capacities.

The second contribution of this dissertation research is the design and development of FOGGY, a framework for continuous automation of IoT application deployment in the three layer (edge-gateway-Cloud) IoT environment. The core idea of FOGGY is to facilitate

dynamic resource provisioning and to automate application deployment in next generation IoT infrastructure, enabling data processing to be performed at or close to where the data is generated.

The third technical contribution of this dissertation research is the design and implementation of privacy-preserving models and algorithms for enabling privacy-aware IoT systems and services. Concretely, we developed two systems: StarCloak and PrivacyZone. StarCloak is a query utility-aware, privacy-preserving, and attack-resilient model and a suite of algorithms for mobile objects in the spatially constrained environments, such as moving on the road networks. A comprehensive performance study shows that StarCloak outperforms existing anonymization techniques on both privacy protection and service performance. PrivacyZone is the first spatial trigger based systems to establish personalized privacy quarantine areas in the physical or cyber space to enable objects and people to maintain spatial and temporal privacy at a selected area of interests at the selected time interval.

# CHAPTER 1

## INTRODUCTION

The Internet of Things (IoT) has been considered the next industrial revolution. Connecting physical objects to the Internet and equipping massive sensor generated data with the powerful data analytics capabilities, the IoT holds the potential to transform our everyday life from work, travel, leisure, to entertainment. Smart cities, self-driving vehicles, health monitoring systems are just a few applications of the IoT to name. Because of the numerous opportunities that IoT provides, the number of connected devices is increasing rapidly. It is expected that by 2025 the number of connected devices will reach 100 billion with more than \$11.1 trillion global economic impact [1, 2]. This increasing number of smart things will generate huge amounts of data that need to be collected, processed and stored. Cisco predicts that in 2020, the total amount of data created by any device will reach 600 ZB per year [3]. At the same time, the large scale IoT deployment comprised of a huge number of interconnected and heterogeneous wireless and mobile devices, enabling continuous monitoring of activities, events and happenings in our environments and our society. We argue that the large scale IoT deployment demands privacy-aware big data systems and services that can deliver real-time and life enhancing experiences while respecting both data privacy and user privacy. This dissertation research contributes original ideas and innovative techniques in the context of Internet of Things from three perspectives: information service orchestration, application service deployment, and privacy protection.

### 1.1 Motivation

Intelligent and scalable orchestration of massive IoT objects using a multi-tier architecture is critical to embrace the vision of IoT. In conventional client-server systems, clients are primarily sending requests and receiving results and are considered too thin to participate

in computation tasks. However, as the growth rate of the IoT devices continues to escalate, and the capacity of wireless devices continues to sprout, in order to scale IoT services provisioning in real time, we need to revolutionize the conventional client-server computing architecture. Furthermore, we argue that the conventional server-only architectures may not be able to meet the requirements of latency-sensitive IoT applications. For example, it is not realistic to send all of the data generated by autonomous cars (40GB per hour) to the centralized Cloud and expect real-time decision. We argue that an effective orchestration framework for large scale IoT system should leverage the computational capacity of smart things.

Deploying, managing, and updating IoT applications on a multi-tier architecture introduce new challenges. First of all, computational partition should be dynamic and adaptive such that it can leverage the capacity of edge devices by taking into account of multidimensional factors, such as data locality, resource capacity and workload demand at all IoT orchestration tiers: edge devices, midway servers, and Cloud data centers. The first research challenge is to develop the system-level facilities for optimizing the performance of edge gateway for scaling distributed orchestration to large number of IoT objects. We need resource-aware allocation models that can dynamically schedule and allocate resources, such as containers, among edge clients running on the same edge device, to enable on-demand execution of edge-client specific tasks with high throughput and low latency. Similarly, we need workload-aware resource scheduling of multiple services provisioning tasks running concurrently on the same midway server(s), while taking into account of the object side processing workloads at each edge gateway. The second research challenge is the resource aware selection of computation and execution environments for a collection of IoT service provisioning requests. This involves the decision on how and what to offload from edge devices to a network of IoT midway stations (servers) and/or to an IoT application specific cloud data center. Another problem related to the second challenge is the placement of midway server nodes and the overlay network management of these servers.

The ubiquitous wireless connectivity, the universal presence of smart mobile devices with multi-modal sensing capability, and the increased investments from industry and government on the Internet of Things (IoT) have fueled the growth of the Location-based services (LBS). Juniper Research [4] forecasts that the LBS market will reach \$43.3 billion in revenue by 2019, rising from an estimated \$12.2 billion in 2014. [5] reports that %74 of adult smartphone owners use their phones to get direction or information based on their current locations. As more and more mobile travelers and vehicles are connected continuously and automatically, and as we are marching towards the grand vision of Internet of Everything, we are embraced by many life-enriching location-based experiences and services (e.g., improved emergency assistance, real-time traffic alerts). However, for many location-based services and mobile data management applications, location queries are the fundamental functionality and continuous location exposure opens doors to the intrusion of location privacy [6]. For example, even if users identifiers are removed from their location queries, successive position updates can be linked to identify individuals with high confidence. Unauthorized trajectory exposure may also expose mobile users of LBSs to significant vulnerabilities for abuse such as unwanted advertisement, stalking, and location spoofing. In addition, when private location data of a mobile user is linked to sensitive public locations such as mental health clinics, cancer treatment centers, nightclubs, and religious organizations, such unauthorized location - identity linkage may cause ethical, professional, and social risks to both individuals and our society at large. The third research challenge of this dissertation is to investigate and develop privacy preserving location service architectures with innovative techniques and systems for protecting both user privacy and location privacy while delivering location based IoT services.

Over the last decade, research on location privacy has gained increased attentions. The most representative location privacy preserving technique, location *k-anonymization*, hide the exact location of a mobile user by replacing it with a *k*-anonymized cloaking region in user's location query, which guarantees that at least *k* mobile users share the same cloaked

region as their published location. A subject is considered *location  $k$  - anonymous* if its location is indistinguishable from  $k - 1$  other users' location [6, 7, 8, 9, 10]. However, location  $k$ -anonymization algorithms suffer from a number of known problems: (1) Most existing algorithms compute the anonymized location solely based on specific location-privacy metric or metrics. Very few of these algorithms has incorporated location utility and location query cost into consideration in their spatial cloaking decisions. (2) Most existing location privacy algorithms are vulnerable to complex replay attacks such as correlation- and query injection-based attacks. (3) Most existing solutions develop spatial obfuscation techniques based on the *random waypoint* mobility model [11, 12], in which mobile travelers move in arbitrary directions at random speed. These solutions fail to address privacy risks when mobile users traveling in a spatially constrained environment such as a road network in which both user mobility and location query processing are constrained by the underlying road network. For example, as a spatial cloaking region may contain a single road segment when we use rectangle-based spatial cloaking techniques, they are vulnerable because adversaries can easily track the whereabouts of mobile users.

Furthermore, the existing privacy protection techniques based on spatial cloaking achieve privacy at high cost of location utility loss and quality loss of location based services (LBSs). This is because the exact location is typically positioning point. Thus, many existing location-based applications (e.g., Apple App store, Google play) are designed to accept focal location as an exact GPS coordinate instead of a spatial region. One can reduce the spatial resolution of cloaking by either minimizing the cloaking region or by not cloaking every positioning location at the same protection granularity. We argue that the capability for differentiation of sensitive locations from less or non-sensitive locations may significantly improve the utility of SBSs while preserving location and user privacy. Another problem with conventional spatial cloaking based location privacy solutions is the inherent limitation and lack of robustness for protecting privacy when providing continuous location monitoring services, such as step-counter applications (e.g., Runkeeper). Well known

vulnerabilities include consecutive region based inference attacks, replay attack, transition attacks, to name a few.

## **1.2 Dissertation Scope and Contributions**

This dissertation makes the following contributions to address the technical challenges described above.

### 1.2.1 Information Service Orchestration

The first contribution of this dissertation research is the development of ISYMPHONY, a distributed IoT service orchestration architecture for scaling real-time and on-demand IoT services provisioning in large scale IoT systems while guaranteeing the quality of service with respect to performance, availability, and security. The main idea is to provide intelligent partition of a real time IoT computing task into an optimal coordination between the server side processing at IoT gateway and smart edge client side processing on edge devices. This computation partitioning and data partitioning architecture enable edge devices and edge clients running on top of edge devices to contribute to the IoT computing tasks based on their computing and communication capacities.

### 1.2.2 Application Service Deployment

The second contribution of this dissertation research is the design and development of FOGGY, a framework for continuous automation of IoT application deployment in the three layer (edge-gateway-Cloud) IoT environment. The core idea of FOGGY is to facilitate dynamic resource provisioning and to automate application deployment in next generation IoT infrastructure, enabling data processing to be performed at or close to where the data is generated. We analyze several applications and identify their requirements that need to be taken into consideration in our design of the FOGGY framework that manages resources and deploys IoT applications with minimal developer efforts.

### 1.2.3 Privacy Protection

The third technical contribution of this dissertation research is the design and implementation of privacy-preserving models and algorithms for enabling privacy-aware IoT systems and services. Concretely, we developed two systems: StarCloak and PrivacyZone.

StarCloak is a query utility-aware and attack-resilient location privacy model and a suite of road-network aware location cloaking algorithms. By utilizing the movement and motion behavior of mobile clients on the spatially constrained road networks, we develop a suite of location cloaking techniques that minimize the location utility losses and maximize the attack resilience. Through a comprehensive performance evaluation, we show that StarCloak outperforms existing anonymization techniques in terms of both location privacy protection and location service performance.

PrivacyZone is the first location privacy protection system based on the concept of privacy zones and the execution model of spatial triggers. We introduce the concept of privacy zones for dual purposes: First, privacy zones can be used to enable mobile users to differentiate their location queries or location service requests when they are residing or moving to sensitive locations or spatial regions of interest. Second, privacy zones are introduced to improve the overall utility and service quality of location queries and LBSs. Unlike the conventional approaches that rely on spatial cloaking of every focal location of location queries, the use of privacy zones allow mobile users to selectively establish their personalized privacy quarantine areas in either physical space (such as on road networks) or cyber space (such as on social networks). By enforcing privacy protection for users entering their pre-defined privacy zones, we can provide high degree of privacy protection while maintaining high degree of service quality and location utility, while maintaining spatial and temporal privacy at both selected area of interests and selected time interval of interest. In order to minimize the battery consumption for continuous monitoring of whether mobile objects are entering their privacy zones, we design and implement a suite of energy efficient PrivacyZone enforcement algorithms, which provide real-time spatial

and temporal alerts for users who enter or approach their personalized privacy-zones and which enforce privacy protection for those users residing in their privacy zones. In addition, we designed a novel PrivacyZone based mobile permission system, which provides mobile users more flexibility to enforce their privacy permissions, while minimizing their efforts of defining privacy zones of interest. Our experimental results demonstrate the utility and privacy protection of PrivacyZone in real road network environments.

### **1.3 Dissertation Organization**

The rest of this dissertation is organized into four technical chapters and conclude in Chapter 6. Each chapter addresses one or more of the problems described above. In each chapter, we introduce the background of the problem being addressed, describe related work, and present our solution techniques followed by experimental evaluation. Concretely, in Chapter 2, we present our vision and our initial development of a distributed orchestration framework, ISYMPHONY, with the ultimate-goal of scaling real-time and on-demand IoT services provisioning in large scale IoT systems, while guaranteeing quality of service with respect to performance, availability and security. In Chapter 3, we introduce Foggy framework that facilitates dynamic resource provisioning and automated application deployment in Fog Computing architectures. In Chapter 4, we present STARCLOAK, a utility-aware and attack-resilient cloaking-based location privacy-preserving approach for mobile travelers on road networks. In Chapter 5, we introduce the novel concept of PrivacyZone and a suit of algorithms for constructing personalized privacy-zones and enforcing privacy-zone protection for mobile users on the roads. We summarize the main contributions of this dissertation and discuss our future research directions in Chapter 6.

## CHAPTER 2

### DISTRIBUTED ORCHESTRATION IN LARGE-SCALE IOT SYSTEMS

With the growing popularity of smart things and the omnipresence of wireless communications, the Internet continues to revolutionize from the Internet of hosts, to the Internet of People, to the Internet of Things (IoT). Intelligent and scalable orchestration of massive IoT objects using a multi-tier architecture are critical to embrace the vision of IoT. In this work, we present our vision and our initial development of a distributed orchestration framework, called ISYMPHONY, with the ultimate-goal of scaling real-time and on-demand IoT services provisioning in large scale IoT systems, while guaranteeing quality of service with respect to performance, availability and security. This work makes two original contributions. First, we present a distributed orchestration architecture that enables edge devices and edge clients running on top of edge devices to contribute to the IoT computing tasks based on their computing and communication capacities. A main idea behind our distributed IoT orchestration architecture is the intelligent partition of a real time IoT computing task into an optimal coordination of server-side processing and IoT object side processing. Second, a set of optimization techniques are introduced to limit the amount of computations handled by the edge clients and to enhance the overall performance and resource utilization of the ISYMPHONY system. Our experimental results show that ISYMPHONY can lead to significant saving in terms of server load and high accuracy by leveraging and coordinating edge client processing capabilities, compared to the solutions relying solely on server level processing for real time IoT services provisioning.

#### 2.1 Introduction

With the growing popularity of smart things and the omnipresence of wireless communications, we witness the continued revolution of the Internet from the Internet of hosts

interconnected via TCP-IP protocol, to the Internet of people interconnected via social networks, to the Internet of smart things interconnected via multi-tier communication overlays. Internet of everything (IoE) will eventually become a reality when everything in physical space can be powered on and connected to the Internet, enabling every object in the physical space to interact with the cyberspace at any time and any place.

In large scale IoT application systems, such as smart cities, smart buildings, self-driving cars, we observe two unique characteristics, compared to conventional computing systems: (i) the number of smart objects in the form of edge devices or edge clients running on an edge device is massive; and (ii) many of these edge devices are fat clients with computation and communication capacity equivalent to traditional laptop and desktop computers. As the growth rate of fat edge clients continues to escalate, and the capacity of edge devices continues to sprout, we need to revolutionize the conventional client-server computing architecture in order to scale IoT services provisioning in real time. Concretely, in conventional client-server systems, clients are primarily sending requests and receiving results and are considered too thin to participate in both client-initiated or server initiated computation tasks. An example of client-initiated IoT computing task is to request the neighborhood traffic information by a moving vehicle. A typical server initiated task is to collect information on traffic density at multiple geographical regions of a city to provide an overall picture of the traffic situations in the major roads of a city.

In this work, we argue that an effective orchestration framework for large scale IoT system should address both vertical orchestration and horizontal orchestration. The former centers on orchestrating the vertical stack of the IoT computing and communication, ranging from hardware, device firmware, computer networking and software defined networks, server virtualization and cloud computing. The latter focuses on leveraging the computational capacity of smart things at the edge of the IoT computing infrastructure. We present our vision and our initial design and development efforts towards the design and implementation of a distributed IoT orchestration architecture. A main idea behind this architecture

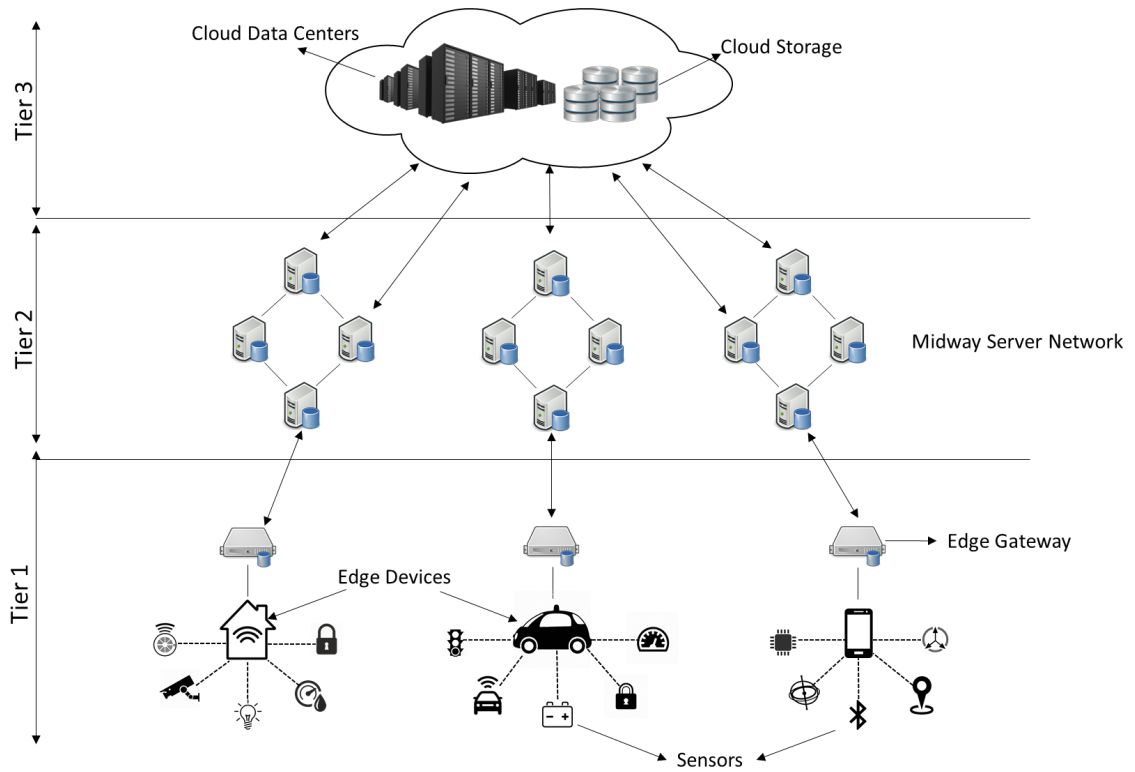


Figure 2.1: IoT Computing Architecture

is the intelligent partition of a real time IoT computing task into an optimal coordination of server-side processing and IoT object side processing. Such partition offers dual benefits: (i) it allows scaling real time IoT service provisioning with a high degree of precision when IoT objects are on the move; and (ii) it enables resource aware and workload adaptive distribution of computation through careful coordination of multi-tier IoT orchestration architecture. We envision that such computational partition should be dynamic and adaptive such that it can leverage the capacity of edge devices and fat clients running on top of such edge devices by taking into account of multi-dimensional factors, such as data locality, resource capacity and workload demand at all three IoT orchestration tiers: edge gateways (edge devices), midway servers (e.g., fog nodes [13]), and cloud data centers. Figure 2.1 shows a sketch of the three tier IoT computing architecture.

In our approach, we treat smart IoT objects as a first-class computation player and treat an edge device as a first-class IoT execution environment. For presentation convenience,

in the rest of the chapter we use edge clients and IoT objects interchangeably and use edge devices and edge gateways interchangeably when no confusion occurs. We envision three main challenges in making edge clients as the first-class players in an IoT computational model and making edge devices as the first class and an integral component in the execution environment of an IoT services provisioning system.

The first challenge is the locality and workload aware computation partitioning between midway servers and edge gateways and between edge gateway and edge clients running on the corresponding edge device. Each edge device has an edge gateway to coordinate the deployment and communication requests of its edge clients running on the respective edge device. For example, an iPhone as an edge device may install an edge gateway module of ISYMPHONY, which will coordinate those iPhone applications to transport their sensed data in either raw or aggregated form to the midway servers and/or the respective cloud data centers. Similarly, a self-driving car as a powerful edge device can have an ISYMPHONY edge gateway to coordinate and manage all sensor readings collected from different types of built-in sensors, such as brake, engine, front, back, side camera and so forth. On each midway server, multiple IoT services provisioning tasks submitted by different edge gateways may be running concurrently. Moreover, on each edge gateway, IoT service requests from multiple edge clients may be running concurrently. For example, Googlemap, iMessage, Step counter applications, and music applications may be running concurrently on an iPhone when this iPhone user is on the move. An intelligent task partitioning mechanism can divide a service provisioning task into server-side processing and object-side processing components to enable real-time services provisioning with high scalability.

The second challenge is to develop the system-level facilities for optimizing the performance of edge gateway for scaling distributed orchestration to large number of IoT objects. We need resource-aware allocation models that can dynamically schedule and allocate resources, such as containers, among edge clients running on the same edge device, to enable on-demand execution of edge-client specific tasks with high throughput and low latency.

Similarly, we need workload-aware resource scheduling of multiple services provisioning tasks running concurrently on the same midway server(s), while taking into account of the object side processing workloads at each edge gateway.

The third challenge is the resource aware selection of computation and execution environments for a collection of IoT service provisioning requests. This involves the decision on how and what to offload from edge devices to a network of IoT midway stations (servers) and/or to an IoT application specific cloud data center. Another problem related to the third challenge is the placement of midway server nodes and the overlay network management of these servers.

In this work, we focus on the first challenge. We present our vision and our initial development of a distributed orchestration framework, called ISYMPHONY, with the ultimate-goal of scaling real-time and on-demand IoT services provisioning in large scale IoT systems. The design of ISYMPHONY is novel from a number of perspectives. First, ISYMPHONY promotes a distributed orchestration architecture. A main idea behind our distributed architecture is the intelligent partition of a real time IoT computing task into an optimal coordination of server-side processing and object-side processing, especially for those moving IoT objects. Second, we develop a road network-based partitioning mechanism that considers the density of each segment in order to minimize the number of inter-partition changes. Third, a set of optimization techniques are employed to limit the amount of computations handled by the edge clients and to enhance the overall performance and resource utilization of the ISYMPHONY system. Last but not the least, in ISYMPHONY, a network of IoT midway servers is maintained dynamically and collaborating in a decentralized manner. This network of ISYMPHONY gateway servers collectively manage the IoT objects that are distributed over a geographical universe of discourse through the edge gateway modules running on edge devices. Each of our ISYMPHONY midway servers is responsible for a geographical region of interest, and in charge of managing and interacting with the edge devices and edge clients residing in the corresponding geographical region.

## 2.2 System Model

### 2.2.1 IoT Computing Reference Model

In the large scale IoT systems managed by ISYMPHONY, we envision three tier IoT computing architecture with cloud data center at the top tier, midway servers at the middle tier and edge gateways at the bottom tier, as illustrated in Figure 2.1. An edge gateway consists of the edge computing modules of ISYMPHONY, which manages the data collection from sensors on edge devices, in-edge aggregation and transport data in raw or aggregated form to the IoT gateway server(s) at the middle tier. IoT cloud as the top tier. Each edge gateway manages sensors enabled on one edge device, such as a self-driving car and its sensors, or an iPhone and its motion sensor or GPS or Wifi positioning sensor. A midway server will interact with the set of edge gateways residing in the geographical region it is responsible for. A failover facility will be enabled upon the pre-defined connection timeout to take over the responsibility of the failed server. The cloud datacenter can request in-transit data aggregation from midway servers or raw data collections from a selection of edge gateways.

We assume that an IoT service request can come from any of the three tiers. However, the requests from cloud tier and gateway tier are typically related to data summarization or data aggregation tasks, whereas the requests from edge clients running on edge devices are typically location-based content queries, such as *”Tell me the traffic situation, such as density of vehicles on the road or the average travel time in the last 10 minutes, on the nearby road segments or road intersections within 5-mile radius”*. We refer to such edge client service request as edge query and we refer to those moving vehicles as IoT objects (or edge clients when edge devices are involved). An edge query is typically targeted on IoT objects which can be in motion or static objects nearby a moving car or a pedestrian in motion, such as gas stations, restaurants, hospitals. In this work, we will focus on how to scale the IoT services requests in the presence of a large number of IoT objects in motion. To

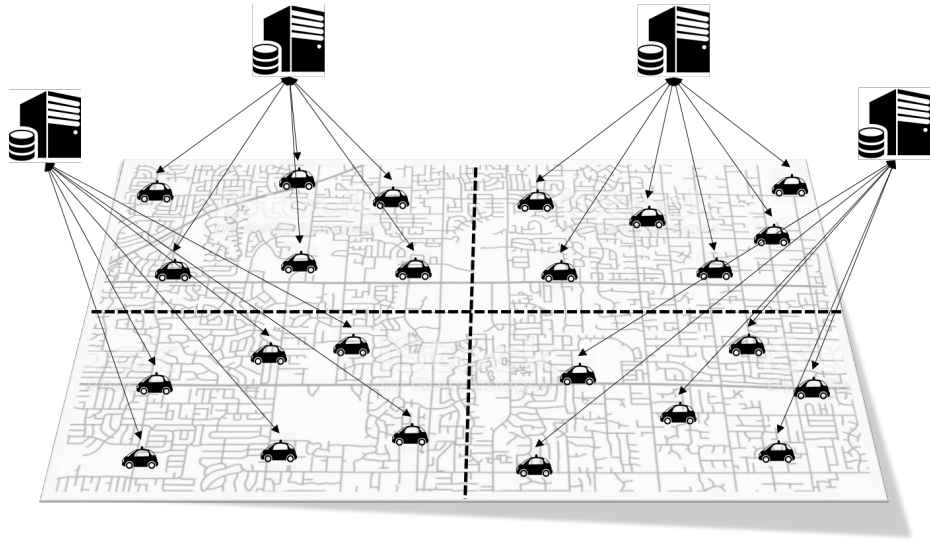


Figure 2.2: Distributed Orchestration in ISYMPHONY (Self-Driving Cars)

simplify the discussion of the system design requirements, execution model and optimization techniques, we focus on self-driving cars as an example IoT application and restrict the type of IoT services requests to be range-based location queries, such as nearby traffic flows or nearby accidents of a moving car. Figure 2.2 uses the scenario of self-driving cars to illustrate the distributed orchestration in ISYMPHONY.

### 2.2.2 System Assumptions

The design of the ISYMPHONY system is based on several assumptions. Most of these assumptions are widely agreed upon by many or have been seen as common practice in monitoring and tracking of moving objects [14].

First, IoT objects are smart objects. They are able to locate their position and determine their velocity vector, e.g., using GPS or WiFi localization when on the move. This is a reasonable assumption as GPS is becoming inexpensive and are used widely in cars and other hand-held devices. Examples are users with iPhone or self-driving cars. For instance, a pedestrian with an iPhone may have multiple applications running to guide her walk path and entertain her during her exercise. We consider each application as an edge client running on the iPhone (edge device) of this user and each edge client may issue an IoT

service request. For example, googlemap on the iPhone may ask for a walk path, music application may search for a song of interest, or recommend a song to this user, and the step count application may plot the summary of steps or display the step counts of other friends on the same day or in the past week. Thus, we treat the user with iPhone as the IoT object and all edge client-based requests issued from the same user as the IoT service requests (queries) with the location of this user as the focal point of the query services. Similarly, in self-driving car scenarios, a smart car is an IoT object and also the edge device on which several sensors are installed and different edge clients (applications) are running on the edge device to collect and report data from different sensors, including camera in front, back and sides of the cars, motion sensor, temperature sensor, brake sensor, and so forth. These query service requests will be coordinated by the edge gateway running on the car object.

Second, we assume that an IoT object has synchronized clocks, e.g., using GPS or NTP [15]. Broadcast is used to establish connections from one IoT object to its nearby objects and from midway server to its edge gateways. For example, an IoT object in motion can communicate with its neighbor objects by broadcasting using RF channels, such as Bluetooth, and WiFi. The neighbor objects can be other moving cars or other mobile users or static active objects such as roadside sensors or smart buildings. We assume that IoT objects are smart things and they can learn about the surrounding environment such as restaurants, gas stations, other types of buildings and houses through geographical database of places or geographical database of static physical objects. These databases are optimized by the spatial index structures. We consider an object (such as a building) a smart object if it has some built-in capability to “see (detect) nearby moving objects and nearby surroundings.

Third, each IoT object has the computational capability to carry out the assigned computational tasks, assuming the assignment is capability-aware. However, the problem of resource allocation and resource scheduling in the presence of concurrent edge clients on

the same edge device and concurrent execution of multiple IoT service requests on the same midway server is considered beyond the scope of this work and is on the agenda for our future work.

In addition, we assume that the geographical area of interest is covered by a network of ISYMPHONY gateway servers. We assume that all IoT query requests issued from the IoT objects are served through the distributed coordination between the midway server and the set of edge gateway nodes. The IoT application specific data mining tasks are issued from either the cloud datacenter or some midway servers. We classify all IoT services provisioning requests into two categories: (i) IoT objects requesting services from other moving objects and (ii) IoT objects requesting information from static objects. Here IoT objects can be moving or static. Given that the second type of IoT services requests can be processed easily by utilizing spatial indexing over the database of static objects [14]. Thus, in this work we focus on describing how ISYMPHONY handles the first type of IoT services requests.

### **2.3 Distributed Orchestration Architecture**

We observe that in a typical IoT system, the IoT objects, such as self-driving cars, are typically interested in other nearby moving objects, regardless of the number of objects in the whole universe. Motivated by this observation, in our distributed orchestration architecture each IoT object determines by itself whether or not it should be included in the result of a location query issued by a nearby moving object. This approach minimizes the server load and communication network usage between edge gateway nodes and midway servers as moving objects that are not nearby of any location queries do not need to report their positions. In order to provide efficient distributed orchestration, we need to develop algorithms to carefully partition the processing cost between the IoT moving objects and midway servers.

Our orchestration platform is build on the partitioning of the geographical universe of

interest. Partitions are used to determine IoT objects that should participate to computation of a certain service request. Specifically, when an IoT object, such as self-driving car or mobile user, issue a range query, it sends following information to the midway server: location, velocity vector (direction and speed), object of interest, and range ( $r$ ) value. Then midway server identify partitions that are within  $r$  distance to the query focal object's current partition. The distance between two partitions is the smallest distance of any point in those partitions, assuming that an IoT object can move to any point in its current partition. Midway server install query on objects that are residing on those nearby partitions. After this point, IoT objects are responsible to identify if they are within the query range or not. IoT objects check all registered queries periodically by comparing their current location and query focal object's predicted location based on its last known location, time, and velocity vector. IoT objects notify midway servers in following situations: (i) object exits from its current partition, (ii) object enters/exits to/from any query range that they are responsible to monitor, (iii) there is a significant change in the object's velocity vector (query focal objects only). This architecture design is especially effective when the number of moving IoT objects in the geographical region of interest is large and the number of IoT queries is relatively small.

The efficiency of our distributed orchestration architecture is highly depended on the structure of pre-generated partitions since it affects the number of objects responsible to answer a certain query and the number of location update from objects to the server due to partition change. In order to minimize the number of inter-partition passes, unlike the related work [14] that use grid-based partitioning, we introduce road network partitioning mechanism. Because, most of the IoT mobile objects' movements are restricted with the underlying road network or walking paths (e.g., autonomous cars) and grid cells may intersect only a small portion of the road segment, which causes frequent partition change during object's trip. Since mobile objects need to monitor their stay in the current partition, objects should be able to check containment with low energy consumption. To this end, we

used the middle of road segment as a border point of the partition instead of junctions. Because, with edge-based partitioning, an object can identify the time of exit by considering single border point, however, junction-based partitioning require expensive polygon containment check for the object as some of the neighbor segments of the junction can be part of the current partition. In addition, we consider the density of segments and our algorithm priorities dense segments to be in the same partitions. Ordered segment queue based on their densities is generated before the partitioning algorithm starts. We also use a system defined distance threshold that limits the maximum diameter (i.e., the distance of any two points in the partition) to control the size of the partition. The algorithm for constructing road network partitions start by assigning each junction and half of its neighbor segments as initial partitions. Then one selects a segment from a queue and combine its neighbor partitions if the new partition satisfies predefined distance threshold. The algorithm continues until all segments in the queue evaluated. Figure 2.5 shows an example partition with grey segments. After generating all partitions one calculates the pairwise distance from each partition to all other partitions and stored in the midway servers in order to minimize search time to identify partitions within a range when an IoT object issued a query.

## 2.4 Optimizations

To provide efficient implementation of our distributed orchestration architecture for processing IoT service requests at both server side and IoT object side, we described two optimizations used in ISYMPHONY.

**Dead Reckoning.** In the real world environment, the velocity vector of the IoT mobile object changes constantly because of several reasons such as road condition, GPS error, or human factors. This cause excessive number of location update from query focal object to the midway server as it notifies other objects for each velocity change to maintain the correctness of the query result even though the change is not significant. In order to control the number of location update, we use a dead reckoning technique to measure the

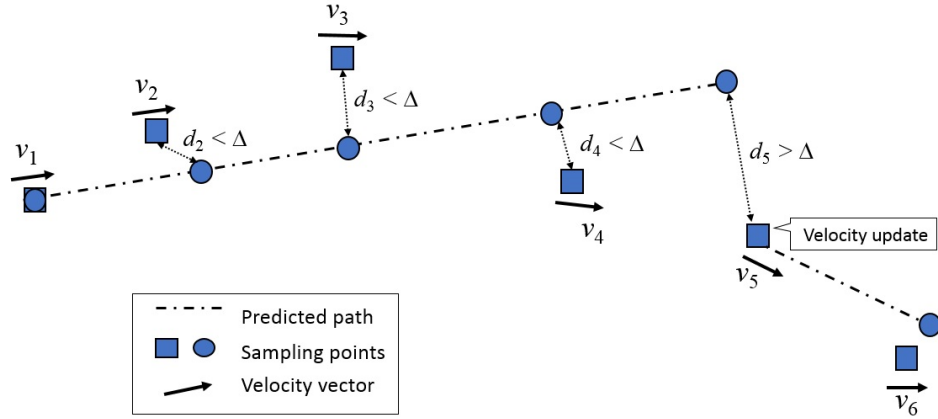


Figure 2.3: Dead Reckoning

significance of the velocity change. Concretely, at each time step, the focal object compare its current location to the location that it would reach if it continued to maintain its previously published velocity vector as other objects rely on that information to predict objects current location. If the distance is larger than the predefined threshold,  $\Delta$ , it updates its velocity vector to the midway servers which broadcast the change to the objects responsible for responding the focal object's query. Figure 2.3 illustrates dead reckoning optimization mechanism. After the object sends its current velocity vector,  $v_1$ , and location to the server, at each time step based on its speed and direction it identifies locations (blue circles) on the predicted path. The distance between object's exact locations (blue rectangles) at time  $t_2$ ,  $t_3$ , and  $t_4$  is smaller than the threshold  $\Delta$  and it is not necessary to update its velocity change to the server. However, when the distance between current location and the predicted location is larger than the threshold as the time step  $t_5$ , it updates its current location to the server and changes it predicted path based on new velocity vector.

**Safe Period.** An IoT object evaluates periodically set of queries that are registered by the midway server and its location to identify if it exits from its current partition. However, frequent evaluations introduce unnecessary computation load to the IoT objects especially when the inclusion of object to the certain query or partition will not change in the near future. A safe-period optimization applied to minimize the number of evaluation that IoT

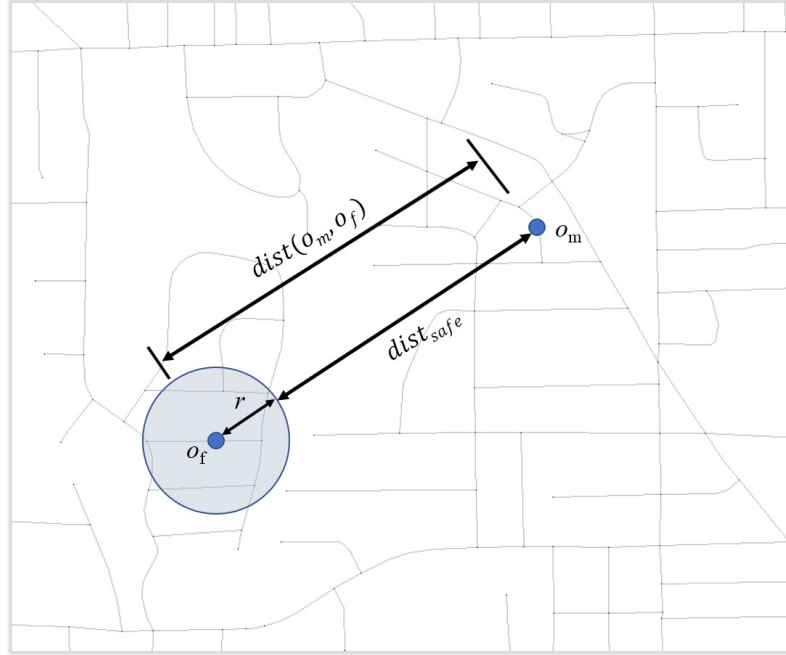


Figure 2.4: Safe period optimization for query evaluation

object needs to perform by introducing a hibernation time which guarantees even the object moves with maximum possible speed within the time interval, objects inclusion to a certain query or its current partition would not change.

We use safe-period optimization in two situations: query evaluation and partition evaluation. In the former case, an IoT mobile object identify hibernation time for each registered query by assuming the worst case scenario where both object move to each with maximum speed speed. In such scenario, the time required to pass for object to locate inside the area of query range is calculated as  $sp(o_m, o_f) = \frac{dist(o_m, o_f) - r}{max\_vel(o_m) + max\_vel(o_f)}$ , in which  $dist(o_m, o_f)$  denote the Euclidean distance from an object  $o_m$  to query focal object  $o_f$  and  $r$  is the distance of range query as illustrated in Figure 2.4.

Similarly, we use safe period optimization for partition evaluation where mobile object checks if its current partition changes. In this case, an IoT object identifies the closest border node from its current location and calculates hibernation time based on the distance and its maximum possible speed, since the border nodes are static in this case. Figure 2.5 shows an example of hibernation time calculation for partition evaluation where the objects cur-

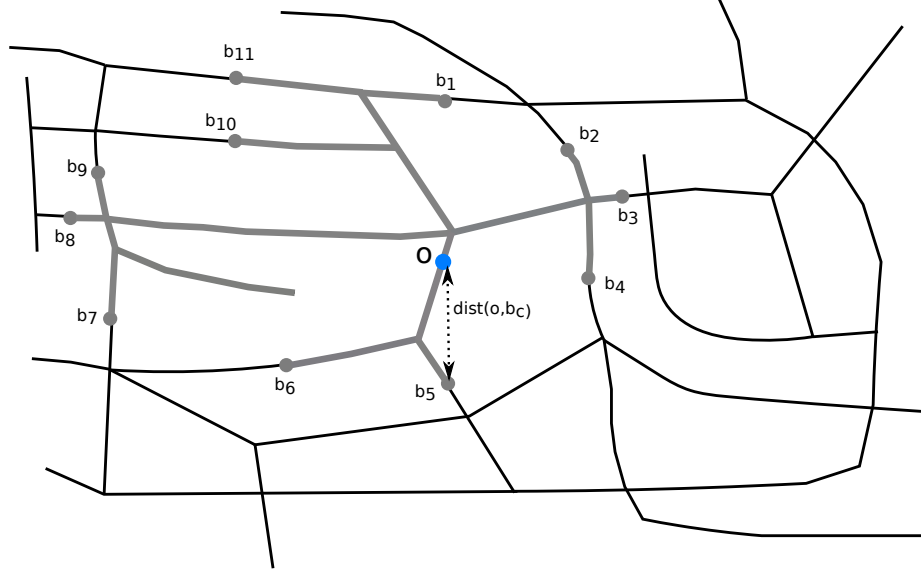


Figure 2.5: Safe period optimization for partition evaluation

rent partition depicted with gray lines. When the IoT objects wakes-up it finds the closest border node,  $b_5$  of the partition and calculates safe period based on their Euclidean distance as  $sp(o, b_c) = \frac{dist(o, b_c)}{max\_vel(o)}$ . As demonstrated in MobiEye [14], SpatialAlarm [16], and RoadAlarm [17], this optimization can be very effective to minimize computation requirement on the resource constraint devices.

## 2.5 Experiments

In this section, we conduct simulation-based experiments to evaluate our solution. We use GTMobiSim [18] with realistic traffic patterns to generate 20,000 of IoT objects moving on the road network of an urban area of Atlanta,GA which covers 11 km ( 6.8 miles) by 14 km (8.7 miles) area. The road network consist of four different road types: residential roads and freeway interchange with 30 mph speed limit (48 km/h), highway with 55 mph limit (89 km/h) and freeway with 70 mph limit (113 km/h). has 431 and 681 road segments having 70 mph and 55 mph speed limit respectively. The other road segments have 30 mph speed limit. The number of IoT services requests range from 20 to 200. We randomly select focal objects of the queries using a uniform distribution. The spatial region of a

query is taken as a circular region whose radius is a random variable following a normal distribution.

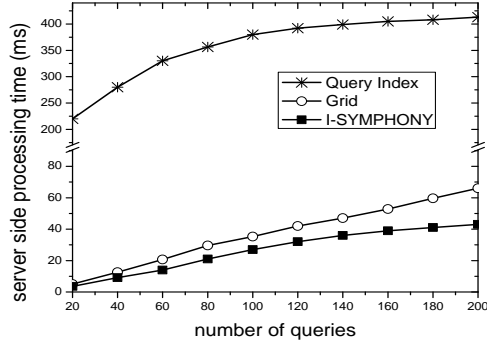
We compare our distributed query processing approach with popular central query processing approach that uses R\*-Tree [19] to index queries. When a new object position is received, it runs through the query index to determine to which queries this object actually contributes. Then, the object is added to the results of these queries and is removed from the results of other queries that have included it as a target object before. We also compare our approach with grid-partitioning-based mobile monitoring system introduced at [14]. All experiments are performed on a Windows 7 platform with Intel(R) Core(TM) CPU (4.00 GHz) and 16GB memory

### 2.5.1 Server Side Computation Cost

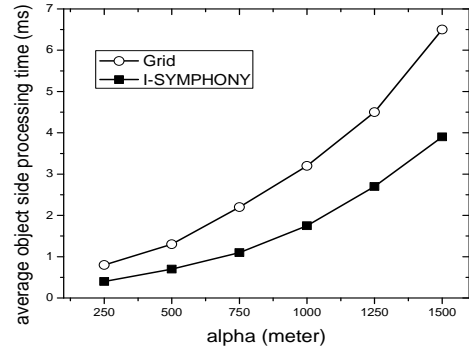
First, we measure the server load as the time required to execute server-side processing for the period of the simulation run with respect to the number of queries. Both grid and our road network partitioning-based distributed orchestration architecture provides significant improvement on server load. The main reason is that the query index approach needs to update the spatial index for every location change of the query focal objects. Also, our road network-based partitioning mechanism outperforms the simple grid-based partitioning system. This is mainly because of the minimal inter-partition change of the mobile objects achieved by the road network partitioning. As expected all approaches' performance worsens as the number of queries increase.

### 2.5.2 Object Side Computation Cost

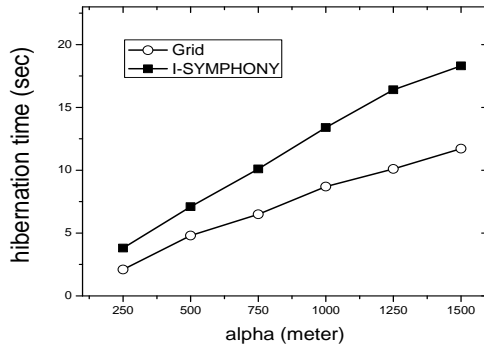
In this section, we study the amount of computation placed on the mobile object side by our distributed orchestration approach to the processing of IoT services requests. Figure 2.6b shows an average execution time at the object side for grid and road network-based partitioning with different partition size threshold ( $\alpha$ ). For the grid-based partitioning,



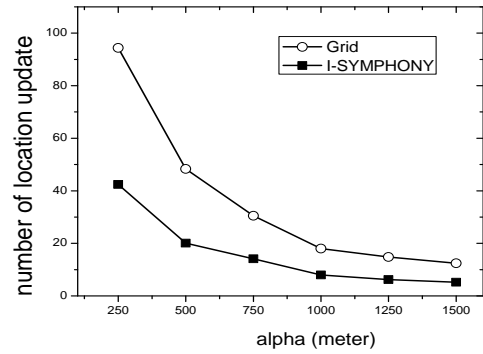
(a) Server side computation cost



(b) Mobile side computation cost



(c) Average hibernation time



(d) Average number of partition change

Figure 2.6: Experimental results of ISYMPHONY

the alpha is set as the diagonal distance of square grid cell. Note that we applied our optimization techniques for both approaches. Results show that our road network-based partitioning mechanism provides lower IoT object overhead because of two reasons. First, when a query issued from any object, the number of object install this query is higher in grid-based partition, since if the query range intersect with any point on the grid cell it install query for all object in that cell even though mobile object cannot reach intersect area because of the underlying road network constraint. Another reason is, for partition evaluation in grid-based approach object needs to consider any location on the square as a possible border point of the partition. However, in our system objects are aware of the border points of the partition and generated hibernation times higher than the grid-based

approach as shown in the Figure 2.6c.

### 2.5.3 Communication Cost

Finally, we measure the communication cost as the number of the message send to the midway server from IoT objects. We only measured the number of location update because of the partition change, since it is the only differentiation factor of the partitioning mechanisms with respect to communication cost. Figure 2.6d proves that considering underlying road network improves communication cost performance by minimizing the amount of inter-partition changes of mobile users.

## **2.6 Related Work**

Scaling large scale IoT systems and applications is a challenging problem. Several research efforts have been centered on innovative designs of underlying IoT infrastructure. LEONORE [20] proposes a service oriented infrastructure for providing elastic provisioning of IoT application components on resource constrained and heterogeneous edge devices for large scale IoT deployment. Geelytics [21] presents a system for on-demand edge analytics over IoT interfaced sensors and actuators with bandwidth optimization. [22] is the first to propose the sensing and actuating as a service (SAaaS) paradigm for geographically distributed IoT infrastructure. [23] presents an abstraction for IoT service provisioning in the Cloud. [24] proposes an approach for auto-configuration of IoT infrastructure based on heuristic based configuration suggestions. [25] presents a smart city based IoT framework and [26] develops an IoT experimentation based on a smart city testbed. We build the first prototype of ISYMPHONY on top of MobiEye [14], a simulation based mobile location monitoring system.

## CHAPTER 3

### FOGGY: A FRAMEWORK FOR CONTINUOUS AUTOMATED IOT APPLICATION DEPLOYMENT IN FOG COMPUTING

Traditional Cloud model is not designed to handle latency-sensitive Internet of Things applications. The new trend consists on moving data to be processed close to where it was generated. To this end, Fog Computing paradigm suggests using the compute and storage power of network elements. In such environments, intelligent and scalable orchestration of thousands of heterogeneous devices in complex environments is critical for IoT Service providers. In this chapter, we present a framework, called Foggy, that facilitates dynamic resource provisioning and automated application deployment in Fog Computing architectures. We analyze several applications and identify their requirements that need to be taken into consideration in our design of the Foggy framework. We implemented a proof of concept of a simple IoT application continuous deployment using Raspberry Pi boards.

#### 3.1 Introduction

The Internet of Things (IoT) connects physical objects to the virtual world and enables them to sense and manipulate their environment by using sensors and actuators [27]. Connecting physical world to the Internet creates life-enhancing opportunities in various domains like smart homes/cities, self-driving vehicles, and health monitoring. Because of the numerous opportunities that IoT provides, the number of connected devices is increasing rapidly, and International Data Corporation (IDC) predicted that number to reach 29 billion by 2020 [28]. This increasing number of things is leading to generate huge amounts of data that need to be collected, processed and stored. Cisco predicts that the total amount of data created by any device will quadruple in five years and in 2020 will reach 600 ZB per year [3]. Providing highly scalable solutions makes Cloud first-class citizen in the IoT environment

along with the end-devices, or things, with limited compute and storage capacity. However, traditional Cloud model is not designed to handle latency-sensitive IoT applications.

The new computing paradigm referred to as Fog Computing [13], extends cloud computing services (compute, storage, and network) to the edges of the network. Instead of sending all the data to the Cloud, Fog Computing uses network elements (e.g., gateways) close to the device in order to provide low-latency and scalability. This plays a big role in reducing the unnecessary bandwidth usage. For example, we can look at autonomous cars, where real-time decisions are critical, they generate 40GB data per hour. It is not realistic to try sending all the generated data to the cloud and expect to get decisions within milliseconds. It is noteworthy that, the Fog paradigm does not substitute the Cloud but extends it instead. The Cloud still remain as main component in IoT for applications that require high computations and long-term storage.

Although, Fog Computing is an extension of Cloud, deploying, managing, and updating IoT applications on such layered environment introduce new challenges. First of all, large scale IoT applications include a huge number of heterogeneous resources with distinct processing, memory, and storage capabilities. Second, workloads are dynamic on each of the nodes in the network. Finally, each IoT application has its own requirements such as latency-sensitivity, computation needs and privacy constraints. Therefore, the problem that we are trying to solve is: how can we offer the needed mechanisms to enable a continuous resources integration, planning, deployment and management in such a heterogeneous and dynamic environment.

In this work, we present a framework, called Foggy, that facilitates dynamic resources provisioning and automated application deployment in Fog Computing architectures. Our specific contributions are:

- Analyzing a set of IoT applications and defining their requirements.
- Providing a framework that manages resources and deploys IoT applications with minimal developer efforts.

- Proposing task scheduling mechanisms to optimize the resources usage and minimize the latency by considering application requirements and resource capabilities, costs, and mobility.
- Using container-based virtualization to provide system isolation and small overhead on resource constraint devices.
- Provide multi-node container update strategies based on the application requirements and the resource availability.
- Identify bottlenecks in the system and provide suggestions for possible resource placements.

## **3.2 Motivation and Background**

Our goal is to propose a framework that minimizes developer effort to deploy, update, and maintain large-scale geo-distributed IoT application codes while providing an efficient management of each resource in an IoT environment. In this section, we will present a common IoT infrastructure that supports large-scale IoT applications. Afterwards, we will identify the applications' requirements that need to be taken into consideration.

### 3.2.1 IoT Infrastructure

In this work, we consider commonly accepted three tier IoT infrastructures to support large-scale IoT applications ( see Figure 3.1). Such infrastructures incorporate various geo-distributed heterogeneous devices including sensors, actuators, compute resources, and networking devices. These devices are logically divided into three categories based on their roles:

- The first tier is composed of Edge devices such as smart phones, CCTV cameras, and smart thermostats. In this tier, devices interact with their environment through

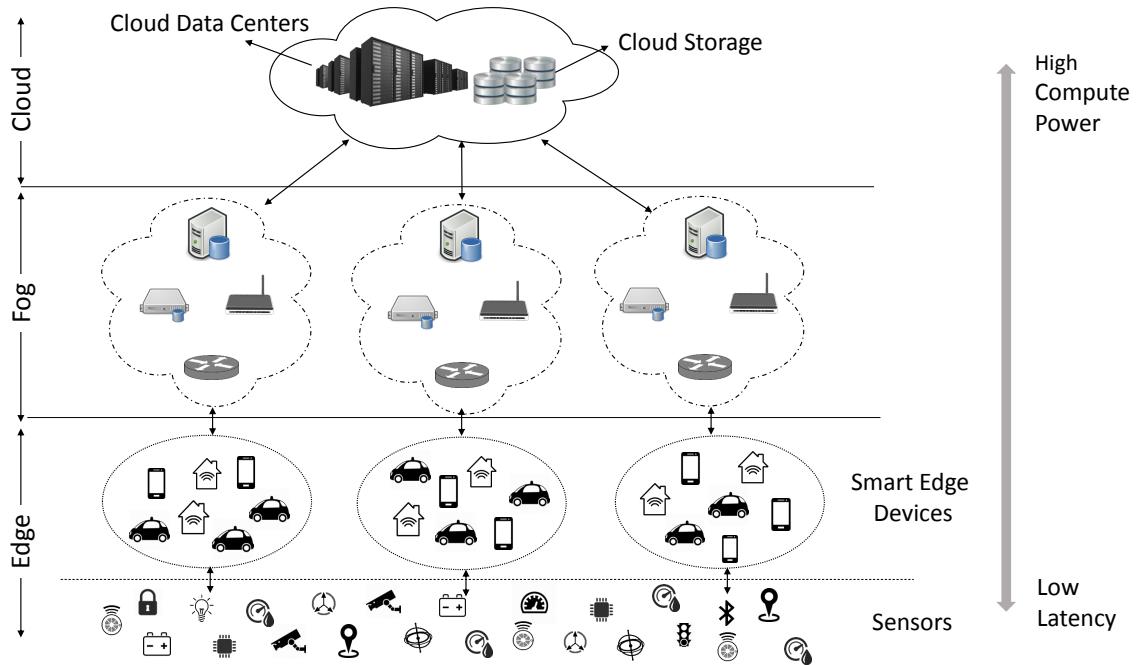


Figure 3.1: IoT Infrastructure

sensors and actuators. Edge devices can be static (e.g., pipe sensors) or mobile (e.g., connected cars) and their geo-spatial location information is provided during the installation process or by their integrated localization capabilities (e.g., GPS).

- The second tier is composed of Network infrastructure responsible of connecting the devices, gateways and Cloud. This tier contains the needed mechanisms to transport the data between the different devices and between the different layers (e.g., Edge, Cloud, ...).
- The third tier is based on Cloud services that provide main compute and storage resources to process, analyze and store the huge amount of data generated by edge devices.

IoT applications often require instant decisions. For these kinds of decisions, using the standard cloud model is not efficient for latency-sensitive geo-distributed IoT applications. Fog Computing has been proposed to decentralize data processing by extending cloud compute power towards the IoT network. To this end, in addition to dedicated re-

sources, current network elements (e.g., gateways) are not only a transferring medium, but also used as compute resources to process data as close as it can be to where it was generated. This approach not only minimizes the latency, but also decreases the bandwidth usage.

### 3.2.2 Requirements

In this section we will identify several requirements that need to be taken into consideration to effectively deploy IoT application on a defined infrastructure.

#### *Latency Sensitivity*

Many IoT applications depend on instant life-critical decisions and even second level latencies are not tolerable. For example, in smart transportation systems roads and intersections are equipped with various sensors to monitor cars moving on the adjacent road segments and to detect whether there is an accident or not. Connected cars communicate with IoT data sources (e.g., traffic lights, road sensors, other cars) to make a decision for the next step. The query expected to be answered immediately by the closest edge node based on real time data. In this scenario, any latency may lead to accidents.

#### *Bandwidth Usage*

Current large scale IoT sensors generate huge amounts of data each second. It is clear that sending all the generated data to the cloud for analytics and processing is not an option. For operational efficiency it is necessary to minimize data sent over the IoT network. For instance, in a city surveillance system millions of cameras could be installed all around the city. Security officers may ask the platform to track specific objects and receive fast notifications whenever some suspicious objects have been detected. It is impossible to transfer concurrent video streams from millions of cameras to the Cloud for detecting and tracking purposes.

### *Resource Constraints*

IoT infrastructures are composed of heterogeneous devices with various compute power, memory size, and storage capacities. An efficient resources management model is necessary to optimize the usage of such physical devices in order to provide scalable IoT applications. The model needs to make an optimal trade-off between processing the data locally or sending it to the Cloud based on the amount of data and the available resources and capabilities. It also needs to take into account the cost of the data transfer between the different nodes (devices, edge, cloud, ...).

### *Mobility Support*

In many IoT applications end devices are mobile such as smart phones and connected cars. As the main goal of the Fog Computing is to move compute power close to where the data is generated, it is necessary to be able to aggregate data at the closest network element while the end devices are moving.

### *Dynamic Workloads*

Unlike the centralized cloud model, in the Fog model, the mobility of the end devices and objects of interest in the environment introduces new challenges related to the dynamic compute, storage, and network requirements of the IoT applications. For example, camera surveillance systems in shopping malls that track shoppers introduce dynamic workloads on different fog nodes depending on the time and the location based on the shoppers' density.

### *Multi-tenancy*

Large IoT infrastructures are built to host multiple IoT applications with different requirements for different tenants. IoT service providers should ensure efficiency and fairness of all applications running on the system to serve the different tenants.

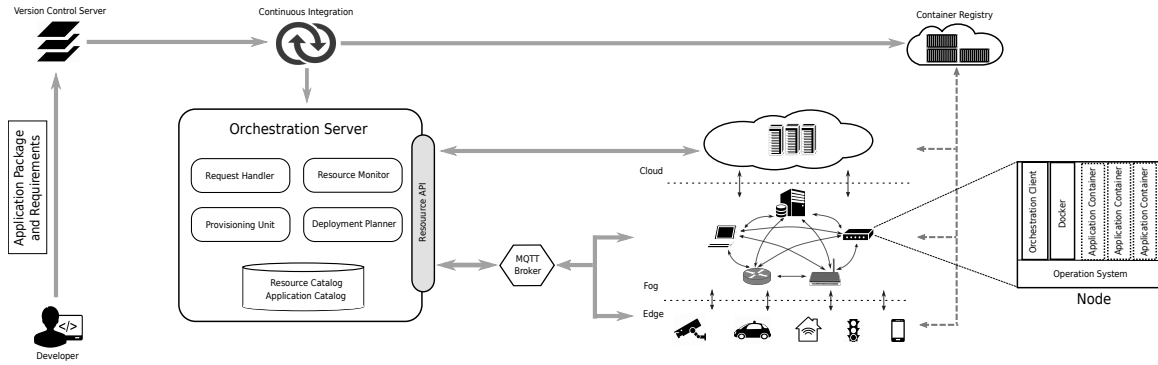


Figure 3.2: Foggy Framework

### Privacy

The types, amount, and specificity of data gathered by billions of devices create concerns among individuals about their privacy and among organizations about the confidentiality and integrity of their data. For example, smart homes enable patient self-treatment and monitoring by using simple devices, which provide standardized outputs for specific physiological conditions. The data can be used by intelligent applications capable of analyzing and processing body signals, sensor-integrated smart devices and wearable sensors. Such systems collect lots of sensitive information about residents. It is important to organize application deployment based on privacy-preserving aggregation at the predefined network level that prevents sending sensitive information to the unwanted resources.

### 3.3 Automated IoT Application Update and Deployment System

In this section we present the design of the framework to support large-scale geo-distributed IoT applications and the overall application deployment process. The framework is illustrated in Figure 3.2.

#### 3.3.1 IoT Applications

Large-scale IoT applications are based on processing data streams within several consecutive tasks. For example airport video surveillance system designed to identify passenger

faces and notify security officers whenever there is a match with any person wanted by the police. Such application requires several steps to run (see Figure 3.3) [29]. Basically, when the frame is captured, some preprocessing techniques (e.g., background subtraction) are applied to the image. Then, face detection techniques identify if the human faces appear in the current frame. Output of this step is a set of patches containing each face in the input image. In the feature extraction step, the faces are transformed into a vector. Finally, face recognition step compares features in the database with the new detected faces (based on the generated vector). If there is a match, a notification is sent to the security officers.

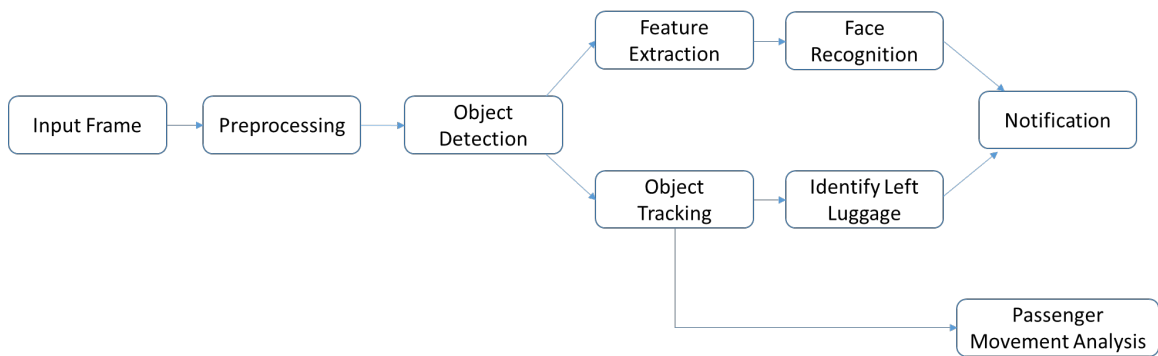


Figure 3.3: Video Surveillance System

In addition, IoT infrastructures are designed to support multiple applications. For instance, in airport video surveillance system another possible application is to detect abandoned luggages to prevent terrorist attacks. Unlike previous application, after the object detection step, it performs some object tracking algorithms. In the final step, it uses pattern recognition techniques to identify if anybody left his baggage and notify the security officers in real time [30].

Each task of IoT application has different compute and storage requirements. For example, passenger movement data can provide useful information such as location-based advertisement pricing. Such data analytics requires large amounts of trajectory data and computation power.

In order to maximize resource efficiency and support multi-tenancy, task-sharing, and easy software updates, we advocate to use container-based virtualization approach. Gen-

erating containers for each task and running it on the different nodes maximize quality of service and ensures the isolation. For example, running background subtraction and motion detection tasks on edge node eliminates any unnecessary video streaming to the high level nodes. In addition, IoT applications target a wide variety of device platforms. During prototyping and early phases of development, IoT projects are often developed using generic micro-controller-based development boards or single-board computers (SBCs) like the Raspberry Pi. The embedded applications that are developed for these devices can later be upgraded to run on custom prototype development boards and then finally on production devices. Each device or device revision might also require different versions and configurations of the development toolkits that are used for flashing, monitoring, and communicating with the device. Containers can be used to capture a development environment that is known to work for each device revision, and to share this environment among a team of developers <sup>1</sup>.

In order to identify user preferences and each container behavior we define specifications using JSON format. Afterwards, the orchestration unit uses these specifications to effectively and fairly place each container in the IoT infrastructure. Below we present an example specification file for the face detection container.

```
{ "FaceDetection": {  
  "Priority": "High",  
  "Privacy": "2",  
  "Computation": "Medium",  
  "Latency": "Low",  
  "Output": "Medium"  
}}
```

Listing 3.1: Container Specification Example

---

<sup>1</sup><https://www.ibm.com/developerworks/mobile/library/iot-docker-containers/index.html>

The *Priority* field defines the importance of the application. It is used to prioritize one application when it is not possible to deploy multiple containers on the same resource constrained device. The *Privacy* field is the limit of the level that the container can run in the IoT infrastructure hierarchy. The level is determined by the network distance from the edge device to the cloud. For example, for privacy reasons raw images captured by the security cameras may not be allowed to be sent beyond two levels of the hierarchy (edge devices and edge gateway). The *Computation* field is used to describe how much compute power is required for a given container. The *Latency* field defines the time-sensitivity of the task and finally, the *Output* field defines the size of the result, which is an estimate bandwidth usage. Note that the developer estimates such requirements for the initial deployment plan but they can change over time due to the dynamic nature of IoT applications (e.g., Computation). The Orchestration Server should monitor the resource usage and dynamically adapt container placement and notify the user about if there are any possible bottlenecks. We can put in place a machine learning algorithm that learns from the behavior of each application/container and estimate its optimal configuration under different circumstances.

### 3.3.2 System Components

In the following, we describe the different components that form our proposed architecture.

#### *Version Control Server*

This server might be any system that allows the continuous collaboration of developers in the development cycle of the IoT application. For instance, we can use any private or public Git server to host the base code of the IoT application.

#### *Continuous Integration Tool*

This is important for using automated build systems for CI (Continuous Integration). It allows early detection of possible build problems. CI makes sense when several contribu-

tors/team are working on the same code repository, and there is a need to make sure that the project is in a consistent state that allows any user to build it with all the team/contributors code inputs/changes. At the end, built deliverables can be pushed to their destination. The CI tool continuously monitor the code repository in the version control server, and whenever a change is detected, it kicks off the build and different tests of the new code (unit tests, integration tests, acceptance tests...). For our IOT projects we can have different automatic builds to be triggered after code commits on the code repository (for example, GitHub). When the build is triggered, it is able to pull all dependencies and frameworks to build the project on the automated build system.

### *Container Registry*

Container Registry (CR) is a stateless and scalable Docker image storage (e.g., Docker Registry <sup>2</sup>). It is a private cloud storage to keep IoT application secure and constantly accessible. Each IoT resource pull images from CR directly with secure HTTPS protocol. All versions of the same image stored in CR for two reasons: (i) If the new version is not working properly for any reason, it is important to roll-back immediately, (ii) Different versions might be suitable for different devices in the IoT infrastructure.

### *Node*

A node is an IoT device with compute and storage capabilities (e.g., gateways). We put in place an Orchestration client code (agent) that runs on each node on top of the operating system. It is responsible of pulling the container image from the registry, running/stopping the application, as well as sending the information about the device and the hosted containers running on it to the Orchestration Server.

---

<sup>2</sup><https://docs.docker.com/registry/>

### *Orchestration Server*

Orchestration server is responsible of the scalable IoT application deployment and resource management. In our design, the orchestration server is a part of cloud-based IoT service provider (e.g., IBM Watson IoT<sup>3</sup>). Two catalogs are maintained within the Orchestration Server; resource catalog and application catalog. Resource catalog stores information about the devices in the IoT infrastructure such as their capabilities, connections, and workloads. Application related information such as the versions and the requirements is stored in the Application catalog. In our architecture, the Resource monitor unit regularly checks each node to identify capacity usage and if there is any bottleneck in the running application. Whenever a new deployment request is received by the Request Handler Unit, the Deployment Planner tries to find the best strategy to run containers on the devices for optimal quality of service. Once the deployment strategy is determined, it is passed to the Provisioning Unit that applies defined container placement strategies by sending requests to the corresponding nodes.

The service provider communicates with the IoT resource nodes through the orchestration client running on each node. In our design, such communication uses MQTT protocol which is a simple, lightweight, publish/subscribe messaging protocol on top of the TCP/IP protocol. It is one of the most used protocols for the IoT world. We identified several interfaces used in this communication that client application running on IoT devices must implement;

- *get\_resource\_info (resource r)* : Returns the current available resources (e.g., CPU, storage) on the device.
- *container\_info()* : Returns the running instances information on the device.
- *install(url u, proxy p, context c)* : Pulls container from the registry and installs the container. The context provides any eventual needed parameters to run the container.

---

<sup>3</sup><https://www.ibm.com/internet-of-things/platform/watson-iot-platform/>

- *change\_proxy(container c, proxy p)* : Change output proxy of the container.
- *remove(container c)* : Removes container from the device.

### 3.3.3 IoT Application Deployment and Update Workflow

The goal of the IoT Application Deployment platform is to minimize developers effort to deploy and update applications in highly complex IoT infrastructures. We listed the overall deployment workflow below. Here, we assume that the resources and applications information are already maintained by the Orchestration Server.

- (i) The user pushes the containerized application package and its requirements file to the version control server.
- (ii) The version control server creates a new version of the code with a unique SHA code.
- (iii) The continuous integration tool uses the latest code (tagged with the latest SHA) generates the application container, uploads it to the container registry, and notifies the orchestration server.
- (iv) The Deployment Planner identifies the containers placement strategy.
- (v) The Provisioning Unit sends installation requests to each node with the container registry url and configuration parameters.
- (vi) The IoT nodes pull the container images and run the application components.

### 3.3.4 Planning and Provisioning Procedure

Determining an optimal container provisioning plan is a challenging process for thousands of heterogeneous devices placed in complex infrastructure. It is important to have fast provisioning mechanisms in order to meet instant update requirements (e.g., security updates). To this end, we propose policy-based resource management procedure. We followed a bottom-up approach to minimize latencies and bandwidth consumption.

In our approach, starting from the first container of the application, the Deployment Planner checks the data gathered by the Resource Monitor and saved in the Resource/Application catalogs to check if the first node that will get the input data is able to host the container based on its computation and storage requirements defined by the specification file introduced in Section 3.3.1. If this node is available and able to host the container, then, the container is assigned to the node and the provisioning step continues with the next container in the sequence.

If the node is not available or not powerful enough next possible node in the hierarchy is taken into consideration. The network nodes may connect with other nodes with similar configurations on the same level (e.g., gateway to gateway) as well as the powerful nodes in the higher level. Next node selection from the hierarchy can be one of the two types, horizontal (on the same layer) or vertical (on the upper or lower layer), depends on certain criteria. Horizontal selection consists on selecting a node from the same level, and it is used when the vertical selection introduces more latency for time-sensitive application. If the current level is the highest possible level that the container can run in, based on its privacy requirements, horizontal selection is used to find the possible node in the same level to run the container.

If the node does not meet container's requirements more suitable nodes can be found with vertical selection (going to the upper layer in this case). The process continues until every container of the application is placed. If there are multiple applications, a priority queue is created based on the application "priority" values defined in the specification file. The next node is selected after all applications' containers are tested to be added in the current node.

We are aware that the planning process might be specific for each use case scenario. In order to remain open to all possibilities, we designed the Deployment Planner to be pluggable in away that allows any user to bring his own deployment planning strategy and add it to the system. If the existing strategies do not respond to the user's needs he can

follow the interface of the Deployment Planner and just plug it to our system. We also provide some predefined strategies that we think are essential in code distribution among the nodes: 1) Deploy on a specific node: in this case, the planner will try to find the first node that responds to the description of the user, if this node is found and contains the needed resources, the provisioning starts in this node. This strategy can be used for fixing issues for a single node. For example, the user may specify to add an encryption mechanism to a specific camera on an airport.

2) Deploy on a specific number of nodes having some criteria: in this case, the planner will try to find the first fixed number of nodes respecting the specified criteria. For example, the user can specify that the code should be deployed on 3 nodes having at least 1 Gb of memory to ensure HA mode for a given application.

3) Deploy on all nodes having a specific property: in this case, the planner will find all the nodes having the specific property and deploy the application. For example, if a camera provider puts in place a new security patch, we need to push this patch to all the nodes containing the camera's software.

4) Deploy on all nodes: this mechanism is similar to broadcasting code. The planner will return all the nodes able to host the code. For example, if we need to upgrade an agent that exists in all our nodes, we need to use this strategy to have the update applied everywhere.

### 3.3.5 Container Update Model

The Provisioning Unit is also responsible of organizing the container instance updates on each resource node. To generate update steps, the Provisioning Unit takes into account two constraints: latency criticality of the application and storage and CPU capacity of the node. We propose several instance update strategies based on these constraints.

**Strategy 1: At most one instance running at a time** In this strategy, we consider applications that tolerate small downtime. For example, smart thermostat applications that requires temperature sensor data can tolerate certain downtime (e.g., several seconds). In this case, first the node removes the running container, than downloads the new instance from the registry, and finally starts the new instance. This strategy reduces the storage and CPU consumption of the node, and it is well suited to nodes that do not have enough storage or CPU to download new instance while the previous container is running. If the node has enough storage and CPU, in order to reduce downtime, the node pulls the new instance first and than removes running instance before starting the new one.

**Strategy 2: At least one instance running at a time** In this strategy, we consider applications that are latency-critical where the data is time-sensitive and even millisecond level latencies can cause operation inefficiencies (e.g., connected cars). For such applications update procedure should be seamless. To this end we propose installing new instance without removing previous version. After starting the new version, we can just reroute the calls to target the new instance of the deployed container. Once all the calls for the old instance are completed we can go ahead and delete it.

### 3.4 Implementation

In this section, we will give a description of the implementation that we realized during a proof of concept preparation for our work. We implemented the Foggy framework with conformance to what we presented previously. We used 4 Raspberry Pi 3 boards model V as our Edge devices connected to the system sensors. Each Raspberry Pi was flashed with Hypriot OS<sup>4</sup> to be able to host Docker<sup>5</sup> containers [31] in order to enable our containerized approach.

As a version control server we used the public git environment Github that hosts our

---

<sup>4</sup><https://blog.hypriot.com/>

<sup>5</sup><https://www.docker.com/>

applications code within different repositories. For the continuous integration, we set up our pipelines using an open source CI tool called Concourse <sup>6</sup>. Concourse is based on a declarative YAML configuration file that is easy to update when the project grows up. Concourse also is based on containers and it is really efficient in large scale environments. We also used the public Docker Registry to store our docker images for the different applications that we have. We setup concourse to continuously monitor the different Github repositories, whenever a change is detected (based on the hash of the commit of the repository), the build of the code is triggered to run the different tests that we have and build a docker image for each component and push it to the docker registry. Concourse then notifies the orchestration server that there is a new version of one or more components of the application.

The Orchestration Server is made up of different loosely coupled components. These components are implemented as micro services exposing rest interfaces that could be deployed in two different manners:

- the services could be deployed as applications on Bluemix®<sup>®</sup>, or
- they could be run as containers in any runtime that supports docker images

The Request Handler is playing the role of the mediator of queries. It receives the notifications from concourse and determines the list of calls to do in order to accomplish any new deployment. Its role is to transform the notification into information that the deployment planner can understand. This component is implemented as a JAVA application using Jersey framework to implement the REST architecture. It is a stateless application that we can scale up or down horizontally when needed.

The Deployment Planner is also a JAVA application that contains the planning capacities of our orchestration server. The advantage of this component is its pluggability. In our design phase we were aware that choosing an optimal distribution of the IoT components

---

<sup>6</sup><https://concourse.ci/>

may change from one use case to another, we decided to make this component pluggable, so any user can define his own by implementing the interface that we specified. If needed, the deployment strategy can be easily changed to a more efficient one whenever we change the context of the application. We are thinking of having different implementations of the planning strategy, and dynamically balancing from one strategy to another according to the events and contexts that we might face.

We also implemented the Resource Monitor component that collects monitoring data related to the IoT devices and the running applications/containers. This component is a JAVA application that makes use of the cAdvisor API <sup>7</sup> to collect data about the different nodes that we have. cAdvisor provides data about the resource usage and performance characteristics of the different running containers in any given node.

It is noteworthy that all of the orchestration meta-data that we have is stored in a Cloudant database offered by Bluemix® that is shared between the different components. This meta-data is used to make the deployment decisions by the orchestration server. This meta-data is also continuously evolving and being fed by the Resource Monitor that updates the different resources usage and status.

### **3.5 Related Work**

In recent years, with the growing popularity of its applications, IoT has gained significant research attention [32]. One of the important directions is the design of IoT system infrastructures to provide a better user experience. Thousands of heterogeneous, geographically distributed, and possibly moving devices introduce new challenges in real-time IoT applications [33]. In order to provide low-latency, minimal bandwidth consumption, and high scalability, a new trend in IoT systems consists on processing data close to where it was generated. [34] propose GigaSight that extends the mobile device-cloud architecture with virtual machine-based cloudlets in order to minimize latency in mobile IoT applications.

---

<sup>7</sup><https://github.com/google/cadvisor>

Similarly, Cisco proposed Fog Computing model [13] in which network elements (e.g., gateways) extend the cloud with data processing and storage capacity close to the edge.

Such complex IoT computing infrastructure creates need for effective deployment, provisioning, and resource management strategies. LEONORE [20] provides elastic provisioning of IoT application components on resource constrained and heterogeneous edge devices for large scale IoT deployment with a service oriented infrastructure. Geelytics [35] presents an edge analytic platform that provides low latency stream processing with optimized edge-to-cloud bandwidth consumption. Mobile Fog [36] proposes high-level programming model for geo-distributed, latency-sensitive IoT applications. [37] presents a service oriented resource management model for Fog infrastructure that uses customer type-based resource estimation in order to facilitate IoT resource provisioning. All these approaches are solely focusing on providing low latency application deployment. However, in real life IoT applications may have other requirements such as privacy protection and bandwidth usage limitation. Unlike those approaches, in Foggy, we incorporates different application requirements in our deployment strategy. [34] and [38] takes into account privacy-preserving while providing resource provisioning in IoT environment. Satyanarayanan et al. [34] proposed using virtual machine (VM)-based cloudlets in order to provide privacy-aware video analytics in the three-tier architecture. However, VM-based virtualization is not suitable for resource constraint device such as gateways. In Foggy, we used container-based virtualization to provide flexibility and easy update.

## CHAPTER 4

### PRIVACY-PRESERVING LOCATION QUERIES FOR MOBILE TRAVELERS: A UTILITY-AWARE AND ATTACK-RESILIENT APPROACH

Location queries provide fundamental services for many mobile travelers. Unfortunately, continuous exposure of location information may lead to breaches of location privacy caused by statistical inference attacks. Location privacy can be protected by spatial cloaking solutions that anonymize sensitive location information to a  $k$ -anonymized spatial region of low resolution so that at least  $k$  users will issue location queries with the same perturbed location (location  $k$ -anonymity). However, existing spatial cloaking solutions have two known problems: (1) the loss of location utility such as increased query cost associated with using spatial cloaked location, and (2) vulnerability to background knowledge-based spoofing and replay attacks, preventing wide deployment of privacy-preserving location query systems. To address these problems, this chapter presents STARCLOAK, a utility-aware and attack-resilient approach to privacy-preserving location queries for mobile travelers on road networks. STARCLOAK is novel in three aspects: (1) It incorporates spatial and temporal location utility constraints as an integral part of its location anonymization model, ensuring the query utility of anonymized locations; (2) its privacy-preserving location query algorithms minimize additional end-to-end latency introduced by anonymous query processing; (3) it embeds in its location anonymization process robust defense capabilities that protect spatially cloaked locations from sophisticated attacks based on background knowledge such as correlation-based replay attacks and query injection-based attacks. Extensive experiments show that STARCLOAK can effectively integrate spatial and temporal location utility constraint, anonymous query processing cost, and complex location inference attacks with high query cost-efficiency and strong attack resilience for scaling anonymous location queries.

## 4.1 Introduction

The growth of location-based services (LBSs) is fueled by the ubiquitous wireless connectivity, the universal presence of smart mobile devices with multi-modal sensing capability, and increased investments from industry and government on the Internet of Things (IoT). Juniper Research [4] forecasts that the LBS market will reach \$43.3 billion in revenue by 2019, rising from an estimated \$12.2 billion in 2014. [5] reports that %74 of adult smartphone owners use their phones to get direction or information based on their current locations. As more and more mobile travelers and vehicles are connected continuously and automatically, we are embraced by many life-enriching location-based experiences and services (e.g., improved emergency assistance, real-time traffic alerts). However, for many location-based services and mobile data management applications, location queries are the fundamental functionality and continuous location exposure opens doors to the intrusion of location privacy [6]. For example, even if users identifiers are removed from their location queries, successive position updates can be linked to identify individuals with high confidence. Unauthorized trajectory exposure may also expose mobile users of LBSs to significant vulnerabilities for abuse such as unwanted advertisement, stalking, and location spoofing. In addition, when private location data of a mobile user is linked to sensitive public locations such as mental health clinics, cancer treatment centers, nightclubs, and religious organizations, such unauthorized location - identity linkage may cause ethical, professional, and social risks to both individuals and our society at large.

Over the last decade, research on location privacy has gained increased attentions. We classify location privacy research and development into three broad categories. The first is policy-based solutions that restrict unauthorized access to location data through privacy policies that typically provide an option for users to turn off location-based services or to block tracking. The second category consist of mix-zone based approaches [39, 40], which protect location privacy by changing pseudonyms at selected locations in so that an attacker

cannot infer mapping between old and new pseudonyms of mobile users, thus protecting the location-identity linking attacks. Mix-zones are effective for LBSs that require identity or pseudonyms of mobile users. The third category is based on location obfuscation models and algorithms. Unlike mix-zone approaches, location-obfuscation approaches are targeted at applications that do not require true identities or pseudo-identities of mobile users. Examples of such applications are finding nearby gas stations or restaurants or notifying a user about the traffic conditions when the user is approaching Piedmont Hospital in Atlanta.

The most representative location obfuscation approaches are location  $k$ -anonymization and differential location privacy. Location  $k$ -anonymization techniques hide the exact location of a mobile user by replacing it with a  $k$ -anonymized cloaking region in user's location query, which guarantees that at least  $k$  mobile users share the same cloaked region as their published location. A subject is considered *location  $k$  - anonymous* if its location is indistinguishable from  $k - 1$  other users' location [6, 7, 8, 9, 10]. Research in differential location privacy [41, 42, 43] extends differential privacy theory [44] by using geo-indistinguishability for location perturbation, which transforms the actual location point of a mobile user to a random pseudo-location that is geographically close to the actual location point and guarantees that the original and the perturbed-location points have similar probabilities to generate the same pseudo-location.

The research presented in this chapter falls into the location-obfuscation category. We argue that both location  $k$ -anonymization algorithms and differential location privacy algorithms suffer from a number of known problems: (1) Most existing algorithms compute the anonymized location solely based on specific location-privacy metric or metrics. Very few of these algorithms has incorporated location utility and location query cost into consideration in their spatial cloaking decisions. (2) Most existing location privacy algorithms are vulnerable to complex replay attacks such as correlation- and query injection-based attacks. (3) Most existing solutions develop spatial obfuscation techniques based on the

*random waypoint* mobility model [11, 12], in which mobile travelers move in arbitrary directions at random speed. These solutions fail to address privacy risks when mobile users traveling in a spatially constrained environment such as a road network in which both user mobility and location query processing are constrained by the underlying road network. For example, as a spatial cloaking region may contain a single road segment when we use rectangle-based spatial cloaking techniques, they are vulnerable because adversaries can easily track the whereabouts of mobile users.

In this work, we present STARCLOAK, a query utility-aware and attack-resilient approach to building a privacy-preserving location query system for mobile users traveling on the road networks. The design of the STARCLOAK is novel in three aspects. One is that it incorporates spatial and temporal location utility constraints as an integral part of its location anonymization model, ensuring the query utility of anonymized locations. In addition, its privacy-preserving location query algorithms minimize additional end-to-end latency introduced by anonymous query processing. We argue that a spatially-cloaked location should be computed by maximizing location privacy and minimizing location query cost introduced by anonymization. Last but not least, we formally study some sophisticated background knowledge-based attack models such as correlation-based replay attacks and query injection-based attacks and provide analytical models to quantitatively evaluate the attack resilience of our location anonymization algorithms with respect to spatially-cloaked locations. We evaluate STARCLOAK through extensive experiments and simulation on road networks with varying scales of mobile users. To the best of our knowledge, STARCLOAK is the first privacy-preserving location query system that generates anonymous location queries by integrating location utility constraints, anonymous query processing costs, and location inference attack resilience.

## **4.2 STARCLOAK Overview**

STARCLOAK can be viewed as a trusted third-party *location anonymization service*. It

serves location privacy protection that forms as a middle layer between mobile users and their untrusted LBS providers and performs location anonymization for mobile users. Mobile travelers who wish to protect their location queries from location spoofing attacks may register to the STARCLOAK. Assume that mobile user Alice issues a location query while she is moving along a certain road segment. Without STARCLOAK, the location query service on Alice's mobile client will directly send her location query with her current position to an untrusted LBS provider that executes a query based on her location information and returns the results to her mobile client. However, STARCLOAK will first compute the anonymized location and then replace Alice's exact location before it sends her query to the untrusted third-party LBS provider. Then it will filter anonymous location query results and return actual location results to Alice's mobile client. All of these steps are executed transparently from mobile users like Alice. This section presents an overview of STARCLOAK, describes utility-preserving spatial cloaking algorithms, and provides a formal analysis of its attack resilience in subsequent sections. We assume that mobile users travel on spatially constrained road networks or walk paths. Thus, we first introduce the basic concepts and models for the road networks, location privacy and location queries, followed by a statement of problem and the novelty of our solution approach.

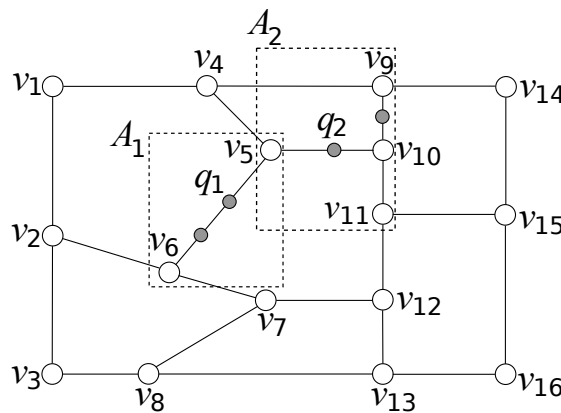


Figure 4.1: A road network model.

### 4.2.1 Road Network Model

This work presents a model of a road network as undirected graph  $G = (V_G, E_G)$  with the node set  $V_G$  and the edge set  $E_G$ , representing road junctions (nodes) and road segments (edges), respectively. Each road segment connects a pair of junctions. Figure 4.1 illustrates a road network. We use  $d_G(v)$  to denote the degree of a node  $v$  with respect to the graph  $G$ ,  $d_G(v) = |\{w | (v, w) \in E_G\}|$ . We call  $v$  an *intersection node* if  $d_G(v) \geq 3$ , an *connection node* if  $d_G(v) = 2$ , and an *end node* if  $d_G(v) = 1$ . In Figure 4.1,  $v_5$  is an intersection node and  $v_1$  a connection node. An anonymized location in the road network environment is a subgraph that can be define as follow;

**Definition 1 (Subgraph)**  $\mathcal{S}$  is a subgraph of the road network  $G$ , denoted by  $\mathcal{S} = \langle V_S, E_S \rangle$ , if and only if  $V_S \subset V_G$  and  $E_S \subset E_G$ .

This definition allows a subgraph  $\mathcal{S}$  with some isolated nodes that have no connection (edges) to any nodes outside the subgraph  $\mathcal{S}$ . This introduce a new concept, the border node, defined as follows;

**Definition 2 (Border Node)** Let  $\mathcal{S}$  denote a sub-graph of road network  $G$ . The set of border nodes of  $\mathcal{S}$ , denoted by  $\mathcal{BV}(\mathcal{S})$ , are nodes in both  $\mathcal{S}$  and  $G$  but have edges that are not in  $E_G$  but in  $E_S$ , or no edges between any nodes that are in  $V_G$ , not in  $\mathcal{S}$ . The equation that represents border nodes follows:

$$\mathcal{BV}(\mathcal{S}) = \{v | \forall v \in V_S, d_G(v) \geq d_S(v), \exists w \in (V_G - V_S), (v, w) \notin E_S\}$$

For example, Figure 4.5b illustrates subgraph  $\mathcal{S}$  with  $V_S = \{v_2, v_4, v_5, v_6, v_7, v_{10}\}$  and  $E_S = \{\overline{v_4v_5}, \overline{v_5v_{10}}, \overline{v_5v_6}, \overline{v_6v_7}, \overline{v_6v_2}\}$ . The border node set of  $\mathcal{S}$  is given by  $\mathcal{BV}(\mathcal{S}) = \{v_2, v_4, v_7, v_{10}\}$ .

### 4.2.2 Utility-aware Location Privacy Model

STARCLOAK assures the location privacy of mobile travelers in terms of both privacy and utility metrics. We promote personalized (customizable) location  $k$ -user anonymity and location  $l$ -segment indistinguishability to measure the resilience of a location anonymization algorithm against location spoofing attacks. In addition, we introduce two location utility metrics: minimal spatial and temporal cloaking resolutions. These utility metrics serve dual purposes: (1) They capture the spatial and temporal location utility desired by mobile travelers who need location-based services from respective LBS providers, and (2) they constrain and regulate STARCLOAK so that it perform location anonymization while meeting the spatial and temporal resolution (utility) constraints.

**Privacy-aware Preprocessing.** Given the assumption that mobile users are moving along the pre-defined spatial network (roads), we observed two cases: First, end nodes are more sensitive since they usually represents semantic locations such as residential houses and hospitals; and second, *connection nodes* (i.e., nodes with degree of two) do not contribute to location privacy protection. Because, an attacker can infer mobile users' future and past locations among the sequence of segments linked by connection nodes. After all, it is unlikely that mobile users make U-turns in the middle of the road. Therefore, we remove all end nodes and merge all connection nodes during the preprocessing step. After this step, a sequence of edges  $(\overline{v_0v_1}, \dots, \overline{v_iv_{i+1}}, \dots, \overline{v_{L-1}v_L})$  in which  $(v_i \neq v_j)$  are merged into one road segment, denoted by  $(\overline{v_0v_L})$ , in which  $d_G(v_0) \geq 3$ ,  $d_G(v_L) \geq 3$ , and  $d_G(v_i) = 2$  for  $0 < i < L$ . Consequently, a road network  $G$  is uniquely partitioned into a set of road segments each represented by its two end points (intersection nodes), and when no confusion occurs  $\overline{v_0 \dots v_L}$  is also presented as  $(v_0, v_L)$ .

Location anonymization by location perturbation (obfuscation) refers to the location transformation process, which translates an exact location into another geometric location of lower resolution that preserves user-defined location privacy requirements. Examples of such perturbed locations are a rectangle geometric region [7, 8] or a subgraph of a road net-

work [bamba:2015, 45]. One criterion for utility preserving location anonymization is the location cloaking capability, which produces privacy preserving location anonymization with minimal loss of location utility and minimal overhead of anonymous location-query processing. The first location privacy metric is **location  $k$ -user anonymity**, which ensures the location indistinguishability of a specific mobile user among a set of  $k$  users (i.e., anonymous set of size  $k$ ). Instead of using a system-supplied  $k$  for all users and all queries [46], we advocate personalized (customizable) *location  $k$ -user anonymity* [7], enabling various settings of  $k$  for users and their queries.

**Definition 3 (Location  $k$ -user anonymity)** *Let  $\mathcal{S}$  denote the published location for a mobile user  $u$  with respect to a query  $q$ . Let  $L_q$  denote the focal location of the location query  $q$ . If at least  $(k - 1)$  other active users have the same published location  $\mathcal{S}$  as user  $u$ , the published location  $\mathcal{S}$  is  $k$ -anonymous with respect to  $u$ ,  $q$  and  $k$ . We use  $\delta_k$  to denote the achieved location  $k$ -anonymity. Formally, given  $k$ ,  $L_q$ , and  $u$ , the output of location anonymization,  $\mathcal{S}$ , with a  $k$ -anonymity guarantee satisfies the following conditions:*

- $\mathcal{S}$  is a subgraph of the road network  $G$ .
- $\delta_k$  users have  $L_{q_1}, L_{q_2}, \dots, L_{q_k}$  as their location points on some segments in  $\mathcal{S}$ . Let  $\text{AnonymousSet}(\mathcal{S})$  denote the  $\delta_k$  users, each described by  $(u_i, L_{q_i}, k_{u,q})$  ( $1 \leq i \leq \delta_k$ ), in which  $u_i$  denotes the user's identity,  $L_{q_i}$  denotes the query focal location, and  $k_{u,q}$  denotes the personalized  $k$  defined by user  $u$  with respect to query  $q$ . The maximum  $k$  in  $\text{AnonymousSet}(\mathcal{S})$  is not more than  $\delta_k$ , that is,  $\forall (u, L_q, k_{u,q}) \in \text{AnonymousSet}(\mathcal{S}), \delta_k \geq k_{u,q}$ .
- All  $\delta_k$  users publish the same  $\mathcal{S}$  as their anonymized location. Each user query  $q$  is transformed into anonymous query  $q_A$  by using  $\mathcal{S}$  to replace their exact query focal location  $L_{q_i}$  ( $1 \leq i \leq \delta_k$ ).

Location  $k$ -user anonymity hides the actual location of user  $u$  in an anonymous location shared by at least other  $k - 1$  users in the vicinity of  $u$ , which makes it harder for an ad-

versary to distinguish  $u$  from the other  $k - 1$  users based on published anonymous location  $S$ . Location-cloaking algorithms use various computation models to find  $k$ -anonymized location  $S$  based on the actual location of user  $u$  with respect to query  $q$ . In Section 4.2.6, we will compare the STARCLOAK approach to existing cloaking methods and analyze its superiority with regard to both location privacy and utility.

The second location privacy metric is **location  $l$ -segment indistinguishability**. It is known that location  $k$ -anonymity fails to prevent the linking of user  $u$  to a sensitive public location on the road network with high probability. We assume that users of queries  $q_1$  and  $q_2$  published their 2-anonymized location as  $A_1$  and  $A_2$ , respectively, illustrated in Figure 4.1. Even though both spatially cloaked locations meet the location  $k$ -anonymity, an adversary more easily tracks the whereabouts of user  $u_1$  than those of the user  $u_2$ , since  $q_1$  is associated with a single road segment  $\overline{v_5v_6}$  in  $A_1$  with 100% confidence. However,  $q_2$ , with published location  $A_2$ , is associated with any of the three segments in  $A_2$  with equal probability. This example demonstrates that in order to protect the sensitive location information of mobile users traveling on the road network from unauthorized privacy leakages, location privacy should be defined and measured by both location  $k$ -user anonymity and location  $l$ -segment indistinguishability ( $k > 1, l > 1$ ).

**Definition 4 (Location  $l$ -segment indistinguishability)** *Anonymized location  $S$  for a user  $u$  with respect to query  $q$  is  $l$ -segment indistinguishable, if  $S$  contains at least  $l_{u,q}$  road segments such that the association of any of the  $l_{u,q}$  segments to with user  $u$  and query  $q$  is indistinguishable from the other  $l_{u,q} - 1$  segments. We use  $\delta_l$  to denote the achieved location  $l$ -indistinguishability. Formally, given  $l, L_q$ , and  $u$ , the output of location anonymization  $S$  with a location  $l$ -indistinguishability guarantee satisfies the following conditions:*

- $S$  is a subgraph of the road network  $G$ .
- Let  $l_{u,q}$  denote the personalized  $l$ , defined by user  $u$  with respect to query  $q$ , and each user in  $\text{AnonymousSet}(S)$  is represented by  $(u, q, l_{u,q})$ . Anonymized location  $S$  must

contain  $\delta_l$  distinct road segments, where  $\delta_l$  is the maximum  $l$  in  $\text{AnonymousSet}(\mathcal{S})$  such that  $\forall (u, L_q, l_{u,q}) \in \text{AnonymousSet}(\mathcal{S}), \delta_l \geq l_{u,q}$ .

We refer to  $(\delta_k, \delta_l)$  as the unified location privacy measures that should be achieved by location anonymization algorithms to meet the user-defined  $(k, l)$  location privacy requirements ( $k$ -user anonymity and  $l$ -segment indistinguishability).

In addition to the two location privacy metrics, we formally introduce two location utility metrics defined by maximum spatial utility,  $\sigma_s$ , and maximum temporal utility,  $\sigma_t$ . The spatial utility ( $\sigma_s$ ) bounds the spatial resolution of the anonymized location, allowing the location-cloaking algorithm to limit the spatial cloaking resolution reduction under the upper bound. The temporal utility ( $\sigma_t$ ) bounds the maximum time delay resulting from the location anonymization operation, prompting the location cloaking algorithm to utilize this temporal upper bound to increase the success rate of location anonymization under user-defined spatial utility constraint  $\sigma_s$ . If a location query  $q$  cannot be anonymized under spatial utility constraint  $\sigma_s$  and temporal utility constraint  $\sigma_t$ , then  $q$  is discarded (failed) because of the spatial or temporal utility constraint violation.

**Definition 5 (Utility-aware Anonymization)** *Let  $q$  denote a location query with focal location  $L_q$  issued by mobile user  $u$ . Let  $(k, l, st, tt)$  denote the user-defined privacy and utility constraints with respect to query  $q$ . The utility-aware location anonymization transforms the exact location,  $L_q$ , associated with query  $q$  to a spatially cloaked location  $\mathcal{S}$  with  $(\delta_k, \delta_l, \sigma_s, \sigma_t)$ , and guarantees that  $\mathcal{S}$  satisfies location  $k$ -user anonymity ( $\delta_k \geq k$ ), location  $l$ -segment indistinguishability ( $\delta_l \geq l$ ) while preserving both the spatial resolution constraint ( $\sigma_s \leq st$ ) and the temporal resolution constraint ( $\sigma_t \leq tt$ ). Formally, given  $L_q$  and  $u$  with  $(k, l, st, tt)$ ,  $\mathcal{S}$  with  $(\delta_k, \delta_l, \sigma_s, \sigma_t)$  are the output of utility aware location anonymization with  $k$ -anonymity and  $l$ -indistinguishability guarantees, satisfying the following conditions:*

- $\mathcal{S}$  is a subgraph of the road network  $G$ , and  $\mathcal{S}$  meets location privacy defined by location  $k$ -user anonymity and location  $l$ -segment indistinguishability.

- Let  $st_{u,q}$  and  $tt_{u,q}$  denote the spatial and temporal resolution constraints defined by user  $u$  with respect to query  $q$ . Let  $AnonymousSet(S)$  denote the  $\delta_k$  users, each  $(u, q)$  is associated with the user-defined privacy constraints ( $k_{u,q}$  and  $l_{u,q}$ ) and the utility constraints ( $st_{u,q}$  and  $tt_{u,q}$ ), such that  $\forall (u, q) \in AnonymousSet(S)$ ,  $\sigma_s \leq s_{u,q}$  and  $\sigma_t \leq t_{u,q}$ .

The spatial utility metric, using the cloaked location, defines the maximum spatial resolution reduction, which is tolerated for executing  $q$ . Similarly, the temporal utility metric states the maximum temporal delay, allowed without exceeding the user-defined deadline threshold for location query  $q$ . We call  $(\sigma_s, \sigma_t)$  the achieved utility guarantee for anonymized location  $\mathcal{S}$ , computed by the location-anonymization algorithm. By introducing spatial cloaking tolerance and temporal cloaking tolerance as the two utility constraints, we allow mobile travelers to define their preferred spatial and temporal resolution of an anonymized location based on the type of location queries and their preferred location utility. Location anonymization algorithms can differ in terms of both privacy and utility metrics. Even algorithms that preserve the same privacy metrics and utility metrics differ in terms of three factors: (i) risk resilience of their location privacy guarantee, (ii) spatial-temporal resolution losses of their location utility, and (iii) cost of anonymous query processing that result from different sizes and graph compactness of anonymized location  $\mathcal{S}$ .

In the next two sections, we describe two baseline location-anonymization algorithms, introduce the cost model for anonymous query processing, and show that both baseline algorithms meet the two location privacy metrics and the two location utility metrics. However, they differ significantly in terms of privacy risk resilience, utility losses, and anonymous query processing costs, the main motivation for the design and development of the STARCLOAK location-anonymization algorithm, which optimizes all three factors during the location-anonymization process.

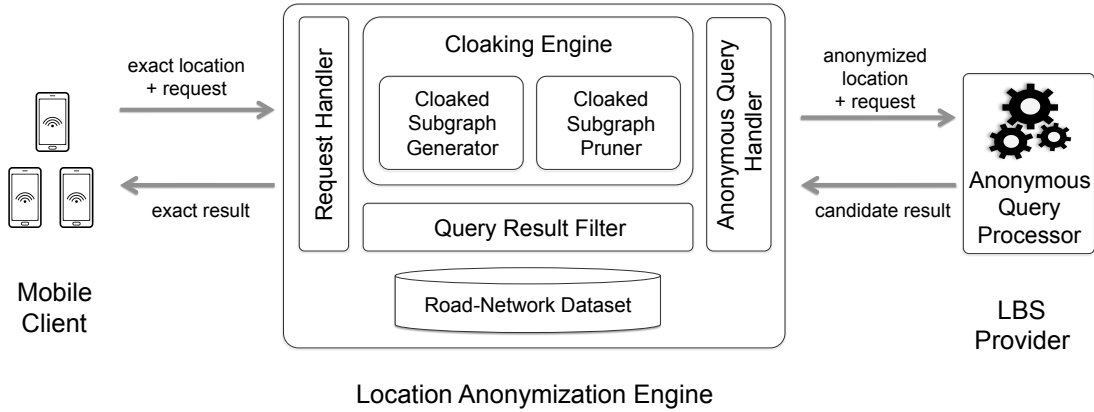


Figure 4.2: Overall architecture of STARCLOAK.

### 4.2.3 Anonymous Location-based Service Architecture

Figure 4.2 illustrates the reference anonymous query processing architecture. Let  $q$  denote the original query by user  $u$  with location  $L_q$  and  $q_A$  the anonymous query of  $q$  by replacing exact location  $L_q$  with cloaked location  $S$  under user-defined anonymization constraints  $(k, l, st, tt)$ . With location privacy protection by anonymization, LBS requests from mobile travelers are transformed into anonymous queries before they are relayed to the respective LBS provider. The system architecture for anonymous location query processing consists of a sequence of four data flows: First, a mobile client sends location query request  $q$  with focal location  $L_q$  to the intended LBS provider, and given that user  $u$  has subscribed to a location anonymization service (e.g., STARCLOAK), the anonymization agent will intercept location query  $q$  of a mobile client, and after performing location sanitization, the agent will relay the anonymized query. Such a relay process can be done by either an anonymization server or an anonymization agent on mobile clients, depending on the choice of a concrete anonymization service.

In the second flow, the location anonymization engine, upon receiving a location anonymization request, computes the anonymized location  $S$  for query  $q$  and generates anonymized query  $q_A$  with  $S$  as the anonymized focal location of user  $u$  with respect to query  $q$  and

relays anonymous query  $q_A$  with  $S$  to the LBS provider. Following this flow, the LBS provider computes anonymous location query  $q_A$  in the same way as it did the regular location queries and returns the results to the relay agent of the anonymization service such as STARCLOAK. Given that the anonymized location has a lower resolution than the original query focal location, the result set of  $q_A$  may be much larger and contain a large number of false positives. Finally, the relay agent will perform post-processing of the results of  $q_A$  to filter all false positives and deliver the exact query answer with respect to original focal location  $L_q$  to mobile user  $u$ .

In STARCLOAK, we use a trusted third-party anonymization server model to provide anonymized location  $S$  with an  $l$ -segment indistinguishability guarantee. As such a guarantee can be satisfied by only accurate road network data, which is usually too large for mobile clients to store. A request handler unit in the STARCLOAK location anonymization engine is responsible for communicating with mobile users and preparing queries for the anonymization process. We explained the preprocessing step in Section 4.3.3. The cloaking engine generates anonymized locations in two units: a candidate subgraph generator and a cloaked subgraph pruner (Sections 4.3.4 to 4.3.8). The anonymous query handler unit communicates with an LBS provider. Finally, the query result filter unit removes false positive results from the result set returned by the LBS provider.

Providing privacy of mobile users introduces time delay and additional network usage, thus it affects their overall experience using location-based services. The anonymization process and anonymous query evaluation are the most dominant factors of the time delay in privacy protection architectures. False positive results, the consequence of using an anonymized location, cause additional network usage. An important goal of a location anonymization system is to minimize these effects. An optimal location anonymization system generates an anonymized location  $S$  with the following properties: (1) It satisfies user-defined privacy  $(k, l)$  and utility constraints  $(st, tt)$ ; (2) It is found whenever  $k$  users can be cloaked; (3) for any other cloaked location  $S'$  ( $S' \neq S$ ) that satisfies  $(k, l, st, tt)$ ,

anonymous query  $q_A$  with cloaked location  $\mathcal{S}$  has the lowest evaluation cost; (4) it causes a minimum number of false positives; (5) it has strong resilience against inference attacks that adversaries may instigate. In Section 4.5.2 we introduce metrics for evaluating the optimality of anonymization systems. In the next two sections, we will formalize anonymous processing costs and provide a set of inference attacks.

#### 4.2.4 Anonymous Query Cost Model

One of the main challenges for finding optimal location anonymization output  $S$  such that privacy preserving location query  $q_A$  with  $S$  as its anonymized location will have the lowest cost. This operation requires that the location anonymization algorithms account for query-processing and communication costs when choosing a spatially cloaked location to anonymize the actual focal location of a query. Thus, the STARCLOAK anonymization algorithms are constrained by anonymous query costs.

##### *Cost of Anonymous Query Evaluation*

Most state-of-the-art query processing approaches for road networks [47, 48, 49, 50, 51] are based on two types of fundamental operations: edge and node. However, these approaches differ with regard to their assumptions. We formalize the cost models for anonymous query processing based on the cost of the edge and node operations.

The **edge-based operation** takes a query  $q$  and an edge  $e$  as input and returns a set of objects on  $e$ , which satisfies the query condition denoted by  $\mathcal{O}_e(q, e)$ . As the most fundamental operation, most query processing techniques construct indexing structures to speed-up edge-based operations. For a segment  $s$  composed by a sequence of edges, if we denote  $\mathcal{O}_s(q, s)$  as the set of objects matching  $q$  on  $s$ , we obtain  $\mathcal{O}_s(q, s) = \cup_{e \in s} \mathcal{O}_e(q, e)$ . We use  $\mathcal{C}_s$  to denote the average computation cost, in terms of both CPU and IO, of evaluating segment-based query.  $\mathcal{C}_s$  depends on the number of edges, length of the edges, the number of objects on the edges, the predicate condition of the query, and the underlying

system implementation, such as edge indexing and caching mechanisms. In our current system, we set  $\mathcal{C}_s$  statically according to the underlying implementation (e.g., fast look-up table, R-Tree, etc), assuming the condition of average edge length, average object density and no specific predicate, though finer model can be developed, which we consider as our future work.

The **node-based operation** takes a query  $q$  and a node  $v$  as input and returns the set of objects in the vicinity of  $v$ , which satisfies the query condition denoted by  $\mathcal{O}_v(q, v)$ , in which  $v \in V_G$ . The computation cost of evaluating a node-based query denoted by  $\mathcal{C}_v$ . Similar to edge-based operations, besides the properties of the spatial network and the predicate of each individual query,  $\mathcal{C}_v$  also varies from the implementation: In *dynamic network expansion* [51], evaluating a node-based query usually involves multiple edge-based operations, hence  $\mathcal{C}_v > \mathcal{C}_s$ ; While in *solution indexing* [49], the query result is cached to speed-up this operation, and evaluating a node-based query involves a simple structure look-up operation, therefore  $\mathcal{C}_v \approx \mathcal{C}_s$ . In our current system, we set  $\mathcal{C}_v$  according to the underlying implementation and the average condition of the spatial network and no specific predicates.

**Theorem 1** *Let  $q$  denote a query issued at some position  $p$  on segment  $s$  and  $v_b^s$  and  $v_e^s$  as the two ends of  $s$ . The exact query result  $\mathcal{R}(q, p)$  is in the union of two parts: (1) matching objects on the segment  $s$ ; (2) results of the query on both end of  $s$ , formulated as follows:*

$$\mathcal{R}(q, s) \subseteq \mathcal{O}_s(q, s) \cup \mathcal{O}_v(q, v_b^s) \cup \mathcal{O}_v(q, v_e^s)$$

The proof of this theorem is straightforward and omitted here. Figure 4.3 gives an illustrative example of anonymous query processing on road network. A  $k$ -NN query  $q$  with  $k = 3$  is issued by a user  $u$  locating on the segment  $\overline{v_5v_6}$ . The exact answer for query  $q$  is  $R(q, p) = \{o_5, o_6, o_7\}$ , which is included in the union of  $\mathcal{O}_e(q, \overline{v_5v_6}) = \{o_5, o_6\}$ ,  $\mathcal{O}_v(q, v_5) = \{o_1, o_6, o_7\}$  and  $\mathcal{O}_v(q, v_6) = \{o_5, o_3, o_4\}$ .

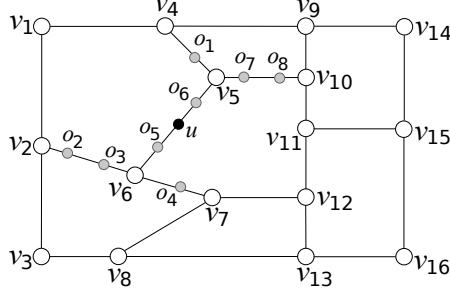


Figure 4.3: Illustration of query processing on road networks.

We extend the query processing model for single segment anonymous location to the set of segments  $S$  by employing the concept of border nodes (See Definition 2). Concretely, the query result of  $q$  with  $S$  as its spatially cloaked location is a subset of the union of (1) the set of matching objects on the segments of  $S$ , i.e.,  $\cup_{s \in S} \mathcal{O}_s(q, s)$  and (2) the set of matching objects on the border nodes of  $S$ , i.e.,  $\cup_{v \in BV(S)} \mathcal{O}_v(q, v)$ . Formally,

$$\mathcal{R}(q, S) \subseteq (\cup_{s \in S} \mathcal{O}_s(q, s)) \cup (\cup_{v \in BV(S)} \mathcal{O}_v(q, v))$$

Finally, with this model, the evaluation cost of anonymous query  $q$  with cloaked location  $S$ , denoted by  $Cost_{eval}(S)$ , can be estimated as follows:

$$Cost_{eval}(q, S) = \mathcal{C}_v \times |BV(S)| + \mathcal{C}_s \times |S| \quad (4.1)$$

in which  $|BV|$  denote the number of border nodes in the anonymized location  $S$  and  $|S|$  denote the number of segments in  $S$ .

#### *Cost of Communication Cost*

In Section 4.2.3 we present reference anonymous query processing architecture and data flow in four communication phases. We analyze the cost of communication in these four phases respectively. Specifically, we focus on the "optimizable" communication, i.e., the cost that varies from the performed location anonymization technique. We measure the cost as the length of the sent and received messages, and use  $\|x\|$  denote the encoded length of

an object  $x$ . For a given query  $q$ , the communication cost in Phase I and IV are not usually not optimizable, since each service request usually takes fixed encoded format, e.g., query ID, query parameters, etc; and the size of the exact answer to  $q$  is fixed, for a given database. Therefore, we focus on the communication cost in Phase II and III.

In Phase II, the query remains intact while the location information is sanitized as a set of segments  $S$ , therefore the communication cost is:  $Cost_{comm-II} = \|q\| + \|S\|$ ; In Phase III, the candidate result  $\mathcal{R}(q, S)$  is returned with the communication cost as  $Cost_{comm-III} = \|\mathcal{R}(q, S)\|$ . As discussed above, a query  $q$  usually takes fixed length. Also, for given location privacy requirements,  $k$ -user anonymity and  $l$ -segment indistinguishability, the number of segments in  $S$  tends to be fairly stable. Therefore  $Cost_{comm-III}$  is the major "optimizable" cost, which can vary significantly depending on the performed anonymization algorithm, and we consider as the dominant communication cost.

As discussed in Section 4.2.4, given a location query  $q$  and the set of segments  $S$ , the size of the candidate result  $\mathcal{R}(q, S)$  can be estimated as:

$$|\mathcal{R}(q, S)| \approx R \cdot |BV(S)| + \sum_{s \in S} \sum_{e \in s} |\mathcal{O}_e(q)|$$

where the first term corresponds to the result size of a query over the boundary nodes of  $S$ , and the second term represents all the objects on the edges of  $S$ . Let  $R$  be the average result size for the query over the node ( $R = k$  for the  $k$ -NN queries),  $\rho_o$  be the average number of object on an edge, and  $C_o$  be the cost of sending/receiving an object  $o$ . The communication cost for a typical query with anonymized location  $S$  is therefore expressed as:

$$Cost_{comm}(S) = C_o \cdot \left[ R \cdot |BV(S)| + \rho_o \cdot \sum_{s \in S} \sum_{e \in s} |e| \right] \quad (4.2)$$

### *Overall Anonymous Query Cost*

Given the estimation of the query evaluation cost and the communication cost in Equation 4.1 and 4.2, it is desired to combine them to give an estimation of the overall cost. In this work, we consider a linear combination scheme:

$$Cost(\mathcal{S}) = \lambda.Cost_{comm}(\mathcal{S}) + (1 - \lambda).Cost_{eval}(\mathcal{S}) \quad (4.3)$$

where  $\lambda$  is the parameter tinning the trade-off between the evaluation cost (mainly CPU computation at the server side), and the communication cost (mainly bandwidth of wireless channel).

#### 4.2.5 Inference Attack Models

The strength of location privacy techniques depends on the attacker's level of confidence in his estimation of the user's exact location. Specifically, in the road network-based location obfuscation systems, the goal of the adversary is to identify probabilities of each segment in anonymous location  $S$  to be the user's actual segment. Ideally, each segment in  $S$  is indistinguishable to the adversaries. In other words, associativity of the mobile user with the segments follows a uniform distribution (i.e., with equal probability  $1/|S|$ ). However, with available background knowledge attacker can associate user to a certain segment with higher probability. In order to formalize attacker's confidence we used the notion of *Linkability* [45]:

**Definition 6 (Linkability)** *For a user  $u$ , with published location information as a set of segments  $S$ , Linkability  $Link(u, s, S, K_B)$  is the probability that an adversary can infer based on  $S$  and background knowledge  $K_B$  that  $u$  is associated with the segment  $s$ .*

The background knowledge particularly addressed here includes (1) the location anonymization algorithm, (2) the underlying road network structure, and (3) the estimation of overall

query cost (Section 4.2.4) of each segment. In this work, we primarily focused on snapshot queries. The ability to correlate multiple locations with single user in continuous queries increase attacker’s chance significantly to infer user’s exact location [52]. While addressing privacy breach in continuous queries is an important direction and our ongoing work, we limit our scope in this work with snapshot queries. Following, we present General Replay Attack [45], Correlation-based Replay Attack, and Query Injection Attack. These attack models used as a resilience measure of the location anonymization algorithms.

### *General Replay Attack*

In the replay attack the adversary is given the set of segments  $S$  and attempts to perform reverse engineering, with her understanding the sanitation algorithm. Specifically, the adversary re-runs the anonymization algorithm,  $A(\cdot)$ , for each segment  $s \in S$  assumed to be the mobile user’s original location. The similarity between  $S$  and the segment set,  $S'$ , generated with replay attack model is used as a metric to estimate the likelihood of  $s$  to generates the anonymous location  $S$ :

$$Likelihood[S|u \leftarrow s, K_B] = \frac{|S' \cap S|}{|S|}$$

where  $|S'| \geq |S|$ . Using the likelihood value of each segment linkability is calculated as follows:

$$Link[u \leftarrow s^*|S, K_{ad}] = \frac{Likelihood[S|u \leftarrow s^*, K_B]}{\sum_{s \in S} Likelihood[S|u \leftarrow s, K_B]}$$

The segment with highest linkability value selected as an estimated location of the mobile user. Algorithm 1 shows the details of the general replay attack model. Starting from each segment (line 2), until the number of segments in  $S'$  is reach to the size of the anonymous location  $S$ , one incrementally add new segments using anonymization algorithm  $A(|S'|)$  (lines 3-4). The likelihood and linkability values calculated in lines 5 and

6, respectively. The segment with the highest linkability value return as the output of the replay attack (line 7).

---

**ALGORITHM 1:** Basic Replay Attack

---

**Input:** the set of segments  $S$ , the anonymization algorithm  $A(\cdot)$

**Output:** the maximum linkability value

```

1 for each  $s \in S$  do
2    $S' \leftarrow \{s\}$ ;
3   while  $|S'| < |S|$  do
4     add new segment(s) to  $S'$  with  $A(|S'|)$ ;
5     calculate  $L[S, K_{ad}|u \leftarrow s] = \frac{|S' \cap S|}{|S'|}$ ;
6   calculate  $Link[u \leftarrow s | \S, K_{ad}] = \frac{L[S, K_{ad}|u \leftarrow s]}{\sum_{s' \in S} L[S, K_{ad}|u \leftarrow s']}$ ;
7 output  $s | \max_s Link[u \leftarrow s | S, K_{ad}]$ ;

```

---

*Correlation-based Replay Attack*

The basic idea of road network based anonymization is to add extra segments in addition to the original segment that user resides to form the anonymous location  $S$  until it satisfies all users' privacy requirements (Section 4.2.2). We classify such algorithms in two categories based on the new segment selection mechanism. In the first category, algorithms selects a new segment to add anonymous location  $S$  with predefined segment selection mechanism only. Random Sampling and Network Expansion 4.2.6 are examples of this category. In the second category, the decision to select a segment to add anonymous location require to take into consideration other queries reside on the new segment. For example, XStar [45] enlarge anonymous location  $S$  with new star if and only if the star contains at least one active query. However, general replay attack model does not consider this distinction and calculates likelihood for each segment based on the road network structure only. To overcome this deficiency, in our new attack model we incorporate the placement of other  $(k - 1)$  queries on the cloaked subgraph  $S$  to the general replay attack model. In this case, likelihood of  $s$  is the sum of all likelihood values for each possible placement of the  $(k - 1)$  queries over  $S$ . Let  $S$  is the multiset, we define all possible placements,  $M$ , as

a list of  $(k - 1)$ -multisubset. Suppose that  $m_i \leftarrow \overline{S}$  shows the selected segments for the mutisubset  $m_i \in M$ , likelihood can be calculated as:

$$Like[S|u \leftarrow s^*, K_{ad}] = \sum_{i=1}^{\binom{S}{k-1}} Like[S|u \leftarrow s^*, m_i \leftarrow \overline{S}, K_{ad}]$$

In addition to background knowledge described earlier we also considered stronger attacker that has additional knowledge about mobile users distribution over road-network. Correlation-based replay attack make it possible to include this information to quantify adversaries success. We, first, calculate the relative probabilities  $p(s)$  based on the popularities of each segment and each multisubset's probabilities  $p(m)$ . Finally, during the likelihood calculation we used this probabilities as follows:

$$Like[S|u \leftarrow s^*, K_{ad}] = \sum_{i=1}^{\binom{S}{k-1}} p(s^*).p(m_i).Like[S|u \leftarrow s^*, m_i \leftarrow \overline{S}, K_{ad}]$$

### *Query Injection Attack*

So far we assume that attacker is an observer only and does not have knowledge about any user's location in the  $k$ -anonymous cloaked region  $S$ . In this attack model, we also consider the case where the attacker play active role and can inject dummy queries into the system. Although anonymized location  $S$  may not satisfy user  $k$ -anonymity anymore, effective road network based obfuscation algorithms continue to satisfy segment  $l$ -indistinguishability requirement.

Anonymization algorithms with strong relationship between generated cloaked region and query issued locations, are vulnerable against query injection attack. For instance, in XSTAR during the super star construction step new star selected if and only if it includes at least one active query and all queries belongs the same star should cloak together. This expose additional information to the attacker. Figure 4.4 shows an example cloaked sub-

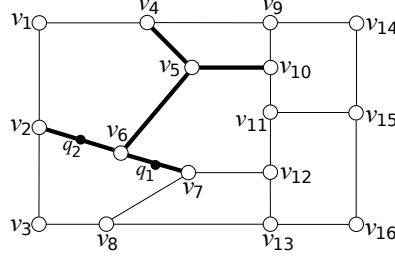


Figure 4.4: Query Injection Example

graph with bold lines for three queries. Suppose that attacker send two fake queries,  $q_1$  and  $q_2$  with  $(\delta_k, \delta_s)=(3,3)$  privacy requirements. Due to the nature of the XSTAR algorithm, attacker infers that third query issued from either segment  $\overline{n_4 n_5}$  or  $\overline{n_4 n_{10}}$ . In order to measure the effectiveness of the anonymization algorithm respect to query injection attack we slightly modify the likelihood calculation in the correlation-based replay attack. In this case, we assign zero to the likelihood value for the multisubset  $m_i \in M$ , if the corresponding segments  $\overline{S}$  conflicts with the known queries location.

On the other hand, in STARCLOAK depend on the third user's segment diversity requirement it is possible to be on any of the segments in  $S$ . Note that we assumed that privacy requirements sent anonymization server with secure channel and attacker do not have this information.

#### 4.2.6 Baseline Anonymization Models

In this section, we motivate the design of STARCLOAK by presenting two baseline location anonymization models, *random sampling* and *network expansion*, represent two ends of the spectrum for utility-aware location-anonymization with privacy  $(k, l)$  and utility metrics  $(ss, tt)$ . Extra segment selection mechanism differentiate the **quality** of location-anonymization schemes.

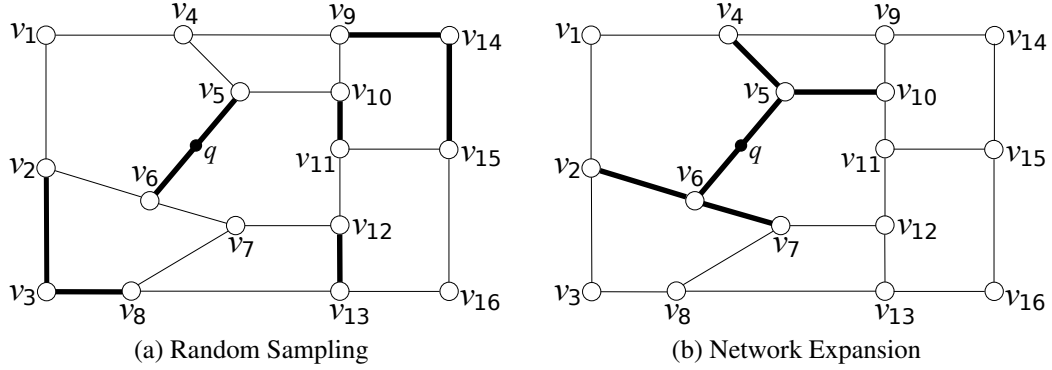


Figure 4.5: Baseline location-anonymization models.

### *Location Anonymization by Random Sampling*

Let  $q$  denote a user query with privacy constraint  $(k_q, l_q)$  and utility constraints  $(ss_q, tt_q)$ . This approach samples one segment with random from the spatial region defined by  $ss_q$  and adds it to the anonymous location  $S$  of  $q$ . The process continues until the two privacy requirements  $(k_q, l_q)$  are satisfied. Figure 4.5a shows the anonymization result of an example query  $q$  with  $(k_q, l_q) = (5, 5)$ . By *Random Sampling* scheme, four additional segments are selected randomly, each containing an active user query, in addition to the original segment  $\overline{v_5 v_6}$  to which query  $q$  by user  $u$  is associated. Selected segments shown with bold edges in Figure 4.5a.

The strength of random sampling scheme is its high resilience against statistical inference attacks since the set of segments is selected with random under the spatial utility constraint. However, the weakness of the scheme is the high anonymous query processing cost using the cloaked region  $S$  with at least  $l$  segments and a large number of border nodes,  $|BV| \geq 2 \times l$  in the worst case scenario. Thus, the query processing cost corresponds to issuing  $l$  or more queries at different locations, leading to high query processing cost as  $l$  and  $k$  increases.

### *Location Anonymization by Network Expansion*

In contrast to random sampling, network expansion represents another end of the spectrum of location anonymization algorithms, namely, one can perturb the focal location point of query  $q$  using a network expansion algorithm [51]. It starts from the segment on which query  $q$  is issued, say  $\overline{v_5v_6}$ . Following Dijkstra’s algorithm, we incrementally add a neighboring segment, ordered by their network distances (mid points) to  $q$ ’s focal position. The process terminates when  $q$ ’s privacy requirements defined by  $(k, l)$  are met. In Figure 4.5b, four segments with the minimum network distance to  $q$ ’s focal position are added incrementally into  $S$ , which defines  $q$ ’s anonymized location. The *Network Expansion* scheme results in a set of segments connected via a densely compact subgraph structure. The advantage of this scheme is the minimal query processing cost, as the number of border nodes grows sublinearly with the number of segments [51]. The main weakness of this scheme is the vulnerability against replay attack as the expansion process follows a deterministic best-first search strategy, which can be potentially exploited by an adversary to perform a reverse-engineering attack.

From these two naive baseline anonymization schemes, we observe that even with utility-aware location anonymization, there may exist multiple anonymized locations that meet both privacy metrics  $(k, l)$  and utility metrics  $(st, tt)$ . Thus, we employ a query-cost based optimization to select the spatially cloaked location that will generate the cheapest query cost.

#### 4.2.7 Solution Approach

Based on the analysis of the strengths and weaknesses of the two baseline anonymization methods, we demonstrate that simply meeting the two location privacy metrics and the two location utility metrics is insufficient to address the problem of utility-aware and attack resilient location anonymization. For example, anonymization solutions that satisfy both location privacy metrics and location utility metrics, such as random sampling and network

expansion, may fail to provide location anonymization that offers both strong privacy guarantee and low cost of anonymous query processing. A key challenge is that high utility and high privacy are conflicting goals. The random sampling based anonymization achieves the strong resilience against location spoofing attacks by paying the high cost of anonymous query processing, whereas the network expansion based anonymization achieves low processing cost for anonymous queries by employing Dijkstra’s deterministic network expansion, making it vulnerable to replay attacks and various background knowledge based attacks. Thus, we argue that an *effective location anonymization algorithm should aim at achieving an optimal balance between low query processing cost, high spatial-temporal location utility, and high privacy risk resilience*. Concretely, adding certain amount of randomness in the baseline network expansion approach can alleviate the problem of deterministic expansion, which is a main cause of vulnerability for replay attacks with background knowledge. By adding random noise into the location anonymization decision process, we can introduce non-deterministic nature into the location cloaking computation, which strengthens the attack resilience of both the computation model of the anonymization algorithm and its anonymized location output. However, adequate control of the amount of random noises is required to avoid the high cost of anonymous query processing resulting from the random sampling based approach.

Bearing the above analysis in mind, we formulate the problem of utility-aware and attack resilient location anonymization in the STARCLOAK system as follows:

**Definition 7 (Utility-aware and attack-resilient Anonymization)**

*Let  $q$  denote a location query with focal location  $L_q$  issued by a mobile user  $u$ . Let  $(k, l, st, tt)$  denote the user-defined privacy and utility constraints with respect to query  $q$ . We define that the location anonymization is a process that transforms the exact location  $L_q$  associated with query  $q$  to a cloaked location  $S$  and the achieved privacy and utility measures  $(\delta_k, \delta_l, \sigma_s, \sigma_t)$  associated with  $S$ . We say that a location anonymization algorithm, denoted by  $\alpha$ , is utility-aware and attack resilient if and only if (iff) the following conditions*

hold:

1.  $\mathcal{S}$  is a subgraph of the road network  $G$ ;
2.  $\mathcal{S}$  satisfies location  $k$  user anonymity ( $\delta_k \geq k$ ), location  $l$  segment-indistinguishability ( $\delta_l \geq l$ );
3.  $\mathcal{S}$  preserves both spatial resolution constraint ( $\sigma_s \leq st$ ) and temporal resolution constraint ( $\sigma_t \leq tt$ );
4. Let  $\mathcal{S}$  is the anonymized location output of the algorithm  $\alpha$ .  $\mathcal{S}$  is the optimal cloaking location with respect to the best balance between high location utility with low anonymous query processing cost and strong privacy with high attack resilience. Formally, let  $\mathcal{A}_{\mathcal{S}}$  denote the set of candidate cloaked locations.  $\mathcal{S}$  satisfies the following condition:  $\forall \mathcal{S} \in \mathcal{A}_{\mathcal{S}}, cost_{total} cost_{total}^{\alpha}(\mathcal{S}) \leq cost_{total}^{\alpha}(\mathcal{S})$ .

This problem statement amounts to say that a location anonymization algorithm delivers an optimal balance between location privacy guarantee and location utility guarantee. The location privacy guarantee should satisfy user-defined location privacy metrics and offer strong attack resilience to the location privacy guarantee provided. The location utility guarantee should meet the user-desired spatial and temporal utility bounds and ensure the low cost of anonymous query processing.

Given that the cost of anonymous query processing is a function of the size of the border nodes,  $|BV(\mathcal{S})|$ , and the size of the edge set of anonymized focal location  $|\mathcal{S}|$ , in the design of STARCLOAK location anonymization algorithm, we focus on developing controlled randomness in each anonymization step of the neighborhood expansion, while minimizing the size of the border node set and the edge set of  $\mathcal{S}$ . STARCLOAK achieves these objectives from three novel aspects. First, STARCLOAK introduces *star-graph*, a road-junction based abstraction and a cost-aware randomness in road-junction based expansion process to reduce the high query cost of random sampling and improve the privacy risk resilience of

baseline network expansion. Second, STARCLOAK introduces a *clique-cloak graph* to generate candidate cloaking-star graphs by grouping queries that can be cloaked together based on their privacy profiles. Finally, STARCLOAK introduces utility-aware and cost effective pruning strategies to generate final cloaking star graph that has low cost of anonymous query process and high attack resilience among the set of candidate star-graphs. Intuitively, STARCLOAK achieves the high attack resilience by employing cloaking star as the basic unit of location anonymization and by injecting controlled randomness into the cloaking-star selection process. At the same time, STARCLOAK assures the low query processing cost by employing the cost-aware star selection scheme to generate the final cloaking star graph with more compact structure.

### 4.3 STARCLOAK: Model and Algorithms

In this section, we introduce the design of STARCLOAK in terms of the *cloaking star*-based location cloaking model and algorithms. We first describe the concept of cloaking star and related concepts. Then we introduce a set of core data structures deployed for implementing the STARCLOAK algorithms before we detail the cloaking graph update, the candidate cloaking star selection and composition, and the cloaking star pruning algorithms.

#### 4.3.1 Star Structure and Star Graph

In the real world, mobile travelers move on either a spatially constrained road network or a walking paths. Thus, we model spatially constrained networks using the graph model of a road network (recall Section 4.2.1). Unlike the conventional location anonymization solutions, STARCLOAK captures both  $k$ -user anonymity and  $l$ -segment indistinguishability. In addition, unlike the conventional solutions, which are based on the random way-point mobility model and use a rectangular region as a unit of location cloaking, STARCLOAK introduces a star as the basic unit of location cloaking and refers to each road junction as a star of the road network graph,  $G = \langle V_G, E_G \rangle$ . Each star is defined by a vertex with its

adjacency segment list in  $G$ .

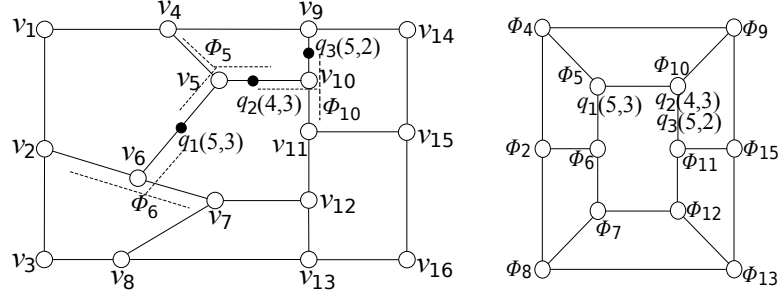


Figure 4.6: Illustration of the STAR model.

**Definition 8 (Star)** Let  $G = \langle V_G, E_G \rangle$  denote the road network of interest. We define a star  $\phi_i$  anchored at vertex  $v_i \in V_G$  as a subgraph of  $G$ , denoted by  $\phi_i = \langle V_\phi^i, E_\phi^i \rangle$ , and  $V_\phi^i = \{v_i\}$  and  $E_\phi^i = \{w \mid w \neq v_i, w \in V_G, (v_i, w) \in E_G\}$ .

According to this definition, every node  $v_i$  with  $d_G(v_i) \geq 3$  is associated with a unique star  $\phi_i$ , which consists of vertex  $v_i$  and all of its adjacent road segments, that is, those segments with  $v_i$  as one of two end nodes. For example, in the left plot of Figure 4.6, star  $\phi_5$  comprises node  $v_5$  and segments  $\{\overline{v_5v_4}, \overline{v_5v_6}, \overline{v_5v_{10}}\}$ .

Introducing the star structure as the basic unit of anonymization has a number of advantages: (1) It preserves the locality of neighboring segments of a vertex and is expected to lead to the compact structure of the anonymous location; (2) it provides high attack resilience against deterministic segment selection process in order to provide high locality; (3) it allows a compact star indexing structure since each star can be uniquely identified by its anchor node identifier, which corresponds to both the node itself and all of its neighboring segments (its adjacency list) in  $G$ ; (4) its compact representation of the anonymized locations also simplifies the implementation of the STARCLOAK algorithms and reduces the storage as well as communication costs of anonymous queries.

**Definition 9 (Star Graph)** Let  $G = (V_G, E_G)$  denote a road network, and each node  $v_i \in V_G$  ( $1 \leq i \leq |V_G|$ ) corresponds to a star  $\phi_i$  with  $v_i$  as its anchor node. The star network

of  $G$ , denoted by  $G_\phi = (V_{G_\phi}, E_{G_\phi})$ , can be constructed as follows: (1) each node of  $G_\phi$  is a star in  $G$ , denoted by  $\phi_i$ , with node  $v_i \in V_G$  as its anchor node; (2) two nodes of  $G_\phi$  are adjacent if their corresponding stars in  $G$  share a segment, i.e.,  $\forall \phi_i, \phi_j \in V_{G_\phi}$ , if  $(\phi_i, \phi_j) \in E_{G_\phi}$ , then  $(v_i, v_j) \in E_G$ . All edges in the star graph  $G_\phi$  are of unit length. Thus, the distance between two stars  $\phi_i$  and  $\phi_j$  in a road network  $G$  is measured by their network hop distance in  $G_\phi$ , denoted by  $h_G(\phi_i, \phi_j)$ , which is the hop count of the shortest path between  $\phi_i$  and  $\phi_j$ .

Figure 4.6 shows an example road network (left) and its corresponding star network (right).  $h_G(\phi_6, \phi_{10}) = 2$ , as their shortest path in  $G_\phi$  is composed of  $\phi_6, \phi_5$  and  $\phi_{10}$ .

### 4.3.2 Data Structures

To facilitate the presentation of the star-based location anonymization model, in this section, we describe important data structures used in the STARCLOAK.

**Query Queue,  $\mathcal{Q}$ :** It is a first in first out (FIFO) basic queue structure, and records the incoming location queries that require to go through the location-anonymization process before relaying them to the respective LBS providers. The incoming location queries are inserted into the queue from the tail and the anonymization engine pops new query from  $\mathcal{Q}$  to find its cloaked subgraph as its anonymized location  $\mathcal{S}$ . Upon receiving a location query  $q$ , issued by user  $u$ , the location anonymization engine will transform the query focal location  $L_q$ , the exact position of  $u$ , to an anonymized location, a subgraph that includes the road segment on which  $L_q$  resides. Anonymization computation is based on the both privacy  $(k, l)$  and utility  $(st, tt)$  constraints associated to the query  $q$  and its user  $u$  and other queries currently residing in the query queue  $\mathcal{Q}$ . The temporal resolution tolerance constraint can be viewed as the temporal expiration deadline of the query.

**Expiration Heap,  $H$ :** It is a max-min heap that maintains the queries in the order of their expiration time defined by their temporal resolution constraint,  $tt$ . Anonymization engine checks  $H$  to identify queries that are close to their expiration deadlines and prioritize

the queries for anonymization processing or to identify queries that have been expired and remove them from the query queue  $\mathcal{Q}$ .

**Cloaking Graph,  $G_C$ :** It is a undirected graph dynamically constructed in-memory. It records the set of queries associated to a star based on their similarities w.r.t. their privacy requirements, their spatial proximity and their expiration deadlines. The structure and the usage of the Cloaking Graph will be explained in Section 4.3.6.

**Star-Map,  $M_S$  and Query-Map,  $M_Q$ :** We create a hash map to index stars, called Star-Node Map, and similarly, another hash map data structure to index queries associated to a node in the Cloaking Graph, called Query-Node Map.

**Candidate Star Set Queue,  $\mathcal{Q}_C$ :** It is a FIFO-based queue structure that records generated candidate cloaking-star sets. Pruning engine pops first arrived set from  $\mathcal{Q}_C$  and apply utility-aware and cost-effective pruning algorithm to generate final cloaking-star graph.

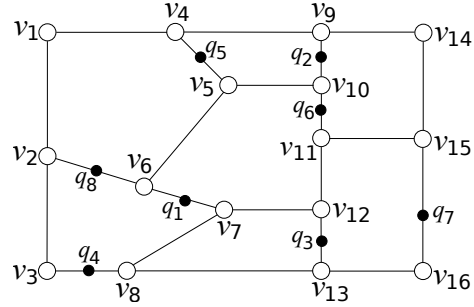
### 4.3.3 Location Query Preprocessing

In STARCLOAK, we first parse an incoming location query  $q_i$  to generate the internal representation of the location query by performing a sequence of tasks as follows:

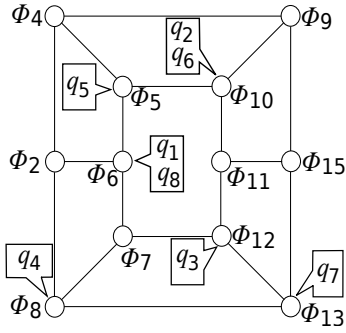
- i) Assign a unique identifier to the query, using secure hash function with user id and query issue time (i.e.,  $h(q_i.u_{id} || q_i.t)$ ). Assumed that any user can send only one query in each time-stamp.
- ii) Determine the road segment of a query focal location defined with latitude and longitude values using spatial index.
- iii) Insert query to anonymization queue  $\mathcal{Q}$ .
- iv) Insert query to the expiration heap ( $H$ ) with the query deadline as the key and the query identifier as the value. Query expiration time is the sum of query issued time  $t$

query	$k$	$l$	$ss$
$q_1$	4	8	2
$q_2$	3	12	3
$q_3$	5	11	2
$q_4$	4	10	1
$q_5$	3	6	3
$q_6$	2	4	1
$q_7$	5	9	4
$q_8$	2	5	3

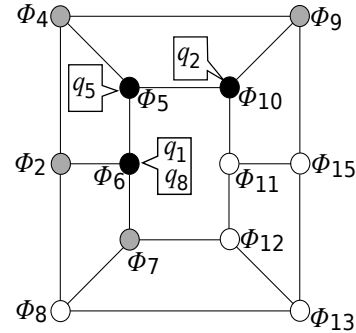
(a) Star Assignment



(b) Road Network



(c) Star Network



(d) Selected Stars

Figure 4.7: StarCloak Example

and user-defined temporal tolerance  $tt$  associated with the query:

$$t_e = q_i.t + q_i.tt$$

#### 4.3.4 StarCloak in a Nutshell

Before getting into details of each steps, first, we summarized the STARCLOAK approach. Algorithm 2 sketches the main procedure of the location anonymization in the STARCLOAK. Location-anonymization engine continuously pops queries from the query queue,  $\mathcal{Q}$ , and process them to find anonymized location  $S$ , which satisfies privacy and utility requirements. Before starting to process a new query  $q$ , STARCLOAK removes all expired queries from the system (line 4-12). When an expired query,  $q_e$ , removed from a cloaking graph node, because of the different privacy and utility requirements it is possible to find

---

**ALGORITHM 2:** StarCloak Algorithm

---

**Input:**  $Q$ : query queue,  $H$ : expiration heap,  $G_C$ : cloaking graph**Output:**  $Q_C$ : candidate star-set queue

```
1 while true do
2   if  $Q \neq \emptyset$  then
3      $L_u \leftarrow \emptyset$ ;
4     while true do
5        $q_e \leftarrow$  first entry of  $H$ ;
6       if  $q_e$  is expired then
7          $n_u \leftarrow$  removeQueryfromCloakingGraph( $q_e$ );
8         if  $n_u \neq \emptyset$  then
9            $\lfloor$  Add  $n_u$  to the  $L_u$ ;
10         $\lfloor$  Pop  $q_e$  from  $H$ ;
11       else
12          $\lfloor$  break;
13       foreach  $n_u \in L_u$  do
14          $C \leftarrow$  SearchStarSet( $n_u$ );
15         if  $C \neq \emptyset$  then
16            $\lfloor$  Add  $C$  to the  $Q_C$  for pruning;
17        $q_i \leftarrow$  first entry of  $Q$ ;
18        $\phi_i \leftarrow$  SelectStar( $q_i$ );
19        $n_u \leftarrow$  addQuerytoCloakingGraph( $q_i, \phi_i$ );
20        $C \leftarrow$  SearchStarSet( $n_u$ );
21       if  $C \neq \emptyset$  then
22          $\lfloor$  Add  $C$  to the  $Q_C$  for pruning;
```

---

cloaked subgraph for the remain queries in the node. All updated nodes stored in the list  $L_u$  after the expired queries removed and STARCLOAK attempts to find anonymized location for those nodes before processing new query (line 13-16). Next, it pops new query from the queue  $Q$  and selects star to assign it (line 18). Thereafter, StarCloak updates cloaking graph with the new query  $q_i$  and search possible cloaked location for the updated cloaking graph node (line 19-22). Finally, in the pruning phase it randomly selects and removes non-active stars from candidate star sets. Details of the each phase presented in the following sections.

### 4.3.5 Select Star

STARCLOAK anonymization engine performs location anonymization by scanning through the FIFO query queue  $\mathcal{Q}$ . All segments that are associated with active queries are marked as *active*. The anonymization engine first selects a star to assign queries on the active segment as the initial cloaking star. Each segment has two end nodes and if both end nodes hold  $d_G(v_s) \geq 3$  and  $d_G(v_t) \geq 3$ , the segment-to-star mapping module needs to determine to which star this active segment  $s$  should be assigned,  $\phi_s$  or  $\phi_t$ . For example, in Figure 4.6 when the query  $q_1$  arrived and segment  $\overline{v_5v_6}$  become active one of the two possible stars,  $\phi_5$  and  $\phi_6$ , is to be determined as an initial cloaking star. Given a road network, one needs to choose a set of stars  $\{\Phi\}$  to cover all the active segments. If a star  $\phi$  is selected for a given active segment  $s$ , it is said that  $\phi$  is “selected”, and  $s$  is “assigned” to  $\phi$ , denoted by  $s \leftarrow \phi$ .

To save the computation cost of query evaluation at the service provider and the communication bandwidth between the service provider and anonymization server, in STARCLOAK we use a cost model based scheme for mapping active segments to stars. Formally, let  $Cost(\phi)$  be the cost of executing a typical query with the anonymous location  $\phi$ ,  $AS$  denote the set of current active segments in the road network  $G$ , and  $\Phi$  be the set of selected stars, then the minimization of the overall cost can be formalized as follows:

$$\begin{aligned} \min_{\Phi} \sum_{\phi \in \Phi} Cost(\phi) & \quad (4.4) \\ \text{s.t. } \forall s \in AS, \exists \phi \in \Phi, s \leftarrow \phi & \end{aligned}$$

This cost-aware segment-to-star assignment scheme aims at finding a set of stars  $\Phi$  that cover all the active segments under anonymization consideration, and yet have the lowest overall cost of anonymous query processing. However, no efficient solution to this optimization problem exists, unless  $P = NP$ , as shown in the following theorem:

**Theorem 2** *The optimization problem in Expression 4.4 is NP-Hard.*

**Proof 1** *By removing all the non-active segments in the network  $G$ , this problem is deduced to the weighted VERTEX-COVER problem, which is known to be NP-Complete. Specifically, if  $\forall \phi \in G, Cost(\phi)=1$ , i.e., all the stars are associated with identical cost, then the problem is equivalent to the classical VERTEX-COVER problem, which is also NP-Complete. Therefor the optimization problem in 2 is NP-Hard.*

Instead of attempting to find a global optimal solution, we propose an efficient randomized algorithm that can find high-quality approximate solution, and is robust against inference attacks. The main idea of our randomized algorithm is to select one of the two stars based on the probability in reverse proportion to their corresponding query processing costs. We illustrate this randomized algorithm below through query insertion operation.

The procedure of inserting a new arrival query  $q$  associated with segment  $s$  (Insert-Query) consists of the following four cases: (1) if certain star already covers  $s$ , the algorithm halts; (2) if both stars  $\phi_s$  and  $\phi_t$  are already selected, yet  $s$  is not covered, one assigns  $s$  to one of the two stars with probability in reverse proportion to their corresponding costs; (3) if only one star  $\phi_s$  or  $\phi_t$  has been selected,  $s$  is assigned to that star; (4) if neither  $\phi_{v_s}$  nor  $\phi_{v_t}$  is selected, one assigns  $s$  to the one with probability reversely proportional to the corresponding cost. Algorithm 3 shows the pseudo code of this procedure.

Essentially, this approach ensures that each active segment  $s$  is assigned to  $\phi_{v_0}$  with probability  $Cost(\phi_{v_1})/[Cost(\phi_{v_0}) + Cost(\phi_{v_1})]$ , or  $\phi_{v_1}$  otherwise. This property guarantees that the quality of the star set  $\Phi$  selected by our randomized algorithm does not deviate far from the optimal one. We formally describe this property in Theorem 3 below:

**Theorem 3** *Let  $Cost^{opt}$  be the cost achieved by the optimal star set. The randomized star-selection algorithm achieves the cost  $Cost^{rnd}$  satisfying  $\mathbb{E} [Cost^{rnd}] \leq 2 \cdot Cost^{opt}$ .*

The deletion of expired location queries can be easily incorporated in three steps: find the active segment containing the expired query, and de-register the query; (ii) if the segment is associated with no other active queries, the segment is claimed as non-active; and

---

**ALGORITHM 3:** InsertQuery Algorithm

---

**Input:**  $q$ : new query,  $I_\phi$ : active star index**Output:**  $\phi$ : selected star

```
1  $s \leftarrow$  the segment containing  $q$ ;  
   //  $\phi_t$  and  $\phi_s$  are two stars that share segment  $s$   
2 if  $s$  is already assigned to  $\phi_t$  (or  $\phi_s$ ) then  
3   return  $\phi_t$  (or  $\phi_s$ );  
4 else if  $\{\phi_s, \phi_t\} \cap I_\phi = \{\phi_{n_s^s}, \phi_{n_t^s}\}$  then  
   // both two stars are active but neither cover  $s$   
5   assign  $s$  to  $\phi_{n_s^s}$  w.p.  $\frac{\text{Cost}(\phi_{n_t^s})}{\text{Cost}(\phi_{n_s^s}) + \text{Cost}(\phi_{n_t^s})}$  or  $\phi_{n_t^s}$  otherwise;  
6 else if  $\{\phi_{n_s^s}, \phi_{n_t^s}\} \cap I_\phi = \phi_{n_s^s}$  (or  $\phi_{n_t^s}$ ) then  
   // only one star is active  
7   assign  $s$  to  $\phi_{n_s^s}$  (or  $\phi_{n_t^s}$ );  
8 else if  $\{\phi_{n_s^s}, \phi_{n_t^s}\} \cap I_\phi = \emptyset$  then  
   // neither star is selected yet  
9   if  $d_G(n_s^s) = 1$  or  $d_G(n_t^s) = 1$  then  
     // only one end corresponds to a star  
10    add  $\phi_{n_t^s}$  (or  $\phi_{n_s^s}$ ) to  $I_\phi$ ;  
11    assign  $s$  to  $\phi_{n_t^s}$  (or  $\phi_{n_s^s}$ );  
12  else  
     // both ends are intersection nodes  
13    assign  $s$  to  $\phi_{n_s^s}$  w.p.  $\frac{\text{Cost}(\phi_{n_t^s})}{\text{Cost}(\phi_{n_s^s}) + \text{Cost}(\phi_{n_t^s})}$  or  $\phi_{n_t^s}$  otherwise;  
14    add  $\phi_{n_s^s}$  (or  $\phi_{n_t^s}$ ) to  $I_\phi$ ;  
15 return  $\phi_t$  (or  $\phi_s$ );
```

---

(iii) if the star  $t$  covering the segment contains no other active segment, then this star is removed from the active star index  $I_\phi$ .

It is worth emphasizing that the quality of the selected stars does not degrade with continuous insertion/deletion of queries. This is because the algorithm makes no assumption about the order of the arrival of the queries, which is a desirable feature in supporting real-time road network based LBS.

#### 4.3.6 Cloaking Graph Update

We use cloaking graph to group nearby queries and efficiently index other query groups that can be cloaked together for easy access. We define cloaking graph as follows;

**Definition 10 (Cloaking Graph)** Let  $G_C(V_C, E_C)$  be an undirected graph where  $V_C$  is the set of nodes, each representing a set of requests, grouped by the star that they assigned and their location privacy and utility requirement similarities.  $E_C$  is the set of edges and there is an edge  $e = (n_i, n_j) \in E_C$  between two nodes  $n_i$  and  $n_j$  iff queries associated with both vertices can be cloak together based on their  $l$ -segment indistinguishability and spatial tolerance requirements.

In each node  $n_i \in V_C$  stores following information:

1. Corresponding star  $n_i.\phi$ . For each active star, there is at least one node in  $V_C$ .
2. Query set  $n_i.Q$ , which stores similar queries assigned to star  $n_i.\phi$ .
3. Combined privacy and utility requirements of the queries in  $n_i.Q$ , which we defined as follow:

$$\langle k_{n_i}, l_{n_i}, ss_{n_i} \rangle = \langle \max_{i=1}^{|n_i.Q|} k_{q_i}, \max_{i=1}^{|n_i.Q|} l_{q_i}, \min_{i=1}^{|n_i.Q|} ss_{q_i} \rangle$$

4. Covered star set  $n_i.\Theta$ , a set contains star identifiers, which are within  $ss_{n_i}$  hop distance from star  $n_i.\phi$ .
5. Segment count  $n_i.sc$ , number of segments associated with stars in star set  $n_i.\Theta$ .
6. Adjacency list  $n_i.N$ . Being neighbor indicates that requests in the corresponded nodes can be cloak together. Two cloaking nodes,  $n_i$  and  $n_j$ , have been considered as a neighbor of each others iff:

- (a) Stars associated with each node are an element of the star set of other node;

$$n_i.\phi \in n_j.\Theta \text{ and } n_j.\phi \in n_i.\Theta$$

---

**ALGORITHM 4:** Add Query to the Cloaking Graph

---

**Input:**  $q_n$ : new query,  $\phi_n$ : assigned star

**Output:**  $n_u$ : updated cloaking graph node

```
1  $n_u \leftarrow \emptyset$ ;  
2  $N \leftarrow M_S.get(\phi_n)$ ;  
3 if  $N \neq \emptyset$  then  
4   forall the  $n_i \in N$  do  
5     if  $ss_{q_n} < ss_{n_i}$  then  
6        $sc \leftarrow \#segment \text{ within } s_{q_n} \text{ from } \phi_n$ ;  
7       if  $sc \geq \max(l_{q_n}, l_{n_i})$  then  
8         add  $q_n$  to the node  $n_i$ ;  
9          $n_u \leftarrow n_i$ ;  
10        break;  
11      else if  $n_i.sc \geq l_{q_n}$  then  
12        add  $q_n$  to the node  $n_i$ ;  
13         $n_u \leftarrow n_i$ ;  
14        break;  
15 if  $n_u = \emptyset$  then  
16    $n_u \leftarrow$  create new node for query  $q_n$ ;  
17 return  $n_u$ ;
```

---

- (b) The number of segments that covered with both nodes is enough to satisfy their  $l$ -segment indistinguishability requirements. Let  $seg(I)$  is the corresponding segment set of star set  $I = n_i.\Theta \cap n_j.\Theta$ ;

$$|seg(I)| \geq \max\{l_{n_i}, l_{n_j}\}$$

In STARCLOAK, we consider personalized privacy and utility model, which allows users to define their requirements for each query they issued. Consequently, even though two queries associated with same star it is possible to not being able to cloak those queries together, because of the conflict between their  $l$ -segment indistinguishability and  $st$  spatial tolerance requirements. Therefore, each active star associates with one or more node in cloaking graph. We used map  $M_{SN}$  to easy access the list of nodes associated with certain star.

Cloaking-graph update requires in two situations: insert new query and remove expired query. Former case outlined in Algorithm 4. When function called to insert a new query, it checks all cloaking-graph nodes associated with corresponding star, to add new query. If there is not possible node to add, it creates new node. Concretely, first,  $N = M_{SN}.get(\phi_n)$  used to obtain list of nodes associated with star  $\phi_n$  (line 2).  $N$  implemented as a queue to increase the chance of generating cloak region for earliest queries added to the graph. New query can be added to an existing node if their privacy profile does not conflict with each other. Conflict occurs when new spatial tolerance is not able to satisfy new  $l$ -segment indistinguishability requirement. In other words, there are fewer number of segments than required within new spatial tolerance distance from the star. Each node  $n_i \in N$  evaluated to add new query (lines 4 to 14). If new query has less spatial tolerance requirement than the node, it first calculates the number of segment within the spatial tolerance of query from star  $\phi_n$ . If the number of segment is enough to satisfy segment indistinguishability requirement, new query added to the node. Similarly, new query added to a node, if it does not change the spatial tolerance of the node and there is enough number of segments within spatial tolerance coverage to satisfy segment indistinguishability requirement of the new query (lines 9-10). In case a new query cannot be added to the one of current nodes, STARCLOAK creates a new node for the new query (lines 15-16). It is important to mention that mobile users define their privacy and utility requirements intuitively and it is possible for them to set conflicted segment indistinguishability and spatial tolerance for their queries. This step detects such conflict in early stage and drops request immediately without waiting in the system until its expiration time.

To be more clear, we used a scenario given in Figure 4.7. In this example, Figure 4.7.a, shows incoming queries and their privacy requirements ( $k$ ,  $l$ , and  $ss$ ), ordered based on their issue time. For the sake of simplicity, we did not include temporal tolerance values and assumed that queries evaluated before they expired. Query locations on the road network and selected stars represented in Figure 4.7.b and Figure 4.7.c, respectively. In

Figure 4.8, we showed updated cloaking graph after each query added. For the first query  $q_1$ , assigned to the star  $\phi_6$ , algorithm generates new node  $v_6$  with the star set  $v_6.\Theta = \{\phi_6, \phi_2, \phi_5, \phi_7, \phi_4, \phi_8, \phi_{10}, \phi_{12}\}$ , stars within spatial tolerance ( $ss_{v_6} = 2$ ). When second query received, new node  $v_{10}$  created and connected with  $v_6$ , since intersection set of their star sets includes stars associated with two vertices. However, with the insertion of query  $q_3$ , even though the intersection set contains  $\phi_5$  and  $\phi_{12}$  there is no edge between node  $v_6$  and newly created node  $v_{12}$ . Because, the number of segments (9) belongs to stars in the intersection set is not enough to meet segment requirement of the node  $v_6$  (11). Query  $q_4$  immediately dropped from the system since it is not possible to satisfy user defined segment requirement with the given spatial tolerance value. When the next query,  $q_5$ , received, generated node  $v_5$  connected with the nodes  $v_6$  and  $v_{10}$ , but there is no edge between  $v_5$  and  $v_{12}$ , since  $h_G(\phi_5, \phi_{12}) = 3$ , higher than the spatial tolerance of the node ( $ss_{v_{12}} = 2$ ). Query  $q_6$  assigned to the star  $\phi_{n_{10}}$ , which is already associated with the node  $v_{10}$  in the graph. Algorithm checks if it is possible to add new query to the existing node. However, new spatial tolerance and segment diversity requirements cause conflict. In this case, another node  $v'_{10}$  associated with the same star generated for the new query and connected with possible nodes ( $v_5$ ). For query  $q_7$  created node  $v_{13}$  connected with nodes  $v_{10}$  and  $v_{12}$ . At this point there is no possible node set that satisfies all queries privacy requirements. Query  $q_8$  can be include to an existing node  $v_6$  without violating their requirements. Inclusion process requires to update node privacy profile with the maximum  $k$  and  $l$ , and minimum  $ss$  values. In our example, new privacy and utility profile is  $k, l, ss = 4, 8, 2$ . Spatial tolerance and segment indistinguishability values are not changed after the insertion, so it is not necessary to check update on edges connected with the node.

The procedure of cloaking-graph update, called after the expired query removal, summarized in Algorithm 5. If a cloaking-graph node, associated with expired query, contains other queries as well, STARCLOAK updates the node based on remain queries (lines 2 to

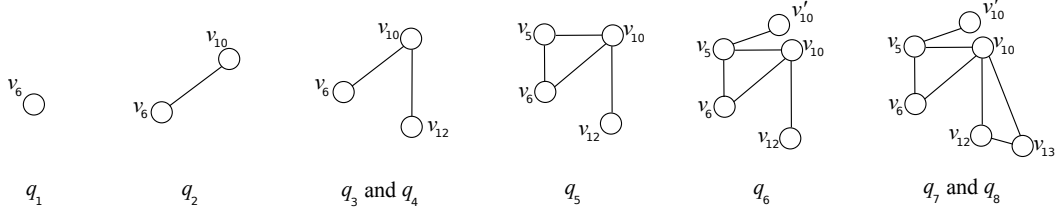


Figure 4.8: Cloaking Graph Update Example

7). It starts with updating privacy and utility profile of the node. Neighbor list can only change if segment indistinguishability requirement decrease or spatial tolerance increase. If the later case occur, the number of star and segment within spatial tolerance coverage increase.

---

**ALGORITHM 5:** Remove Query from Cloaking Graph

---

**Input:**  $q_e$ : expired query  
**Output:**  $n_u$ : updated cloaking graph node

- 1  $n_u \leftarrow MQN.get(q_e)$ ;
- 2 **if**  $|n_u.Q| > 1$  **then**
- 3     update  $k_{n_u}$ ,  $l_{n_u}$ , and  $st_{n_u}$ ;
- 4     **if**  $l_{n_u} < l_{q_e}$  or  $st_{n_u} > st_{q_e}$  **then**
- 5         **if**  $ss_{n_u} > st_{q_e}$  **then**
- 6             update  $n_u.\Theta$  and  $n_u.sc$ ;
- 7             update  $n_u.N$ ;
- 8 **else**
- 9     remove  $n_u$  form  $G_C$ ;
- 10      $n_u \leftarrow null$ ;
- 11     update  $M_S$ ;
- 12 remove  $q_e$  from  $M_Q$ ;
- 13 **return**  $n_u$ ;

---

Update procedure return a graph node that would be the input for the next step, candidate star-set selection. To eliminate unnecessary search, we checked if a node can generate cloaked region after expired query removed from the node. It is clear that, it is not possible to generate cloaked subgraph if the removing query does not decrease any of the privacy requirements or increase the number of neighbor node if the spatial tolerance increase.

### 4.3.7 Candidate Star-Set Selection

The goal of this step is to discover set of stars, called candidate star-set, which constitutes a possible anonymized sub-graph for certain queries. In order to find such star set, STARCLOAK searches over cloaking-graph and identifies set of nodes,  $NS$ , that satisfy privacy requirements of all queries associated with each node. Formally, let function  $seg(Star\ Set)$  returns all segments associated with input stars,  $NS$  meets  $k$ -user anonymity and  $l$ -segment indistinguishability requirements if and only if it satisfy following:

$$\forall n_i \in NS, k_{n_i} \leq \sum_{n_j \in NS} |n_j \cdot Q| \text{ and } l_{n_i} \leq |seg(\bigcap_{n_j \in NS} n_j \cdot \Theta)|.$$

Such  $NS$  forms candidate star-set  $\vartheta$  with all stars shared within the covered star set of each node in  $NS$ ;

$$\vartheta = \bigcap_{n_j \in NS} n_j \cdot \Theta$$

Before we explain the detail of candidate star-set selection algorithm, we elucidate contributory method, which checks node set if it satisfy privacy requirements, in Algorithm 6. First, it initialize the maximum  $k$ -user anonymity ( $max_k$ ), maximum  $l$ -segment indistinguishability ( $max_l$ ), total query count ( $k_c$ ), and covered star set from all nodes ( $S_{star}$ ) with the first node of the set (lines 1 to 3). Afterwards, for each other node in the set, it updates  $max_k$ ,  $max_l$ ,  $k_c$ , and  $S_{seg}$  as follows:  $max_k$  and  $max_l$  with the  $k$  and  $l$  values of the node, respectively, if they are higher than their maximum value;  $k_c$  with the sum of the  $k_c$  and the number of query in the node; and  $S_{star}$  with the union of the current star set and stars covered by new node (lines 4 to 8). If number of query is not enough to satisfy maximum  $k$ -user anonymity, it returns empty set. Otherwise, algorithm finds number of segments within the star set and checks if it is higher than the maximum  $l$ -segment indistinguishability requirement. If there is enough segments, algorithm return star set  $S_{star}$ , otherwise, returns empty set.

---

**ALGORITHM 6:** Requirement Check

---

**Input:**  $NS$ : node set  
**Output:**  $S_{star}$ : star set

- 1  $max_k \leftarrow k_{n_0}, max_l \leftarrow l_{n_0};$
- 2  $k_c \leftarrow |n_0.Q|;$
- 3  $S_{star} \leftarrow n_0.\Theta;$
- 4 **forall** the  $n_i \in NS - n_0$  **do**
- 5      $max_k \leftarrow max(max_k, k_{n_i});$
- 6      $max_s \leftarrow max(max_s, l_{n_i});$
- 7      $k_c \leftarrow k_c + |n_i.Q|;$
- 8      $S_{star} \leftarrow S_{star} \cap n_i.\Theta;$
- 9 **if**  $k_c < max_k$  **then**
- 10     $\text{return } \emptyset;$
- 11  $S_{seg} \leftarrow \emptyset;$
- 12 **forall** the  $\phi_i \in S_{star}$  **do**
- 13     $S_{seg} \leftarrow S_{seg} \cup \phi_i;$
- 14 **if**  $|S_{seg}| < max_l$  **then**
- 15     $\text{return } \emptyset;$
- 16 **return**  $S_{star};$

---

In Algorithm 7 we outlined candidate star-set selection step. Searching  $NS$  over cloaking-graph starts with the updated node in the previous step, cloaking-graph update. If the number of assigned query to the starting node is fewer than the  $k$ -user anonymity requirement of the node <sup>1</sup>, algorithm continues searching process over the neighbor nodes, which is ordered by the hop distance between their associated star and the starting node's star. For each neighbor node, it applies requirement check for the two nodes, updated and neighbor nodes, together (line 6-8). If candidate star set cannot be found, neighbor node evaluated with the all possible node combinations generated with the previously processed neighbor nodes. Possible node combination is the set of nodes that satisfy segment diversity requirement yet not contains enough query to meet k-anonymity requirement. After each neighbor node's evaluation process, if there is no candidate star set available, we update list of node combinations for the next neighbor node evaluation. Let  $Q_{Comb}^i$  is the all possible node combinations for the neighbor nodes  $n_0, n_1, \dots, n_i$ , all possible node

---

<sup>1</sup>Each node in cloaking graph satisfies its segment  $l$ -segment indistinguishability requirement

---

**ALGORITHM 7:** Search Star Set

---

**Input:**  $n_u$ : updated node

```
1  $NS \leftarrow n_u$ ;  
2 if ( $SS \leftarrow RequirementCheck(NS)$ )  $\neq \emptyset$  then  
3    $\lfloor$  return  $SS$ ;  
4 else  
5   forall the  $n_n \in n_u.N$  do  
6      $NS \leftarrow n_u \cup n_n$ ;  
7     if ( $SS \leftarrow RequirementCheck(NS)$ )  $\neq \emptyset$  then  
8        $\lfloor$  return  $SS$ ;  
9     else  
10       $Q_{Temp} \leftarrow Q_{Comb}$ ;  
11      forall the  $C \in Q_{Comb}$  do  
12        if  $\forall n_c \in C, n_c \in n_n.N$  then  
13           $NS \leftarrow n_u \cup n_n \cup C$ ;  
14          if ( $SS \leftarrow RequirementCheck(NS)$ )  $\neq \emptyset$  then  
15             $\lfloor$  return  $SS$ ;  
16          else  
17             $C_{new} \leftarrow C \cup n_n$ ;  
18             $Q_{Temp} \leftarrow Q_{Temp} \cup C_{new}$ ;  
19       $Q_{Comb} \leftarrow Q_{Temp}$ 
```

---

combinations for the neighbor  $n_{i+1}$ ,  $Q_{Comb}^{i+1}$ , is the set which includes  $Q_{Comb}^i$  and new combinations that can be generate with  $n_i$  and each combination  $C \in Q_{Comb}^i$  (i.e., form a clique together and satisfy segment diversity requirement).

Suppose that we are searching candidate star set for the cloaking graph node  $v_6$  updated due to the query  $q_8$  insertion in the previous step. It first checks if the  $v_6$  satisfies privacy requirement, since there is not enough query it stars evaluating neighbor vertices starting from the closest one  $v_5$ . Two nodes cannot meet user privacy nodes. Before evaluating next neighbor node  $v_{10}$  we update possible combination set which is currently empty. Node  $v_5$  is the only neighbor node combination available and added to the set. In the next step it first evaluates vertices  $v_6$  and  $v_{10}$  together and then combination  $\{v_5\}$ . Note that node set  $\{v_5, v_6, v_{10}\}$  satisfies all queries privacy requirement. Selected vertices removed from the cloaking graph and the candidate star set  $(\phi_{n_2}, \phi_{n_4}, \phi_{n_5}, \phi_{n_6}, \phi_{n_7}, \phi_{n_{10}})$  generated.

#### 4.3.8 Star-Set Pruning

Final step is the randomly pruning extra segments from candidate star set. This step plays an important role in the generation of the low cost cloaked subgraphs and the attack resilience by selecting stars randomly from outer to center of the candidate star-set. It also provides attack resilience by selecting star randomly. Note that this step is independent from the earlier stages of the STARCLOAK, so anonymization engine can process other queries while one prunes other candidate star set. It is also possible to run pruning algorithms in parallel.

Algorithm 8 sketches the pruning procedure. For each candidate star sets one first identifies boundary stars, which are the stars that has at least one neighbor star not in the candidate star set. It also identifies active stars that can not be remove from the set. In each iteration STARCLOAK selects randomly one star  $\phi_r$  from boundary star set and removes it from final star set,  $SS$ . Then it if  $SS$  still satisfies  $l$ -diversity requirement, updates boundary node and moves to the next iteration. Otherwise, it adds back  $\phi_r$  to the  $SS$  and finalize anonymization step.

To make it more clear we use an example given in Figure 4.9, where candidate star set shown in the left side of the Figure 4.9 with the black and gray circles. Black circles depicts blocked nodes that cannot be removed, because of the association with the selected queries. Suppose that maximum segment requirement is 9. First, one generates boundary star list  $\{\phi_{n_5}, \phi_{n_7}, \phi_{n_9}, \phi_{n_{13}}\}$ , gray circles that are connected with the white circles. Once this is done, one of the boundary star selected randomly, lets say  $\phi_{n_5}$  in our example. After removing  $\phi_{n_5}$ , remaining stars still meets the segment requirement. Boundary star set updated with the new star  $\phi_{n_{10}}$  and another star selected randomly from the set. Suppose that star  $\phi_{n_7}$  selected for pruning. Segment diversity requirement satisfied with the remaining stars. Note that there is no new boundary star, because  $\phi_{n_{12}}$  is an active star. One selects another star from the boundary star set,  $\{\phi_{n_5}, \phi_{n_9}, \phi_{n_{10}}, \phi_{n_{13}}\}$ . Assume that  $\phi_{n_9}$  selected to remove. Boundary star set updated with new star  $\phi_{n_{15}}$ . In the next

---

**ALGORITHM 8:** Prune Candidate Star Set
 

---

**Input:**  $Q_C$ : candidate star set queue

```

1 while true do
2   if  $Q_C \neq \emptyset$  then
3      $SS \leftarrow$  pop the first entry of  $Q_C$ ;
4      $BS \leftarrow$  find boundary stars of  $SS$ ;
5      $FS \leftarrow$  select active stars;
6      $\sigma_s^{max} \leftarrow$  find max segment requirement;
7     while true do
8        $\phi_r \leftarrow$  randomly select star from  $BS$ ;
9       if  $\phi_r \notin FS$  then
10         $SS \leftarrow SS \setminus \phi_r$ ;
11        if # of segment in  $SS > \sigma_s^{max}$  then
12          update  $BS$ ;
13        else
14           $SS \leftarrow SS \cup \phi_r$ ;
15          Output  $SS$ ;
16          break
  
```

---

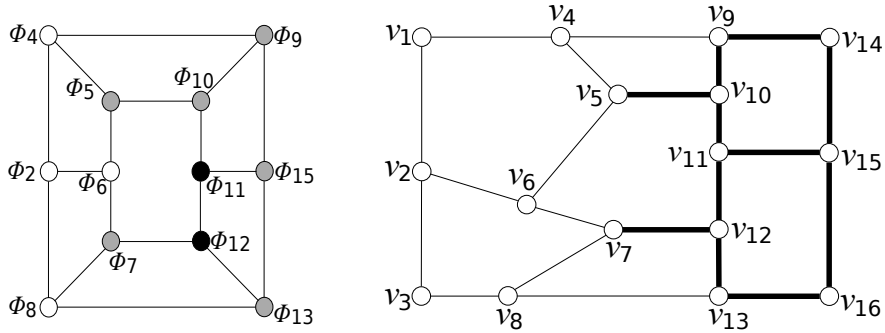


Figure 4.9: Pruning Example

iteration  $\phi_{n_{13}}$  selected and new boundary star set would be  $\phi_{n_5}, \phi_{n_{10}}, \phi_{n_{15}}$ . However, removing any of those star violates queries segment diversity requirement, thus we added back selected star to the star list. Associated segments generates cloaked sub-graph and can be used as an anonymized location for the selected queries. On the right side of the Figure 4.9, we showed generated cloaked sub-graph with bold lines.

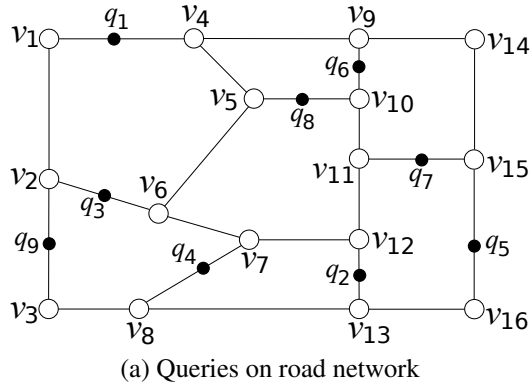
## 4.4 Optimizations

In this section, we deal with the issues in implementing STARCLOAK in the location anonymization engine (LAE), and propose multiple optimizations to improve its performance, bringing in considerable enhancement over the basic version.

### 4.4.1 Spatially Bounded StarCloak

Basic StarCloak algorithm generates cloak regions whenever it finds query set that satisfy all queries' privacy requirement. However, this approach may cause generating high cost cloak regions since it is possible to select stars which are far, yet within spatial tolerance, from each other. For instance, Figure 4.10 shows an example scenario for queries ( $q_1, q_2, \dots, q_9$ ) uniformly distributed over road-network (4.10a). Suppose that queries issued in the order of their id and all has 3  $k$ -anonymity requirement. For the sake of simplicity we did not include segment diversity requirement in this example and spatial tolerance value is high enough to cover all stars in our small road network. In Figure 4.10b we gave an example star assignments for the nine queries. When we apply basic StarCloak approach for given queries selected stars would be like black circles in Figure 4.10c, 4.10e and 4.10e. It can be noticed that selected stars spread all over the network. On the other hand, Figure 4.10f, 4.10g and 4.10h show more compact star selection possibility for the same queries with different combinations.

We propose Spatial-Bounded StarCloak algorithm to generate more compact cloaked subgraphs. The essence of this optimization is to sacrificing anonymization time to generate cloak regions with low query processing and communication costs. We define a system parameter  $\lambda \geq 1$ , called **compactness factor**, that controls the maximum hop distance between selected vertices in the candidate star set. To this end, we made some modifications on candidate star-set selection step given at Algorithm 7. First we grouped neighbors by their distance,  $d$ , to the starting node.  $\lfloor d/\lambda \rfloor$  determines the level of the each group



query	star
$q_1$	$\phi_4$
$q_2$	$\phi_{12}$
$q_3$	$\phi_6$
$q_4$	$\phi_8$
$q_5$	$\phi_{13}$
$q_6$	$\phi_9$
$q_7$	$\phi_{11}$
$q_8$	$\phi_5$
$q_9$	$\phi_8$

(b) Star Assignment

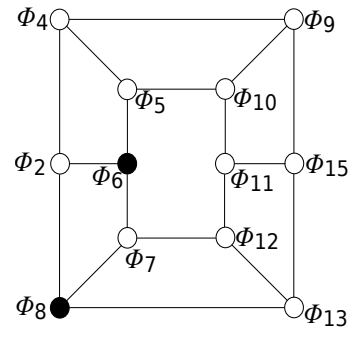
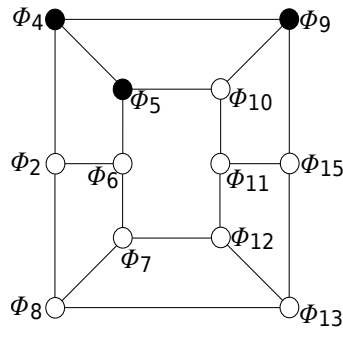
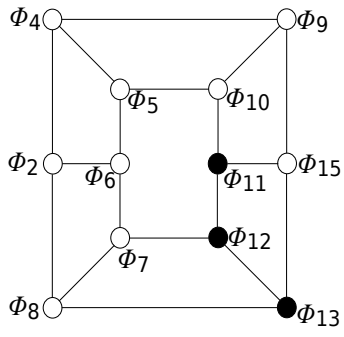
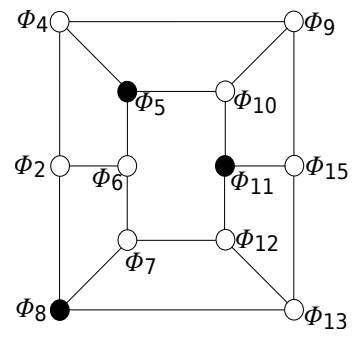
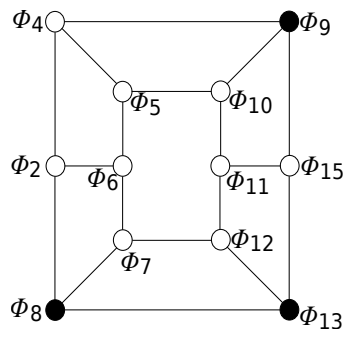
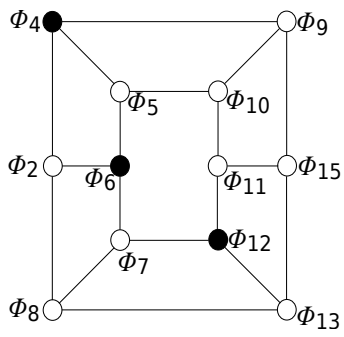


Figure 4.10: Spatially Bounded StarCloak Example

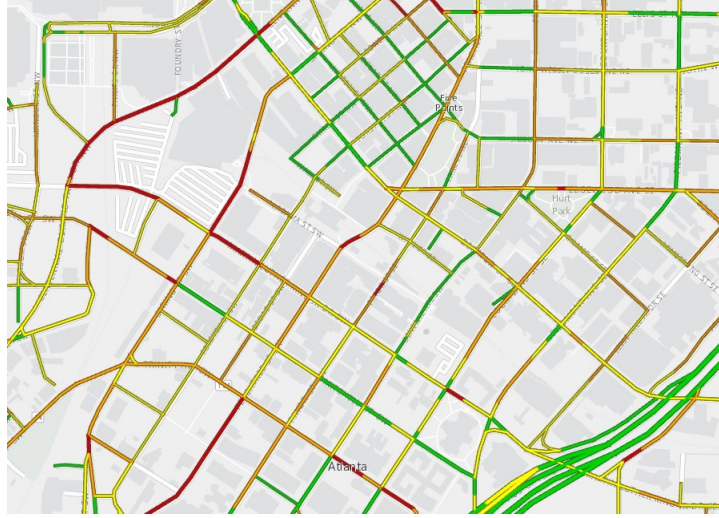


Figure 4.11: Average traffic in downtown Atlanta. The color codes red, orange, yellow and green represents relative traffic speeds with stop and go, slow, moderate and free flow, respectively.

element. At each level one only consider neighbor nodes, which can be cloak with the node combinations generated in the previous level. If any level cannot generate any node combination algorithm halts. Spatial-Bounded StarCloak enforce compactness by selecting active stars that for each star in the star set there is at least one other star which is not further then  $(2.\lambda - 1)$  hop distance.

Suppose that for the previous example we used Spatial-Bounded StarCloak with compactness factor  $\lambda = 1$ . During the process of query  $q_3$ , one first checks neighbor nodes which is 1 hop distance from the star  $\phi_6$ . Since there is no neighbor in the level 1, process halts and anonymization engine accept new query to process. However, while it is processing  $q_7$ , it first adds neighbor node associated with the star  $\phi_{12}$  which is in the 1 hop distance level. Two nodes does not satisfy  $k$ -anonymity requirement, it continue to process neighbor nodes at the second level ( $\phi_9$  and  $\phi_{13}$ ). Since we used FIFO based query processing ordering to decrease waiting time,  $\phi_{13}$  would select for next iteration. Three nodes meets all users privacy requirement and can be remove from cloaking graph.

#### 4.4.2 Hybrid StarCloak

Optimal compactness factor  $\lambda$  for the Spatially Bounded StarCloak can be determine by the request density in the area, which is highly dynamic and can change street-by-street or star-by-star. For instance Figure 4.11 shows average traffic density in downtown Atlanta. It can be notice that even neighbor segments may have different densities. It is not possible to define optimal compactness factor for each star at each time. Determined  $\lambda$  value based on the number of request per second in the predefined area, may cause low success rate for the queries issued by relatively sparse sub-areas. To overcome this problem we propose hybrid approach which leverage advantages from both Basic and Spatially Bounded StarCloak approaches. In this method first we try to generate cloak regions with Spatially Bounded StarCloak method. After that for those queries, could not cloak for a while and it is close to expire, we apply basic StarCloak algorithm. We used **consideration factor**  $\alpha$  as the system level value to decide when we apply basic StarCloak algorithm for requests. One periodically checks expiration heap if any query is close to expiration time.

When we used Spatially Bounded StarCloak with compactness factor  $\lambda = 1$  for the previous example at Figure 4.10, queries  $q_3, q_4$  and  $q_9$  would remain in the system until new queries issued in the neighborhood, since the distance between two stars  $\sigma_6$  and  $\sigma_8$  is two hop. However, if any of those query expire at this time, it would be drop even though there is a possible cloaked subgraph. With Hybrid StarCloak we able to cloak those vertices together.

### 4.5 Experimental Evaluations

We performed extensive experiments to evaluate the performance of StarCloak approach on real road-network datasets. We used evaluation metrics described in Section 4.5.2.

### 4.5.1 Experimental Setup

We used two different real road network datasets, California<sup>2</sup> and Georgia<sup>3</sup>, with varying sizes to observe the effect of query density on the efficiency and effectiveness of StarCloak approach. California road network dataset contains only highways with 21,693 edges and 21,048 nodes. 87,635 points of interest from 62 different classes (e.g. hospital, school etc.) associated with the road network. Larger road-network dataset, Georgia, contains primary and secondary roads with 430,849 edges and 428,708 nodes.

To simulate user movements we used moving object dataset generator developed by Brinkhoff<sup>4</sup>. We assign a same number (10,000) of moving objects to each map. Since the two maps are of significantly different scales, we intend to simulate both high user density (rush hour) and low user density (non-rush hour) conditions. In each simulation, we defined two classes of moving objects, with speeds corresponding to slow (e.g., trucks) and fast (e.g., passenger cars) vehicles, respectively. With a randomly assigned probability, each vehicle generates a set of (or none)  $k$ -NN queries during the simulation, with the parameters specified as follows: (1) the requested number of nearest points of interest as  $k$ ; (2) the category of the points of interest as  $c$ , e.g., church, hospital, etc.; (3) the privacy requirements as  $k$ -anonymity ( $\delta_k$ ) and  $l$ -diversity ( $\delta_l$ ); and (4) the service quality requirements as the spatial ( $\sigma_s$ ) and temporal tolerance ( $\sigma_t$ ). The values of each query are drawn independently following certain distributions, with parameters listed in Table 4.1. Note that all the parameters except  $c$  follow normal distributions;  $c$  follows a uniform distribution over the interval  $[0, 62]$ ; the values of  $\sigma_t$ ,  $\gamma$ , and  $\alpha$  are in the unit of second. After issuing a query, the vehicle waits for some normally distributed inter-wait time  $\gamma$ , until the request is either answered or dropped, before issuing another service request. Compactness  $\lambda$  and consideration  $\alpha$  factors are used only for the Spatially Bounded and Hybrid versions of STARCLOAK, respectively. All algorithms are implemented in Java

---

<sup>2</sup><http://www.cs.utah.edu/lifeifei/SpatialDataset.htm>

<sup>3</sup><https://www.census.gov/geo/maps-data/data/tiger-geodatabases.html>

<sup>4</sup><http://iapg.jade-hs.de/personen/brinkhoff/generator/>

Table 4.1: Default parameter setting for query generation.

parameters	$k$	$c$	$\delta_k$	$\delta_l$	$\sigma_s$	$\sigma_t$	$\gamma$	$\lambda$	$\alpha$
mean	5	$N/A$	5	5	4	10	20	1	2
deviation	1	$N/A$	1.5	1.5	1	2	2	0	0

and tested on a Windows 7 platform with Intel(R) Core(TM) CPU (4.00 GHz) and 16GB memory.

#### 4.5.2 Evaluation Metrics

To evaluate the performance of the proposed location privacy preserving mechanisms, we used following metrics: success rate, successful throughput, anonymization time, query processing time, candidate list size and entropy.

**Success Rate:** One of the most important measure for evaluating the effectiveness of the anonymization algorithm. The main objective of cloaking technique is to anonymize maximum number of service requests. Success rate can be define as the fraction of successfully anonymized queries with respect to number of total queries issued by mobile users.

**Anonymization Time:** Since anonymization process increases the response time, it affects users experience while they are using location-based services. Average anonymization time metric measures the run-time performance of the cloaking algorithms in terms of average time elapsed in the anonymization engine for each successfully anonymized query. Note that we use only successfully anonymized queries for a fair comparison to eliminate the effect of the success rate performance of the algorithm.

**Query Processing Time:** In order to respond anonymous location-based queries, LBS provider needs to evaluate over cloaked sub-graph instead of users' exact location. This metric measures the processing cost at the LBS server site. Number of border nodes and edges of generated cloak region effects processing time as described in Section 4.2.4.

**Candidate Set Size:** Measures the communication cost between the anonymization engine and the LBS w.r.t size of the candidate result set. More compact sub-graphs lead

less communication cost.

**Successful Throughput:** To evaluate scalability of the anonymization algorithms, we used successful throughput as a metric which considers responsiveness of the model and the execution time of anonymizing a query [45]. We calculate successful throughput as;

$$\text{query execution rate} \times \text{anonymization success rate}$$

**Entropy:** We used **entropy** as a quantitative measure of uncertainty achieved by cloaking approaches. Segment entropy calculated by

$$H(S) = - \sum_i^{|S|} p_i \cdot \log_2 p_i$$

where  $p_i$  denotes the probability of the user identity being associate with segment  $s_i \in S$ . Note that, the number of segments in the generated cloaked subgraph may vary for each anonymization algorithm based on it's segment selection strategy. Using simple entropy for different sized cloaked region may mislead to capture actual strength of the algorithm. For this reason, we used **randomized entropy**,  $d$ , as a privacy metric in our road network based anonymization model[53]:

$$d = \frac{H(S)}{\log_2 |S|}$$

### 4.5.3 Experimental Results

#### *Results on Query Processing Time*

We now take a close examination of the impact of the location anonymization operation over the service performance. First we evaluate anonymized query processing cost. Specifically, in terms of the query execution cost, we measure the average execution time of processing a query at the server side. Note that each algorithm has different success rate

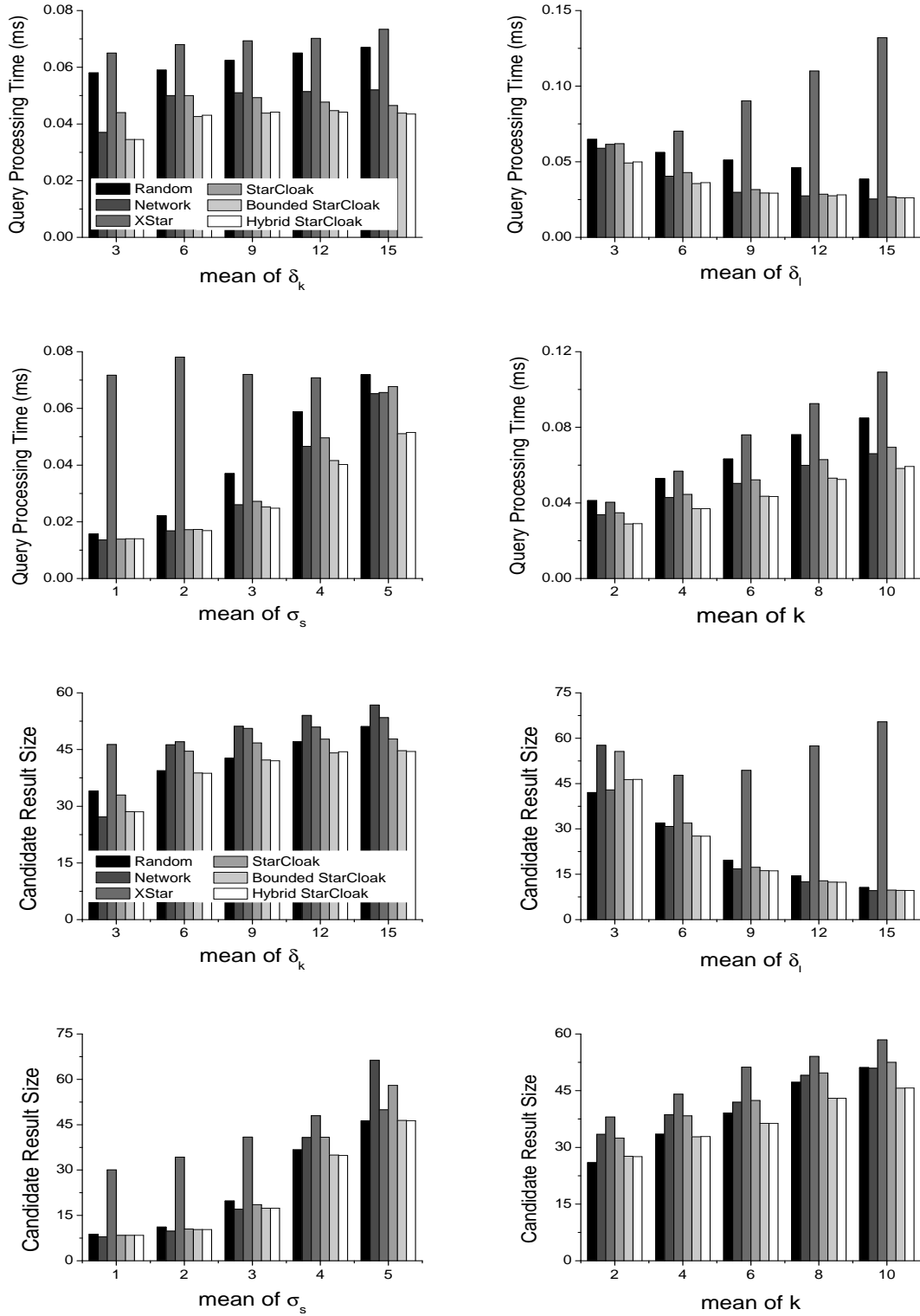


Figure 4.12: Query execution cost (in terms of the average query processing time per query) and communication cost (in terms of the average size fo candidate result per query) with respect to varying settings of parameters  $\delta_k$ ,  $\delta_s$ ,  $\sigma_s$  and  $k$ .

performance and considering all anonymized locations to get average anonymization time is not fair comparison. For example Hybrid StarCloak generates mostly low cost cloaked sub-graphs, but sometimes rather than dropping query, it generates very high cost cloaked sub-graphs. Including these cost does not reflect actual performance of the algorithm. For this reason we select the same number of anonymized locations,  $N$ , generated by each algorithm.  $N$  is the number of cloaked region generated by the algorithm with minimum success rate.

Figure 4.12 shows that the StarCloak approaches generates lower costs in most cases with respect to query processing time. As expected among three StarCloak approaches, basic one generates high cost anonymized locations, since unlike others it select stars without considering their closeness.

#### *Results on Candidate Result Size*

In this set of experiment we evaluate anonymization algorithms over their communication cost performance. We measure the average number of object return in the candidate result of a query to reflect communication cost. Similarly we consider same number of anonymized location for each algorithm. Figure 4.12 illustrates that our StarCloak approaches generates more compact cloaked subgraphs than the XStar in most cases.

#### *Results on Success Rate*

Our experimental results show that StarCloak approaches has high success rates because of the ability to handling different user requirements and finding possible cloaked regions immediately. Figure 4.13 and 4.14 plot the percentage of the incoming queries that successfully anonymized with varying privacy and service requirements for California and Georgia maps respectively. Success rate for Hybrid StarCloak is between for the naive StarCloak and the Spatially Bounded StarCloak. It is expected because it generates cloak regions for queries in the sparse area where Spatially Bounded StarCloak cannot generate, however

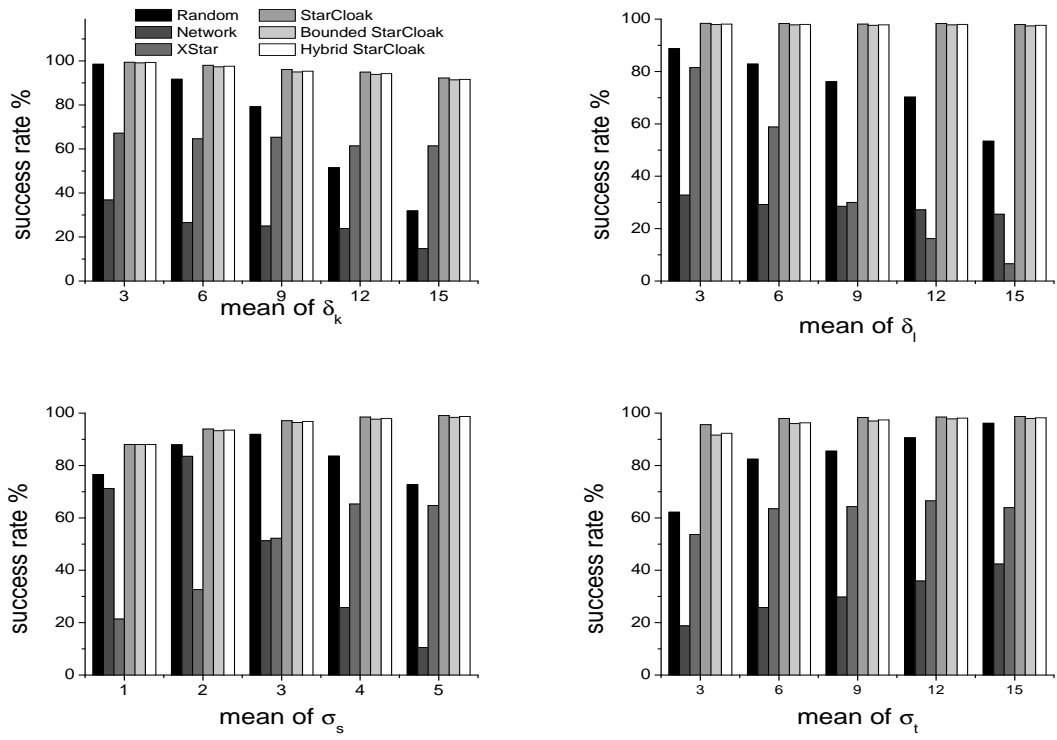


Figure 4.13: Success rate for California Map

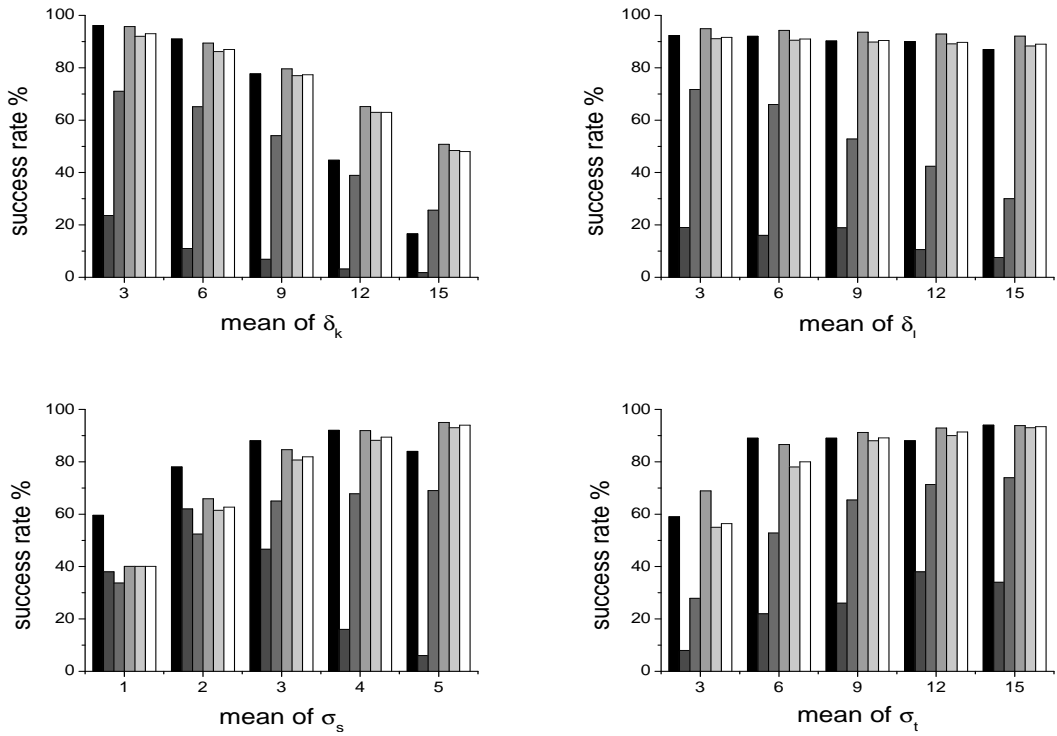


Figure 4.14: Success rate for Georgia Map

it is not the same with the basic StarCloak since in the second evaluation those queries does not have enough neighbor queries that they can cloak together as in the case of SC. Meanwhile, while the number of  $\delta_k$  requirement increases the effect on success rate for the Hybrid StarCloak decrease. The reason is, since the queries wait in the system longer the chance of finding neighbor node with Spatially Bounded StarCloak increase.

As shown in the figures, increasing privacy requirement decrease the success rate for both anonymization approaches as expected, but the effects are different for each maps and privacy requirements. While  $\delta_k$  requirement is increasing, in Georgia map success rate decreases more than in California map. This can be explained by the density of queries. For the same number of users in California map it is possible to find queries in the neighborhood, however in the sparse map like Georgia, the chance of finding enough query in the same spatial tolerance is lower. On the other hand, for the XStar increasing  $\delta_l$  requirement effects success rate in California map more than the Georgia unlike the StarCloak approaches. Interestingly, query density is also same reason for this case. The XStar requires to anonymize queries on the same star together. This means that cloaked subgraph needs to include the number of segment that meets maximum  $\delta_l$  requirement among the queries in the star within the minimum  $\sigma_s$  spatial tolerance value, which may not be possible. However, in StarCloak approach if there is a conflict between two queries based on their segment diversity and spatial tolerance values they group on different cloaking network vertices.

Third and forth of the Figure 4.13 and 4.14 display the impact of changing service requirements on success rate. As expected increasing spatial and temporal tolerances increase the chance of the successful anonymization for the queries. Clearly we can see that, low spatial tolerance values effects XStar more than StarCloak approaches due to the lack of handling query conflict problem, especially when the area dense like California. It can be notice that, when the  $\sigma_s = 1$  all StarCloak approaches generate cloaked subgraphs with the same success rate. It is expected because if the compactness metric ( $\lambda = 1$ ) is

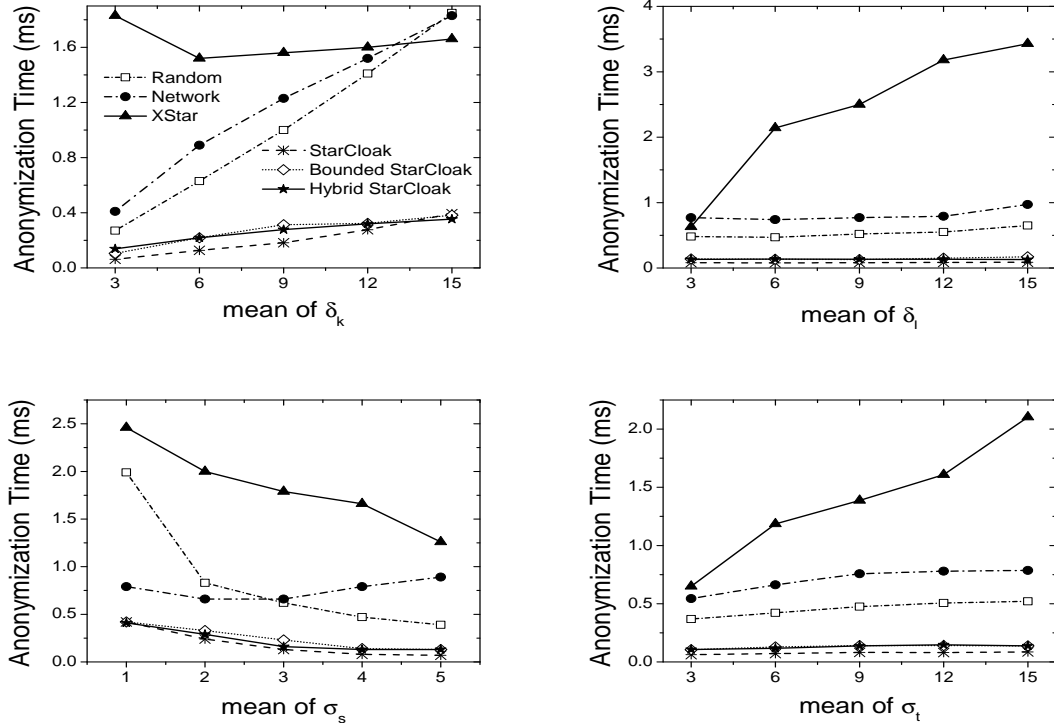


Figure 4.15: Anonymization time for California Map

equal the spatial tolerance value all the StarCloak approaches process in the same manner.

### Results on Anonymization Time

In this set of experiments we measure the anonymization performance with respect to time elapsed from the query issued time to successfully anonymized time. Not that we did not include dropped queries to eliminate the effect of algorithm success rate. The results shows that StarCloak approaches produce significantly better anonymization time with varying privacy and service requirements. SC generates cloak regions whenever there is a possible set of queries that meets all users privacy and service requirements. This leads to lowest average anonymization time. On the other hand, all other approaches waits other queries to be in the neighborhood. However, even though we used  $\lambda = 1$  as a compactness metric anonymization time for the StarCloak optimizations are better than XStar. Besides the ability to handling query conflicts, searching process that StarCloak approaches consider is

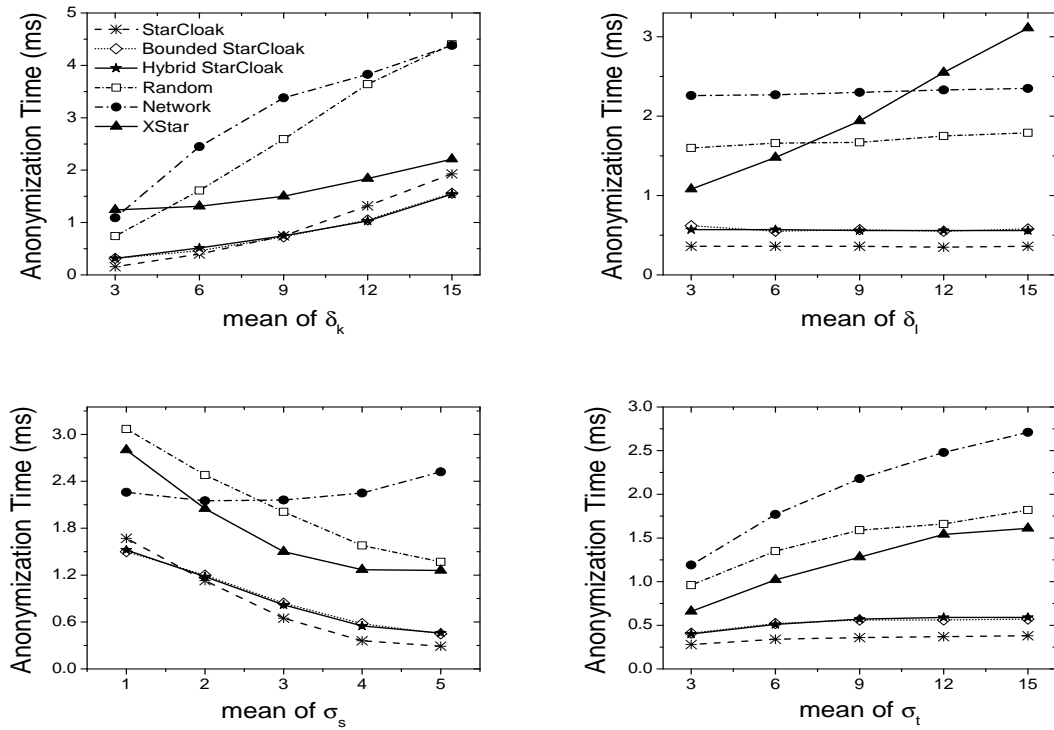


Figure 4.16: Anonymization time for Georgia Map

also effects anonymization time. In the XStar during the super-star construction step one randomly selects additional stars until it satisfies requirements. However, this selection is strict and there is no star removing option. On the other hand in StarCloak it checks all the possible star combinations to generate super-set, which leads to less anonymization time. Hybrid StarCloak performs worst with respect to anonymization time in our StarCloak approaches yet there is not much difference between Spatially Bounded version. This is expected, because queries that anonymized with Hybrid StarCloak but not with Spatially Bounded StarCloak, are close to their expiration time. Which means that they are in the system for a while and increase the average anonymization time.

#### *Results on Successful Throughput*

This set of experiment shows that STARCLOAK approaches produces high and relatively stable successful throughputs with the various service requirement parameters. As Figure

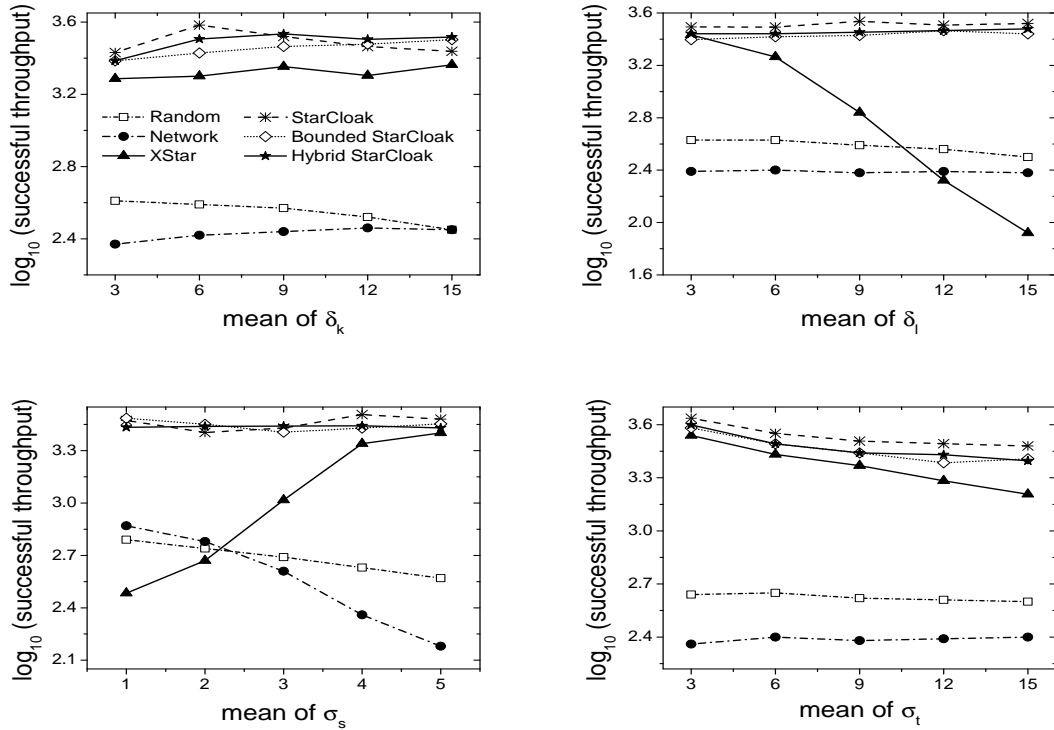


Figure 4.17: Successful throughput for California Map

4.17 and 4.20 show that baseline algorithms are not applicable w.r.t their scalability performance. Because, they are not carefully design to quickly identify possible cloaked regions and each time search from the segments in the spatial tolerance.

STARCLOAK approaches perform better than XStar based on the successful throughput metrics. Searching process in XStar is faster, since it search on only neighbor active stars. However, the number of unsuccessful attempts is high, which decrease the successful throughput. On the other hand, with efficient index mechanism STARCLOAK able to cloak more user in less time.

### Attack Resilience

We considered replay attack model described in Section 4.2.5 to evaluate the anonymization algorithms strength w.r.t. attack resilience. Figure 4.19 and 4.20 show the normalized entropy results for the various parameters. As expected while Random Expansion model

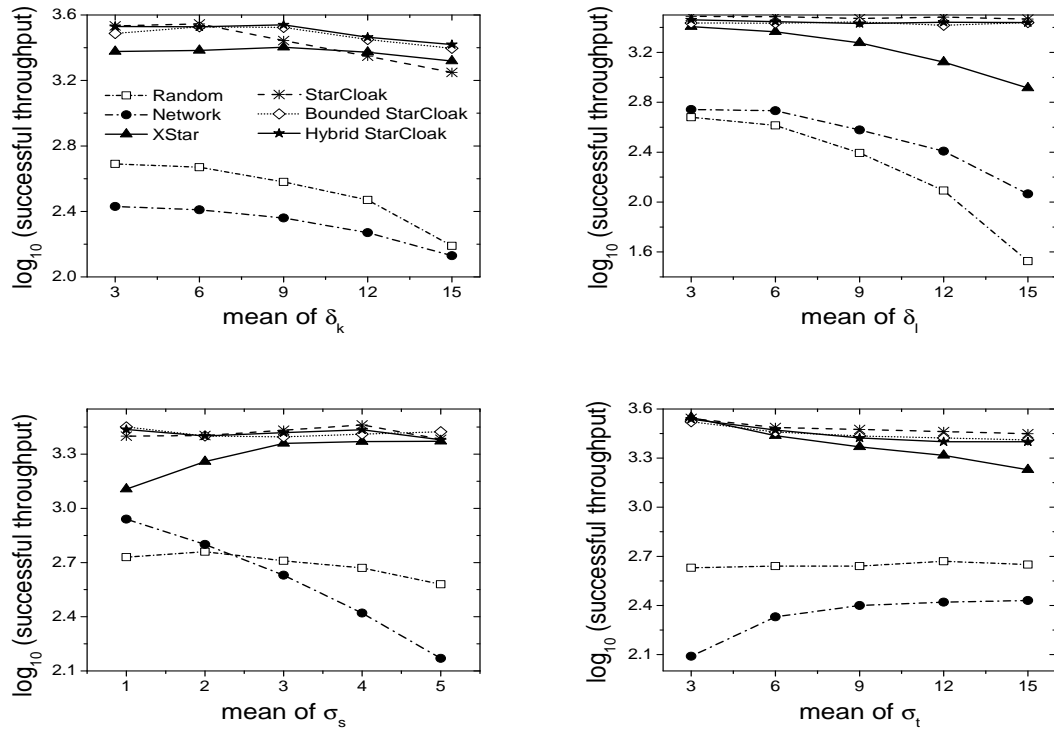


Figure 4.18: Successful throughput for Georgia Map

shows high attack resilience, Network Expansion is the weakest anonymization model. Note that, even though Random Expansion uses completely random selection, it still not reach the optimal entropy value which is 1 for normalized entropy. The reason is the using spatial tolerance constraint to generate cloaked regions, since the segments in the center of the cloak regions tend to be high likelihood values. Experiments shows that STARCLOAK approaches shows strong attack resilience, where in most cases it is close to Random Selection model and higher than the XStar model.

As we argued in the Section 4.2.5, first graph shows that, if the same cloak region generated for more user identifying starting segment getting harder. Note that it is not possible to capture this whit General Replay Attack model which does not consider other queries in the cloaked region. However, with Correlation-based Replay Attack model we showed the effect of the  $\delta_k$  value on the attack resilience. Network Expansion is the exception in this case, since it needs to expand more to satisfy k-anonymity requirement and the chance to

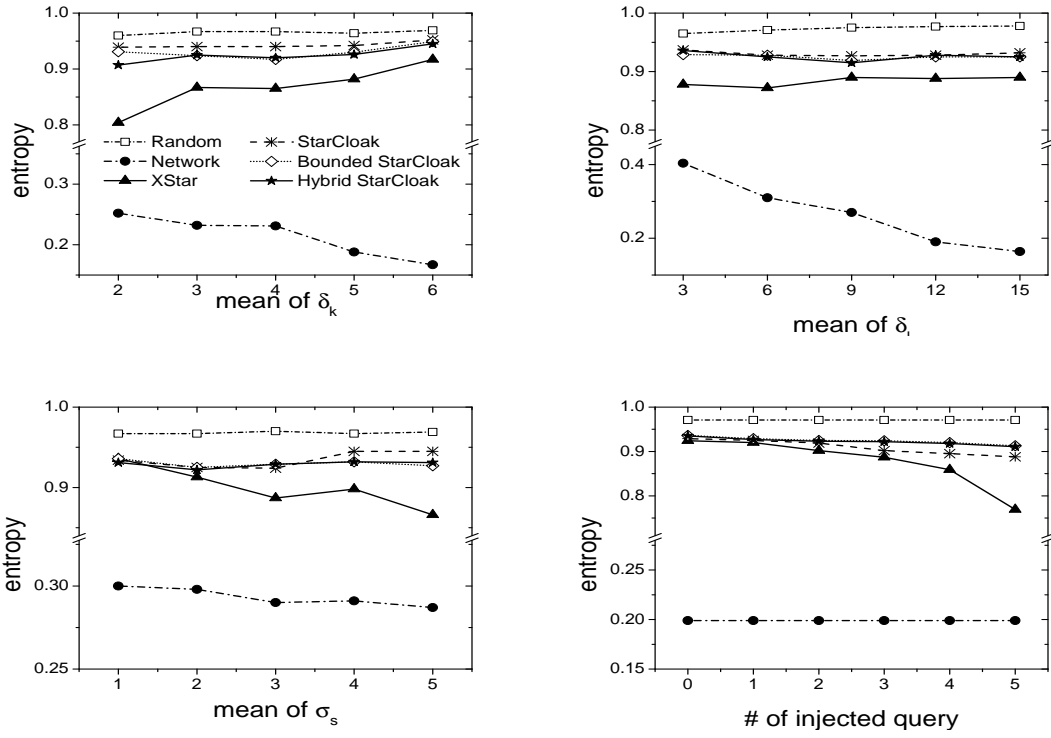


Figure 4.19: Average entropy for California Map

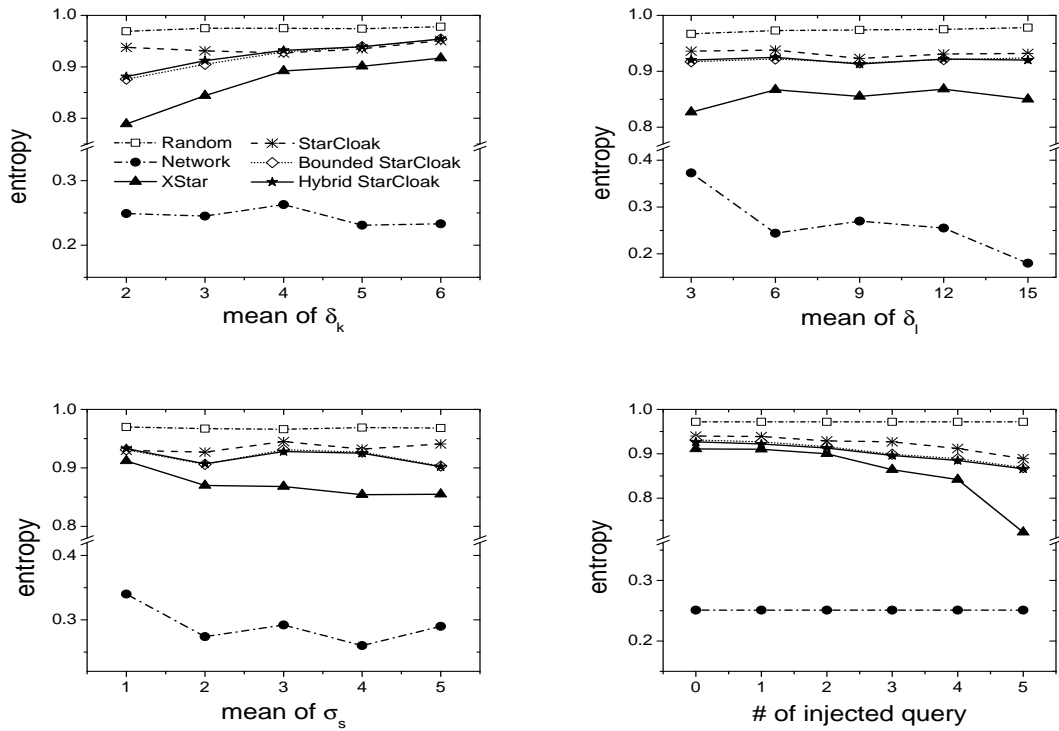


Figure 4.20: Average entropy for Georgia Map

pinpoint starting segment gets higher.

Final set of experiments show that XStar is vulnerable against our newly defined attack model, Injection-based Attack. Random and Network Expansion models are not based on the other queries and entropy value for those models remain still. As the number of queries in the cloak region known by attacker increases XStar perform worse attack resilience than the STARCLAOK approaches. Note that in this experiments we used the same user movement simulation data. In this case known queries can be spread over the cloak region, which decrease the attack success. However, smart attacker may inject queries to the system as the same segment, which increase the chance of the identify possible star in XStar model.

## CHAPTER 5

### PRIVACYZONE

In this chapter we address limitations of the current spatial-temporal cloaking techniques and present a first spatial trigger based system to establish personalized privacy quarantine areas in the physical or cyber space to enable objects and people to maintain spatial and temporal privacy at selected area of interests at the selected time interval. We introduce a novel concept of PrivacyZone and a suit of algorithms for constructing personalized privacy-zones and enforcing privacy-zone protection for mobile users or moving objects on the roads. Our PrivacyZone system has four innovative properties. First, we provide efficient mechanisms to enable mobile users or owners of moving objects to specify their desired privacy-zones with both spatial and temporal customization. Second, we develop a suite of privacy-zone construction algorithms that can effectively install personalized privacy-zones on road networks or travel paths in both outdoor and indoor environments. Third, we devise an energy efficient privacy-zone enforcement algorithms that provide real-time spatial and temporal alerts to ensure no location exposure for users who enter their personalized privacy-zones. Forth, we designed a novel mobile permission system based on PrivacyZone, which provide mobile users more flexibility to enforce their privacy permissions, while minimizing their efforts. Our experimental results demonstrate the utility of PrivacyZone in real work environment.

#### 5.1 Introduction

Recently, several spatial cloaking algorithms have been proposed in order to protect location-based service users' privacy [7, 8, 54, 45]. The main idea of the spatial cloaking is to replace user exact location with a spatio-temporal region that satisfies certain privacy requirements such as  $k$ -anonymity and  $l$ -indistinguishability. However, conventional cloak

region-based location privacy techniques have limited applicability in current mobile environment. Most importantly existing location-based applications in the market (e.g., Apple App store, Google play) are designed to accept focal location as an exact GPS coordinate instead of a region. Another limitation is that most of the spatial cloaking algorithms are based on snapshot queries (e.g., where is the closest gas station?) and new random pseudo-ID used for each location update to eliminate possible linkage attack between two consecutive cloaked region. However, most of the popular location-based applications require continuous location monitoring in order to provide useful services such as step-counter applications (e.g., Runkeeper).

In addition to applicability limitations the quality of service is the another limitation of the current spatial cloaking techniques. Generally mobile users concern is to be associate with certain location. For example, while going to hospital may be a sensitive information for someone, he may not care sharing his location when he is in the restaurant. However, generating cloak region for each location update cause unnecessary service utility degradation. Also, spatial cloaking systems require a trusted middle ware servers in order to provide anonymity which is susceptible to a single-point failure.

In this work, we focus on developing a privacy protection mechanism that can apply directly on current smart phone applications as part of the mobile permission systems. Modern smart phone platforms such as Android or iOS provide numerous apps that facilitate the user's daily life like Google Maps <sup>1</sup>. Most of the applications relies on to access several personal information about the user in order to provide services (e.g., GPS, camera). However, many mobile applications today aggressively collect much more personal context data than what is needed to provide their functionalities. In current mobile platforms protecting personal information rely on permission system that allow user to control (allow or deny) third-party apps' access to certain personal information. Unfortunately, current mobile permission systems have several limitations. First of all, current permission system

---

<sup>1</sup><https://www.google.com/maps>

is based on "take it or leave it" approach. To avoid the risks, a user can decide not to install application or not to release any context information to them but then the user might not be able to enjoy the utility provided by these applications. If a user allows an application, the user must blindly trust that application will properly handle their private data. [55] shows that a better trade-off can be achieved by providing users with additional decision types, where sensitive information is only partially revealed to apps in exchange for some utility.

Another limitation of the current permission system is related to the static nature of the access decisions. The assumption is the user permission decisions are not frequently change. However, it is possible for a user to allow apps to access his location while the user in the park but would like to deny when he is in the hospital. One basic option for this problem is to ask user for each context request issued from the apps [56]. However, this approach requires a significant effort from the user since a single app can make tens to hundreds of sensitive requests every day [57, 58]. Hence, to support context-aware permission policies, advanced mechanisms are needed to help users with the overhead introduced by runtime permissions.

Recent years several research project propose new permission mechanisms [59]. However, proposed solutions are follows one-size-fits-all paradigm. We believe that each context needs to handle separately since each of them required specific privacy protection mechanism. For instance, location permission should be considered separately from the microphone permissions. Particularly, in this work we focus on efficient location permission mechanism. Most of the mobile application request access to users location data, because to advertisers, location is one of the most valuable information. However, location access has introduced a new class of privacy threats. These threats range from an adversary's ability to localize an individual to profiling and identifying him based on the places he visits. For example, personal medical condition by knowing users frequent visits to specific clinics.

To overcome these limitations we propose new location privacy protection system called

PrivacyZone. The main idea is to block location disclose only in the personalized privacy quarantine areas, otherwise share obfuscated position such as randomly selected location in the privacy zone in order to continue application usability. In this system user first define his personal private zones and required privacy level. Private zones can be selected either by choosing a location on the map or providing semantic location (e.g., health care) to select all related locations in the area. Privacy level can be define as a spatial or time distance to reach a selected location with considering road network constraint. This system provides users more flexibility to enforce their privacy permissions, while minimizing their efforts.

Generating privacy zones and hiding location when user in the quarantine areas is only part of the story. Supporting privacy zones as a service introduce several technical challenges. Negligent management of privacy zone processing can lead to excessive energy consumption of mobile devices, especially when the location-based application relies on continuous tracking of mobile devices. We argue in this work that intelligent technique can be developed for scalable processing of privacy zones by minimizing the amount of continuous monitoring of mobile objects locations. Furthermore, the performance of privacy zone processing can be affected by a number of factors, such as *frequency of wake-ups* - how often mobile devices should wake up because of possible privacy zone hits and *frequency of privacy zone checks* - how many privacy zones should be evaluated at each wake-ups. Since frequent and possible unnecessary wake-ups and privacy zone checks not only reduce battery life of mobile devices considerably but also increase the loads of as privacy zone processing server, we need efficient privacy zone processing that can reduce the number of unnecessary wake-ups and privacy zone checks at each wake-ups. Finally, the privacy zone processing system should scale to a large number of privacy zones and mobile objects, while meeting the low latency goal.

Our main contributions are as follows:

- We provide efficient mechanisms to enable mobile users or owners of moving objects to specify their desired privacy-zones with both spatial and temporal customization.

- We develop a suite of privacy-zone construction algorithms that can effectively install personalized privacy-zones on road networks or travel paths in both outdoor and indoor environments.
- We device an energy efficient privacy-zone monitoring algorithms that provide real-time spatial and temporal checks to ensure no location exposure for users who enter their personalized privacy-zones.
- We designed a novel mobile permission system based on PrivacyZone, which provide mobile users more flexibility to enforce their privacy permissions, while minimizing their efforts.

Our experimental results demonstrate the utility of PrivacyZone in real work environment.

## 5.2 System Overview

PrivacyZone enables users to consider context information and make fine-grained location privacy decisions. This provide better trade-off between privacy and utility. In this section, we describe the concept and the architecture of the PrivacyZone as a new privacy protection mechanism, road network model, and user defined privacy zone model.

### 5.2.1 Concept

In this work we focus on the applications that requires access to users' location information in order to provide their services. For example, Groupon<sup>2</sup> shows user about deals in their vicinity, or Yelp<sup>3</sup> gives user information about nearby business. Such applications can run in the foreground or background. Even though this application provide useful information, mobile user may not want to share his exact location with these application all the time.

---

<sup>2</sup><https://www.groupon.com/>

<sup>3</sup><https://www.yelp.com/>

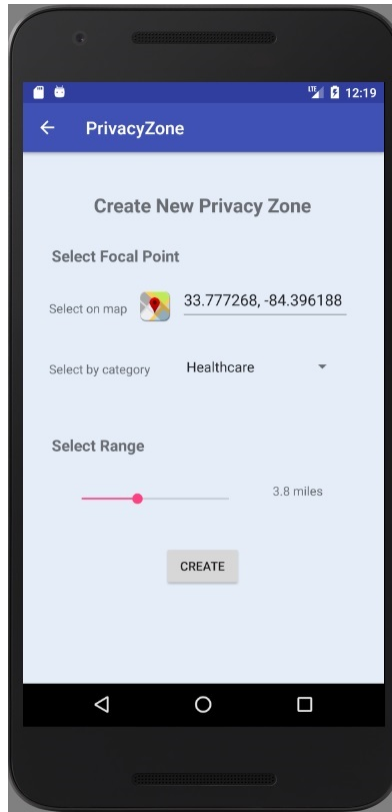


Figure 5.1: Demo: Create Privacy Zone

In PrivacyZone, mobile users can define the privacy preferences with spatial and temporal customization. When entering his/her privacy zone, the mobile users exact location service requests will be on-hold and will only be released until the mobile user moves outside of his/her privacy zone. The spatial and temporal delay imposed by the privacy zone is customizable based on mobile users specific interests. When application running on the background, to prevent application crashes instead of holding the location request, we provide random location in the current privacy zone. This approach is especially useful for the applications that providing exact location is not critical such as weather applications. However, if the application running on the foreground, decision about providing exact or fake location will be made by the user only if he/she is in the privacy zone. This approach provide user high flexible privacy and utility control over the application with minimal permission management effort.

Figure 5.1 shows a screen of Privacy Zone system that user define their privacy zones.

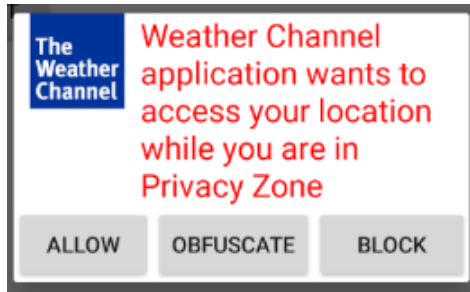


Figure 5.2: User alert example

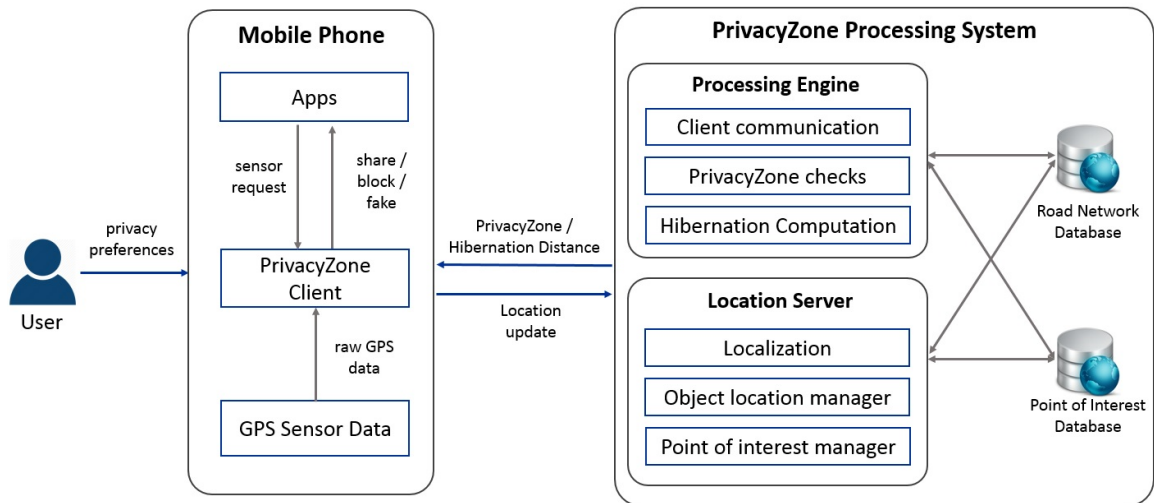


Figure 5.3: System architecture

Figure 5.2 shows an example dialog that PrivacyZone alerts user if the application wants to access user location while the user is in one of his privacy zones. This dialog shows only if the application running in foreground, otherwise it will apply predefined rule either block or obfuscate.

### 5.2.2 Architecture

In the design of the PrivacyZone system we consider the following goals: *Efficiency* in the resource-constrained mobile devices; *Privacy guarantee* with not allowing adversaries to access raw location data while user in the privacy zone; *Scalability* in the PrivacyZone server to handle large number of user; *Flexibility*, to enable users to configure their privacy preferences for each application.

In our model untrusted applications access user location data through PrivacyZone client and do not have access to raw sensor data. PrivacyZone Processing System with the available road network information is responsible to generate privacy zones with respect to users' privacy preferences. PrivacyZone client and server collaborate to decide whether to allow exact location, suppress the request, or send fake location in the privacy zone. In order to provide efficient system it is important to orchestrate PrivacyZone server and client carefully. In addition, it is reported that data communication via SSL is still vulnerable to the *Man-in-the-Middle* attack and no convincing solution is ready yet. In order to provide privacy guarantee it is important to design orchestration that not require exact location to send to the PrivacyZone server when the user is in the privacy zone. In Section 5.3 we present several approaches, that each focus on different design decisions.

### 5.2.3 Road Network Model

We model the road network as a directed graph  $G = (V, E)$ . The node set  $V = \{n_0, n_1, \dots, n_{|V|}\}$  represents road junctions and the edge set  $E = \{n_i n_j | n_i, n_j \in V\}$  represents road segments. We refer to an edge  $n_i n_j$  as a segments that connects two road junction node  $n_i$  and  $n_j$ . The listing order of the two end nodes of the edge  $n_i n_j$  represents the direction of the road segments. In other words, while the edge  $n_i n_j$  represents one direction starting from  $n_i$  to  $n_j$ ,  $n_j n_i$  represents the opposite direction. For each edge we maintain road segment related information such as segment length (e.g., 0.8 miles) and speed limit (e.g., 55 miles/hr). We used  $length(n_i n_j)$  and  $speed\_limit(n_i n_j)$  to represent length and speed limit of the road segment  $n_i n_j$ , respectively. Speed limit is used to identify travel time distance of the road segment  $travel\_time(n_i n_j)$  which calculated as  $\frac{length(n_i n_j)}{speed\_limit(n_i n_j)}$ .

We denote a *road network location* by  $L = (n_i n_j, p)$  which includes two elements; road segment,  $n_i n_j$  and the progress  $p$  along the segment (i.e., distance from  $n_i$ ). Note that, road junction  $n_i$  as a road-network location can be also represented with this notation

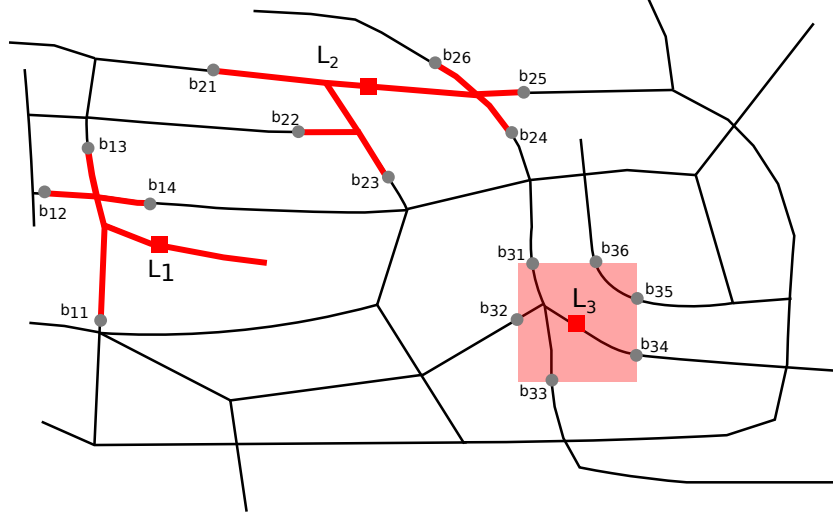


Figure 5.4: Star and Rectangular Shape Privacy Zones

by the segment that starts from the node and the zero progress (i.e.,  $L(n_i n_j, 0)$ ). The road network distance between two locations  $L_1 = (n_i n_j, p_1)$  and  $L_2 = (n_k n_l, p_2)$  is the total distance of the shortest path between the two positions  $L_1$  and  $L_2$ . We used two type of road network distance; length-based and travel time-based. Formally we define length-based road network distance as follows:

$$d_l(L_1, L_2) = \text{length}(n_i) - p_1 + p_2 + \sum_{s \in SP(n_j, n_k)} \text{length}(s)$$

in which  $SP(n_j, n_k)$  denote the set of segments that forms a path from  $n_j$  to  $n_k$  with the minimum total distance value. Similarly, we define travel time-based road network distance as follows:

$$d_{tt}(L_1, L_2) = \frac{\text{length}(n_i n_j) - p_1}{\text{speed\_limit}(n_i n_j)} + \frac{p_2}{\text{speed\_limit}(n_k n_l)} + \sum_{s \in SP(n_j, n_k)} \text{travel\_time}(s)$$

#### 5.2.4 Privacy Zones

Privacy zone is a sensitive location defined by user. In our work we consider underlying road network and define privacy zone as a star-shaped subgraph centered at the focal location denoted as  $PZ(L_f, r)$  in which  $L_f$  is the road network location of the focal point and  $r$  is the size of the privacy zone, represented by a range from the focal location. Range can be define with both travel time-base (e.g., 5 minutes distance to Lenox Mall) or length-based (e.g., 2 miles distance to Klaus Building). Figure 5.4 shows three star shaped privacy zone examples with focal locations  $L_1$ ,  $L_2$ , and  $L_3$ . The set of segments inside the privacy zone  $PZ$  denoted as  $S_{PZ}$ . We called those points on the road-network which bound a privacy zone *border points*. For example,  $b_{24}$  is one of the seven border points of the privacy zone with the focal location  $L_2$ .

In addition to star-shaped privacy zones, it is also possible to define rectangle shape privacy zones as shown in the Figure 5.4. In this case user define the privacy zone as  $PZ(L_f, r_x, r_y)$  in which  $r_x$  and  $r_y$  are the spatial dimension of the PrivacyZone. However, this approach may require to block some location, even if the user is not close to focal location with respect to underlying road network. In Section 4.5 we evaluated the performance of these two privacy zone shapes through simulated experiments.

User may also define privacy zones as batch by selecting the semantic of the focal locations. One considers all locations with the specified type as a focal object and generates respected privacy zone for each focal object. For example, Figure 5.5 shows privacy zones for all religious buildings as a focal object with 2 minutes travel distance.

#### 5.2.5 Privacy Model

The goal of the PrivacyZone is to assure to hide mobile travelers exact location while the user in one of his privacy zones. Intuitively, from the adversary perspective, the difficulty of tracking a user is in proportion to the number of segments that she is possibly associated with. We use *segment indistinguishably* as a resilience measure of the privacy protection



Figure 5.5: Privacy Zones for all religious buildings

mechanism against location spoofing attacks.

**Definition 11** *Privacy zone  $PZ$  for a user  $u$  is  $l$ -segment indistinguishable, if  $PZ$  contains at least  $l$  road segments such that the association of any of the  $l$  segments with user  $u$  is indistinguishable from the other  $l - 1$  segments.*

Given a privacy zone  $PZ$  with a set of segments  $S_{PZ}$ , the ideal protection is achieved if each segment is indistinguishable to the adversary: the mobile object is associated with each segment in  $S_{PZ}$  with equal probability  $1/|S_{PZ}|$ . However, with effective attacks (see Section 5.2.6), the adversary can identify that the association between  $u$  and a specific segment  $s \in S_{PZ}$  has higher probability than  $1/S_{PZ}$ . In order to capture such vulnerability we use entropy as a quantitative measure of uncertainty achieved by PrivacyZone.

$$H(PZ) = - \sum_{s \in S_{PZ}} p_s \cdot \log_2 p_s$$

in which  $p_s$  denotes the probability that an adversary can infer user's association with segment  $s$  in privacy zone  $PZ$ . The Entropy is a measure of the amount of information require to break the indistinguishability provided by the system. Adversary apply one of

the attack model to calculate association probabilities of each segment in the privacy zone by estimating the likelihood of user to pass each segment  $like[s]$ . Under this model, the association probability is calculated as;

$$p_{s_i} = \frac{like[s_i]}{\sum_{s \in S_{PZ}} like[s]}$$

### 5.2.6 Attack Models

In this section we provide attack models that adversary may apply in order to find likelihoods of each segment in the privacy zone based on his prior knowledge. Background knowledge considered in this work includes (1) the underlying road network, (2) all users location recordings except while they are in the privacy zone, and (3) privacy zones defined by each user. Note that with sufficient observation, attacker can identify user privacy profile. For example, from the adversary perspective if the user's location always block when the user is nearby the hospitals attacker can infer that the user specified health-care buildings as private. Also, by observing enter and exit point he can identify the range of the privacy zone. In this attack models we consider the worst case scenario where the attacker have access users' locations while they are not in the privacy zone with high frequency that they can identify each consecutive segments. We below describe three types of attacks: (1) Timing Attack, (2) Minimum Travel Time-based Attack, and (3) Markov Model-based Transition Attack.

#### *Timing Attack*

In timing attack, for the privacy zone  $PZ$ , the attacker observes the user time of entry  $t_{in}$  and the time of exit  $t_{out}$  from border nodes  $b_{in}$  and  $b_{out}$ , respectively. He assigns one to likelihood of the segments that are reachable in the time period between times of entry and exit to the privacy zone (i.e.,  $t_{out} - t_{in}$ ). If the segment is not reachable, than he assigns 0 to the segments likelihood:



Figure 5.6: Minimum Travel Time-based Attack

$$like[s_i] = \begin{cases} 1, & \text{if } t_{out} - t_{in} \geq d_{tt}(b_{in}, s_i, b_{out}) \\ 0, & \text{otherwise} \end{cases}$$

in which  $d_{tt}(b_{in}, s_i, b_{out})$  defines the minimum time required for the mobile object to travel from the border node  $b_{in}$  to the border node  $b_{out}$  with the path that includes segment  $s_i$ . This time is the sum of the travel time distance from  $b_{in}$  to start node of the segment, segment travel time, and travel time distance from segment end node to the  $b_{out}$ . For two way segments we also need to consider shortest paths that visit segments end node first. Formally;

$$d_{tt}(b_{in}, s_i, b_{out}) = \min( \\ d_{tt}(L_{b_{in}}, L(s_i, 0)) + d_{tt}(L(s_i, length(s_i)), L_{b_{out}}), \\ d_{tt}(L_{b_{in}}, L(s_i, length(s_i))) + d_{tt}(L(s_i, 0), L_{b_{out}})) \\ + \frac{length(s_i)}{speed\_limit(s_i)}$$

#### *Minimum Travel Time-based Attack*

In this attack model, the attacker assumes that mobile users prefers to use the path with minimum travel time. Based on this assumption and background knowledge of users trajectory before and after he enters the privacy zone, attacker identify segments weather there

is an alternative path that may lead to a lower travel time in order to reach certain segments. Likelihood assigned for each segment based on the similarity of the travel time. Note that in this attack model we used travel time similarity instead of path similarity, because in urban setting it is possible to have multiple routes that leads to have similar travel times (e.g., areas like Manhattan, NY). Figure 5.6 shows an example scenario. Assume that blue line indicates user trajectory that enters to a privacy zone from north and exits from the south. It is clear that the likelihood of user to visit east side of the privacy zone is minimal.

The attacker, first, identifies the longest path  $Path_{before}$  taken by a user before he enters the privacy zone from the border node  $b_{in}$  that has minimum travel time. Starting from the segment that  $b_{in}$  resides on, one add segments from user trajectory until the shortest path from new segment's end node to the border node does not match with the user trajectory. Similarly attacker identify the longest path taken by user after he exits the privacy zone from border node  $b_{out}$ . Supposed that  $L_{start}$  is the start location of the path  $Path_{before}$  and  $L_{end}$  is the end location fo the path  $Path_{after}$ . Attacker then calculates two shortest path from  $L_{start}$  to  $L_{end}$  that includes  $s_i$ ; with and without the condition to pass through  $b_{in}$  and  $b_{out}$ . The likelihood of segment  $s_i$  is calculated as a ratio between two path's travel time;

$$like[s_i] = \frac{d_{tt}(L_{start}, s_i, L_{end})}{d_{tt}(Path_{before}) + d_{tt}(b_{in}, s_i, b_{out}) + d_{tt}(Path_{after})}$$

#### *Markov Model-based Transition Attack*

In this attack model, the attacker estimates the transition probability for each segment based on his previous observation over all users travel pattern. It uses a simple Markov model to make a probabilistic predictions of the mobile objects visited segments in the privacy zone by looking at a drivers' driven path before and after the privacy zone. The model is trained from the all users' long term trip history. Figure 5.7 shows a sketch of this basic approach. We model the sequence of segments taken by user as  $s_t$ , with  $t$  representing a discrete time variable.

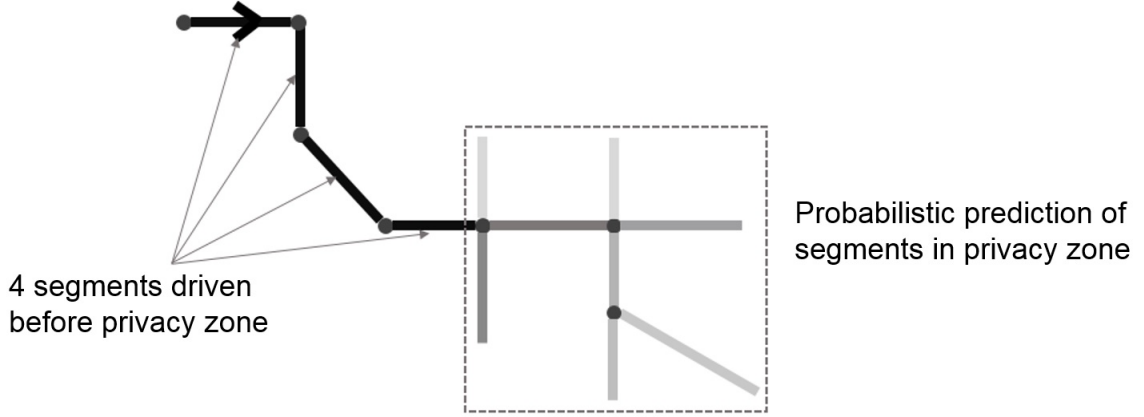


Figure 5.7: Markov model-based transition attack

$$Path_{before}\{s_{t_{in}-n}, \dots, s_{t_{in}-2}, s_{t_{in}-1}\}$$

$$Path_{before}\{s_{t_{out}+1}, s_{t_{out}+2}, \dots, s_{t_{out}+n}\}$$

The likelihood of segment  $s_i$  is calculated as probabilistic prediction based on other users taken same paths.

$$like[s_i] = P[s_i|s_{t_{in}-1}, s_{t_{in}-2}, \dots, s_{t_{in}-n}] + P[s_i|s_{t_{out}+1}, s_{t_{out}+2}, \dots, s_{t_{out}+n}]$$

### 5.3 Privacy Zone Processing

The focus of this work is to develop a privacy zone enforcement model that minimize latency, mobile client energy consumption, and the network usage while providing scalable service large number of mobile object. First, we describe two basic methods and analyze their problems. Then we introduce two models, hibernation-period and hibernation-region, which improves privacy zone monitoring efficiency significantly.

#### 5.3.1 Motivating Approaches

**Request Dependent Processing:** In this method, each location access intercepted by the PrivacyZone client is evaluated by checking if the user is in any privacy zone or not. Even though this approach provide privacy guarantee, since it delays the response for location

access requests it affects user experience. In addition for applications that require high frequent location update this approach can be extremely energy inefficient.

**Periodic Processing:** In an alternative approach it is possible to conduct periodic privacy zone checking. With this method location access request responded immediately based on the last privacy zone check. However, high frequency is essential in this approach to ensure that location requests by the applications are permitted while the user in the privacy zone.

Note that given methods are not flexible to share privacy zone processing between client and server. Each privacy zone check should be employ on either client side or server side. However, client-only approach suffer with high battery consumption. On the other hand server-only approach introduce high network usage and latency in addition to be vulnerability against Man-in-the-Middle attack. Also, with given methods privacy zone check should be employ even the user is very far from the any privacy zone. To overcome these problems we present two hibernation distance-based privacy zone monitoring approaches that can efficiently orchestrate client and server computations. The idea is as follows; the server calculates hibernation distance, either in time based or region based, that guarantees client will not be in any privacy zone within given distance. Hibernation distance-based monitoring can reduce the frequency of wake-ups and increase sleep time of mobile clients and minimize the computation cost of privacy zone checks by filtering out those zones that are irrelevant or far away from the current location of the mobile object.

### 5.3.2 Hibernation Period-Based Monitoring

The first hibernation distance-based monitoring approach is hibernation period which computes the time period during which the mobile client can continue to sleep without missing any of her privacy zones. We compute this hibernation period using the distance from current location of the mobile client to all border nodes of her privacy zones and the possible speed limit of the mobile client. We consider Euclidean distance and road network distance

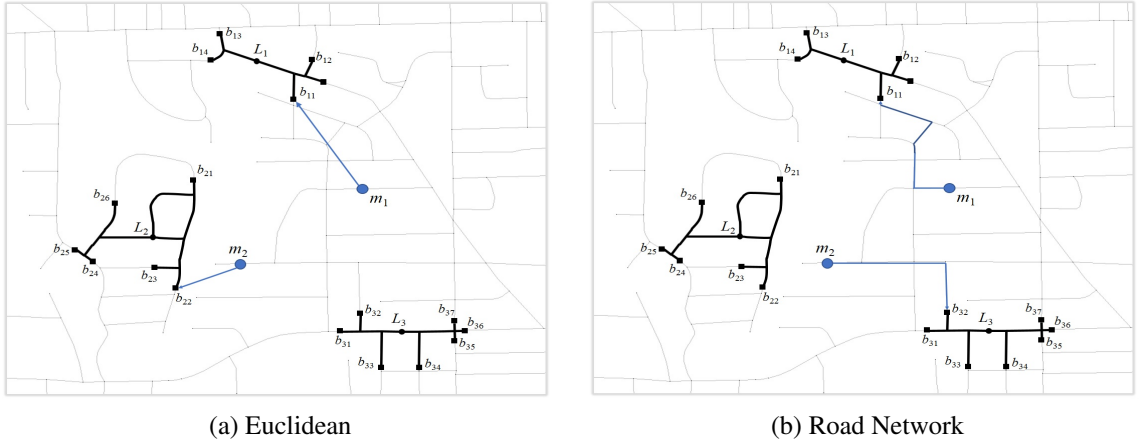


Figure 5.8: Euclidean vs Road Network Shortest Path-based Hibernation Period

as two alternative distance function. The Road Network Distance measure offers a more accurate estimate of the distance from a mobile clients current location to the spatial region of the privacy zone, but it introduces additional overhead with expensive graph processing.

#### *Euclidean distance-based hibernation period*

For mobile object  $m$ , upon the expiration of its hibernation time,  $m$  wakes up and contacts the PrivacyZone server to obtain its new hibernation time. The server first retrieves the index of all privacy zones border nodes. Then the PrivacyZone server computes the Euclidean distance between the current location of  $m$  and each of the border points for all privacy zones and selects the border point that is the nearest to the current location of  $m$ , denoted by  $b_{nearest}$ , and calculates the new hibernation time for  $m$  based on the Euclidean distance and a maximum speed ( $V_{max}$ ). Although using the global maximum speed is too pessimistic to calculate hibernation time, it provide privacy guarantee by ensuring that in the given time period mobile object can not reach any privacy zone. The end time of the new hibernation time for  $m$  based on Euclidean distance-based method using the global maximum speed is defined as  $currenttime + \frac{D_E(m, b_{nearest})}{V_{max}}$  in which  $D_E(m, b_{nearest})$  is the Euclidean distance between  $m$  and the  $b_{nearest}$ . The object  $m$  will be in the sleep mode during the above hibernation time.

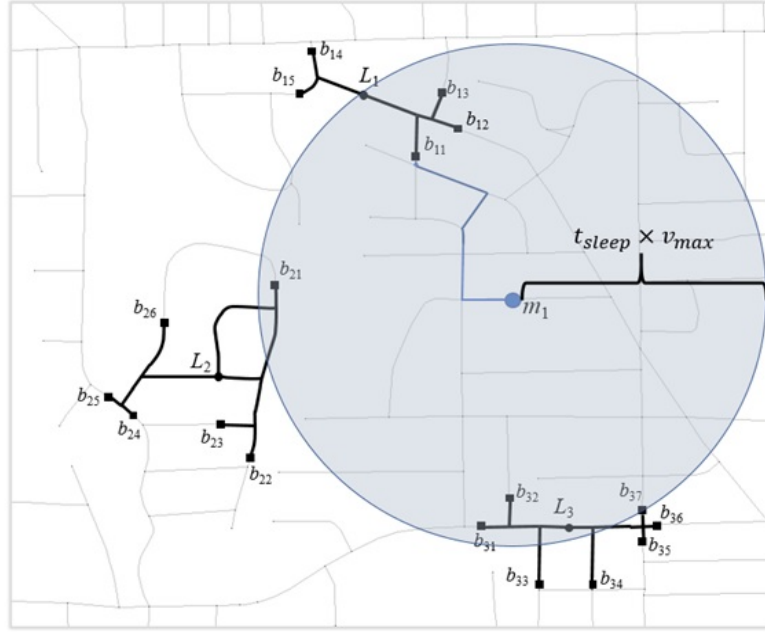


Figure 5.9: Euclidean lower bound-based filtering

Although the Euclidean distance-based hibernation period approach is simple to implement, assuming random way-point model and global maximum velocity limit leads to underestimate the hibernation time. Consequently, mobile objects need to wake up frequently, making PrivacyZone consuming higher battery than necessary. For example, Figure 5.8a shows closest border nodes selected for two mobile objects  $m_1$  and  $m_2$ . Since border selection does not consider underlying road network, calculated hibernation period for  $m_2$  to reach closest border node  $b_{22}$  is much shorter than the actual travel distance that mobile object needs to take.

#### *Road Network distance-based hibernation period*

In this approach, in order to provide better hibernation period estimation we consider underlying road network constraint which use Dijkstra's network expansion algorithm [60]. Similar to Euclidean distance based approach when a mobile object  $m$  wakes up for each privacy zone, the set of border points obtained. PrivacyZone server takes the current location of  $m$  and each border point as input to calculate the travel time-based shortest

path using Dijkstra’s network expansion algorithm. The shortest path having the smallest travel time-based distance chosen to compute the new hibernation period for  $m$ . Figure 5.8b shows shortest paths calculated for mobile objects  $m_1$  and  $m_2$  to their closest border nodes.

The network expansion-based approach is easy to implement, but the computation cost is very high because of the expensive shortest path computation for each border nodes of user’s privacy zones. To minimize shortest-path computation we use *Euclidean Lower Bound* (ELB) [17] to filter out some border points that are not possible to be the shortest network distance. The idea is travel time-based shortest path distance between two location  $L_1$  and  $L_2$  is at least equal to or longer than the minimum possible travel time calculated with the Euclidean distance divided by the global speed limit. In this approach, instead of computing shortest paths from the current location  $L_m$  of the mobile object  $m$  to every border point of all privacy zones defined by  $m$ , starting from the closest border node based on Euclidean distance PrivacyZone server computes shortest paths until the distance can be reach with minimum travel time and global maximum speed is smaller than the Euclidean distance to the remaining border nodes. As shown in Figure 5.9, after calculating shortest path travel time  $t_{sleep}$  to a border node  $b_{11}$  it is not necessary to calculate shortest paths for those border nodes (e.g.,  $b_{34}$ ) whose distance to mobile object is larger than it can reach even moving with the maximum speed limit  $v_{max}$ .

#### *Hybrid hibernation period calculation*

Euclidean-based approach calculate hibernation period faster than road-network-based approach, however it generates lower hibernation time. Calculation time in road network approach increase drastically while the distance between mobile object and border nodes increase. Additionally, when the closest border node is far from the object location, we may not leverage from the Euclidean Lower Bound optimization. In the hybrid approach we optimize the calculation time to generate hibernation period for a road network-based

processing by limiting the distance between mobile object and the closest border node. We use *alpha*, system defined parameter, as an Euclidean distance limit from mobile object to the closest border node to calculate shortest path travel time.

### 5.3.3 Hibernation Region-Based Monitoring

The hibernation period model is simple to implement and it provides minimal overhead for mobile object, however it introduce more wake-ups than necessary because of two main reasons. First, it assumes that users are moving all the time. However, mobile users may stay at certain point for a long time and even though user is not getting closer to any privacy zone it needs to send location update to the server at the end of the hibernation period. Second, assuming user will move with speed limit for each segment is an overestimate and cause unnecessary location updates. In order to overcome these limitations we presents hibernation region model, in which hibernation distance is defined as an area instead of time. Mobile objects only send their location update to the server when they move out of the hibernation region. We show that this approach enables controlled distribution of privacy zone processing load between the server and mobile objects. Such distributed processing can significantly enhance server scalability with nominal resource consumption at the client end. Note that we use hibernation period model in mobile client for area-based evaluation, which minimize number of wake-ups in mobile client as well.

The main idea underlying our distributed architecture design is twofold. First, we want to use the concept of hibernation region to reduce the amount of unnecessary privacy zone checks as the mobile objects travel on the road. Second, we promote the distribution of hibernation region-based privacy zone monitoring by the mobile clients; each mobile client determines by itself whether it has moved outside its hibernation region without requiring global knowledge of relevant privacy zones. An immediate advantage of our distributed architecture is significant savings in terms of server load and communication bandwidth. The main challenge in the design of our distributed architecture is the development of hiber-

nation region computation techniques that can provide a careful trade-off between server load and client energy consumption by taking into account: (i) the bandwidth required to communicate the safe region from the server to its corresponding mobile client, and (ii) the computation cost at a mobile client for monitoring its position with respect to the hibernation region. In summary, hibernation region computation must satisfy the following constraints:

**Lightweight construction,** The hibernation region computation should induce a low processing overhead at the server as this computation may need to be performed frequently for large number of mobile objects.

**Compact representation,** The hibernation region should have a compact representation as it needs to be communicated back to the user resulting in consumption of network bandwidth.

**Fast containment check,** Mobile users need to monitor their position the hibernation region frequently. A simple and fast containment check will enable the clients to perform this function with low energy consumption.

**Large area coverage,** Generated hibernation region should be as large as possible in order to minimize location update requirements.

These requirements motivate us to design two alternative hibernation region techniques and study the impact of size and shape of region on server load, network bandwidth and client energy consumption.

#### *Shortest Distance-based Hibernation Region*

In the simplest hibernation region, the area is determined based on the closest border node. Similar to the Euclidean hibernation period computation, PrivacyZone finds the closest border node and send the distance,  $d_{hib}$  to the mobile client. For each containment check, mobile client calculate the distance  $d_{current}$  between the last location send to the server and his current location. User guaranteed to be in the hibernation region if the current

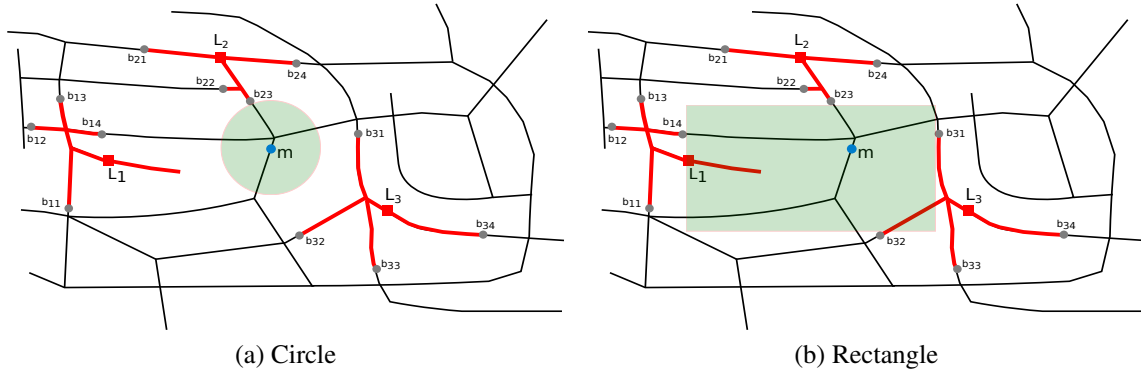


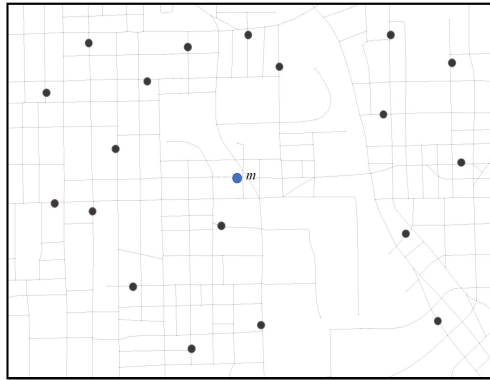
Figure 5.10: Shortest Distance vs Rectangle-shaped Hibernation Region

distance is lower than the hibernation distance (i.e.,  $d_{current} < d_{hib}$ ). Note that, instead of expensive Haversine formula we assume embedding on the plane, which [61] shows that error is negligible. Figure 5.10 shows an example hibernation region for user  $m$ , calculated based on the closest border node  $b_{23}$ .

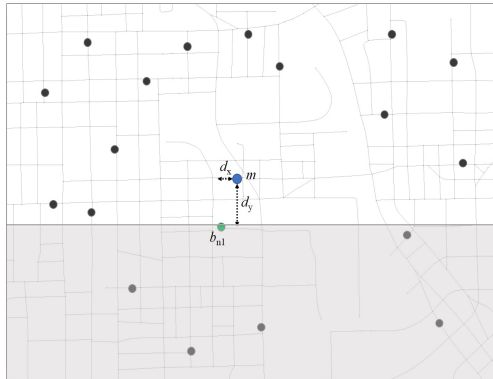
The shortest distance-based hibernation region satisfies lightweight construction, compact representation, and fast containment check requirements. However, it suffers with the relatively small area coverage, which leads to unnecessary location update.

### *Rectangle-based Hibernation Region*

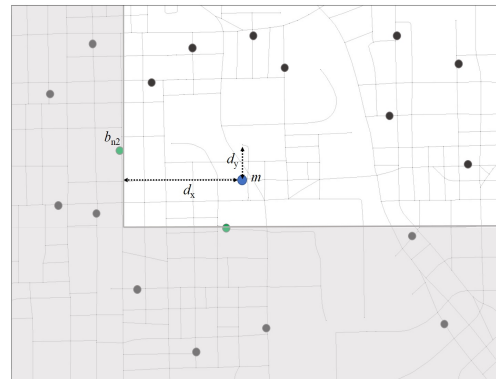
In order to minimize location update we propose simple and effective rectangle-based hibernation region generation method. As shown in Figure 5.10 generated rectangle-shaped hibernation region can cover much more area yet it provides minimal computation requirement for containment check on the mobile devices. In order to maximize the time user stays in the hibernation region we use an approach with guarantees the distance between a mobile object and the closest edge of the selected rectangle is to be the maximum. To this end until one find all the edges of the rectangle by in each step selecting the closest border node, choose the  $x$  or  $y$  axis of the node as an edge that has largest distance to the user and eliminate all the remaining points further from selected axis. Figure 5.11 shows the process step by step. First it selects closest border node  $b_{n1}$  and calculates the distance



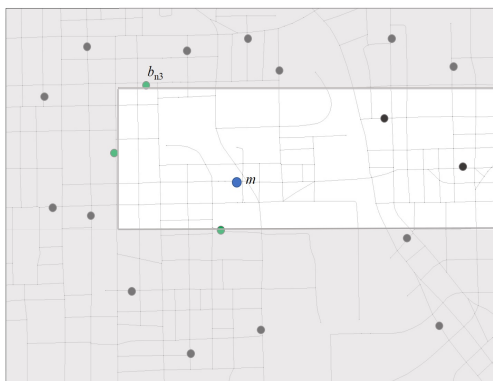
(a) Locations of mobile object and border nodes



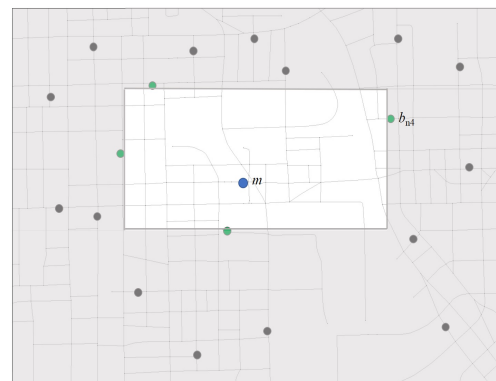
(b) Step 1



(c) Step 2



(d) Step 3



(e) Step 4

Figure 5.11: Calculating rectangle shape hibernation region

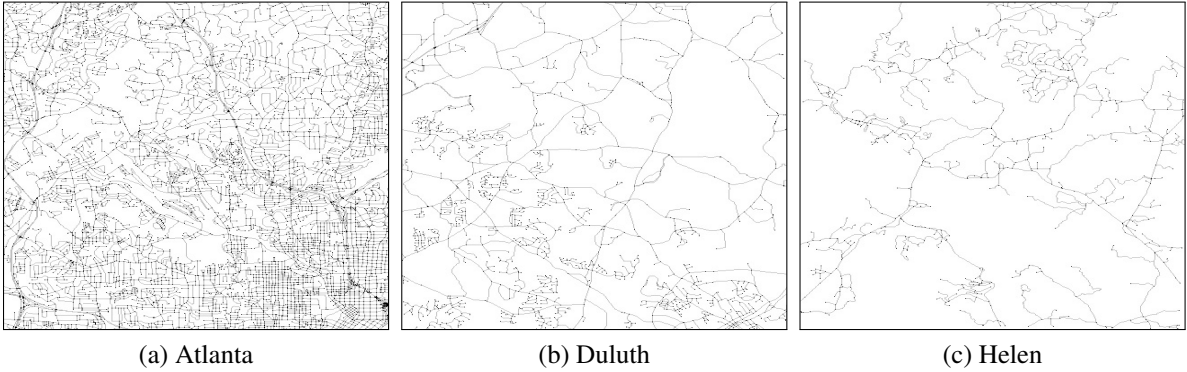


Figure 5.12: Road networks used in experiments

Table 5.1: Road network properties.

map	type	number of segments	number of junctions	avg. segment length (m)
Atlanta	urban	9,187	6,976	150.7
Duluth	suburban	1,600	1,486	258.3
Helen	rural	765	711	356.3

from mobile object to  $x$ -axis ( $d_x$ ) and  $y$ -axis ( $d_y$ ) of the border node. Since  $d_y > d_x$ ,  $y$ -axis of the  $b_{n1}$  selected as the lowest  $y$  value of the rectangle and the nodes that has lower  $y$  values eliminated for further consideration. The process continues until it finds all the edges of the rectangle.

## 5.4 Experimental Evaluation

In this section we evaluate the performance of our PrivacZone methods through set of experiments. We provide performance comparison between algorithms described in section 5.3.

### 5.4.1 Evaluation Setup

We use gt-mobisim simulator [18] to generate mobility traces on real road networks downloaded from U.S. Geological Survey <sup>4</sup>. We used three different types of road networks, Atlanta,GA, Duluth,GA, and Helen,GA as a urban, suburban, and a rural road network

<sup>4</sup><http://www.usgs.gov/>

respectively (See Figure 5.12 and Table 5.1). All three road networks cover almost the same size, 11 km ( 6.8 miles) by 14 km (8.7 miles), but have totally different number of road segments and road junctions. We use Atlanta map as a default road network in our experiments.

The road networks consist of four different road types: residential roads and freeway interchange with 30 mph speed limit (48 km/h), highway with 55 mph limit (89 km/h) and freeway with 70 mph limit (113 km/h). The urban road network has 431 and 681 road segments having 70 mph and 55 mph speed limit respectively. The other road segments have 30 mph speed limit. 24 and 218 road segments of the suburban road network have 70 mph and 55 mph as their speed limit respectively. The rural road network has 27 and 66 road segments having 70 mph and 55 mph speed limit respectively.

In our experiments we use 20,000 mobility traces generated on three maps using the random trip model [62]. As a default, nine privacy zones generated for each user. Ranges of privacy zones are chosen from a Gaussian distribution with a mean of 500 m and standard deviation of 100 m. For hybrid approach default alpha value selected as 1000 m. All experiments were conducted on a desktop having one Intel i7-4790K quad-core processor, 16GB of RAM, and one 1TB 7200 rpm hard disk.

#### 5.4.2 Experimental Results

We measure the performance of all approaches on different evaluation metrics; number of the location update to the server, average hibernation time, number of wake up on mobile device, and server/mobile processing time.

##### *Effect of growing number of privacy zones and privacy zone size*

The first set of experiments compares given approaches with respect to different number of privacy zones per person and different privacy zone ranges.

**Hibernation time.** Figure 5.13 shows the average hibernation time of moving objects.

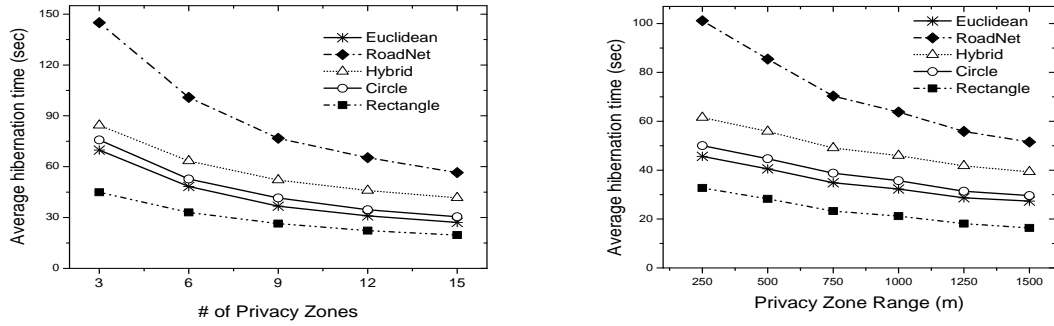


Figure 5.13: Hibernation time comparison

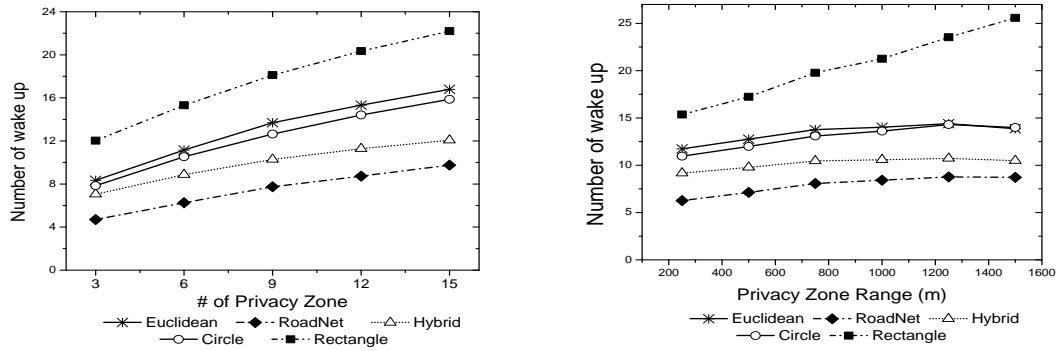


Figure 5.14: Number of wake-up comparison

The longer the hibernation time is, the more energy the mobile clients can conserve. The hibernation time of the shortest path-based filter is two times longer than that of the Euclidean distance based approach using the global maximum speed and more than three times longer than the Rectangle-shaped hibernation region model. The reason of the low hibernation time for rectangle-shaped hibernation region model is that after the sleep time ends this model needs to consider closest points on the rectangle is a possible border node location. This results also shows as expected that increasing the number privacy zones and growing privacy zone range decrease hibernation time. The effect of new privacy zone is larger than the increment on size of the privacy zone, since it increase the chance of the new border nodes to be closer to mobile object.

**The number of wake-ups.** Figure 5.14 shows that the number of wake-ups is inversely

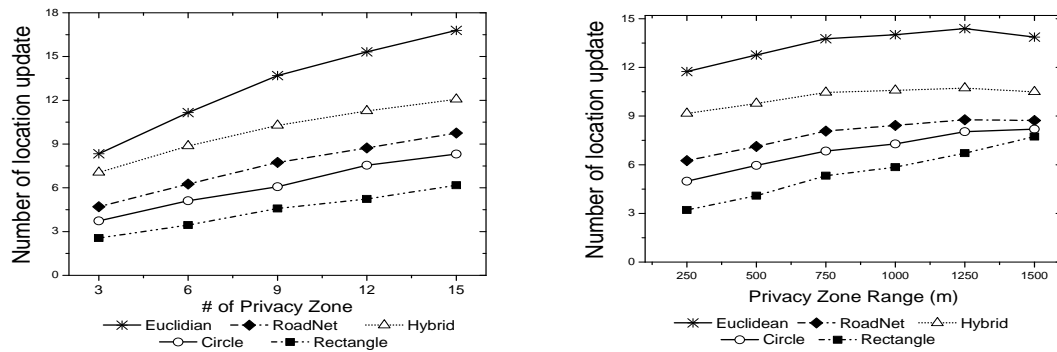
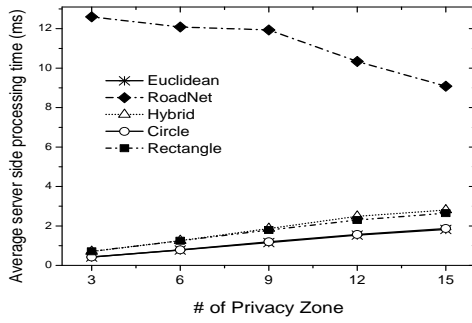


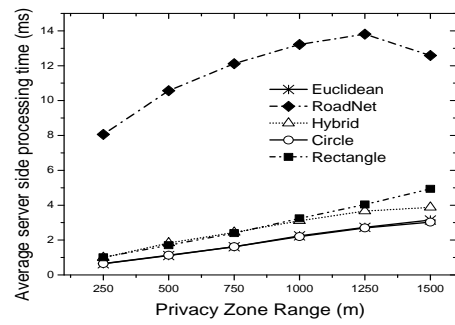
Figure 5.15: Number of location update to the server comparison

related to the hibernation time. For hibernation period-based monitoring approaches, the smaller number of wake-ups indicates the lower server loads since the server computes the hibernation time whenever a moving object wakes up. On the other hand, for hibernation region-based approaches, as the number of wake-ups increase both mobile and server loads increase, because after each wake-up if the user still in the hibernation region mobile side computes new hibernation time, otherwise it computed at the server side.

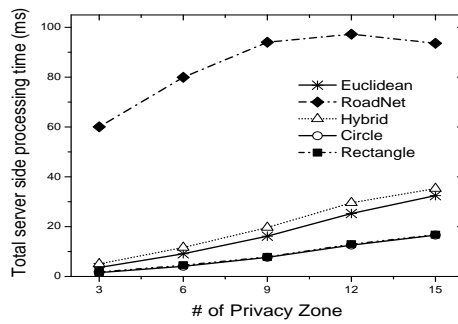
**Number of location update to the server.** Whenever mobile object wake up after the hibernation period end or exit from the hibernation zone, sends his new location to the server for new hibernation period/region calculation. Figure 5.15 shows the average number of location update send to the server during each users trajectory with respect to number of privacy zones and the size of each privacy zone, respectively. Note that, while hibernation period-based methods sends location update to the server after each wake-up, hibernation region based approaches calculates new hibernation time within client if the mobile object still locates in the hibernation region. Also, since the area coverage is larger than the circle-based approach, rectangle-based monitoring provide lowest network usage. It is important to point that, our trajectory generator does not reflects real-world user stops at certain locations or traffic waits. In real-world environment as the time user stops increase the difference between hibernation region and hibernation period models would also increase.



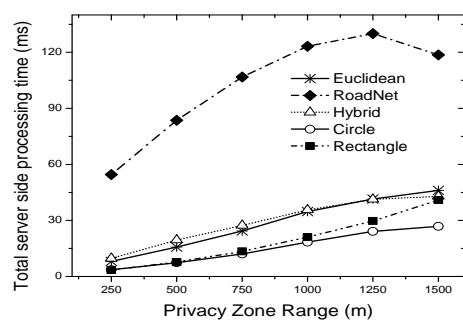
(a) Computation time per update



(b) Computation time per update



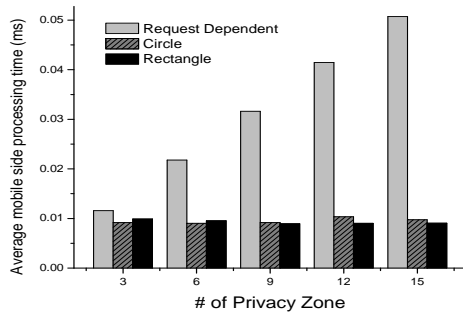
(c) Total computation per user



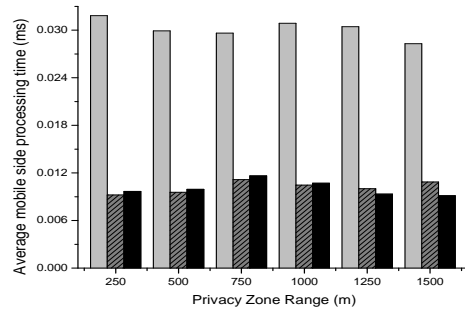
(d) Total computation per user

Figure 5.16: Server side processing time comparison

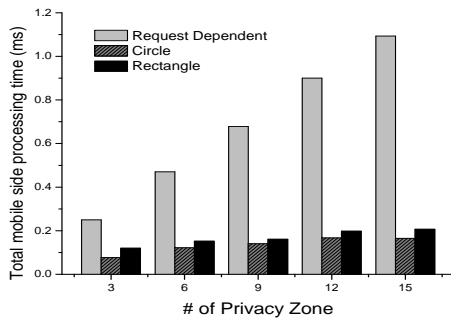
**Server side computation.** Figure 5.16 shows the server side processing load for calculating the hibernation period or the hibernation region. Average computation time for single location update presented in Figures 5.16a and 5.16b and total computation time for single user presented in Figures 5.16c and 5.16d. Compared to euclidean-based calculations, IER shows at least four times higher average computation time caused by expensive shortest-path calculation. As expected hibernation region approaches provide lower overall server load because of the low location update. Note that all approaches except IER require more time to calculate hibernation period/regions as the number of privacy zone increase. Because, as the nearest border node getting closer, *Euclidean lower bound* technique become more effective. Another important result is that even tough rectangle-based monitoring approach require more time to calculate hibernation region than the circle-based approach since the number of location update is low, total server load is similar between



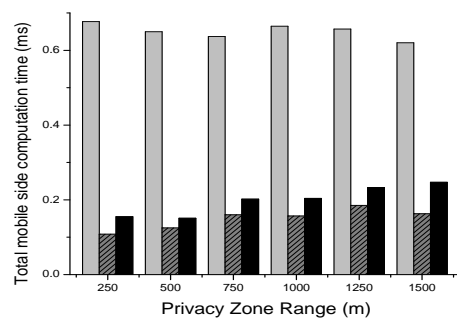
(a) Computation per request



(b) Computation per request



(c) Total computation



(d) Total computation

Figure 5.17: Client side processing time comparison

this approaches.

**Client side computation.** Figure 5.17 shows the mobile side processing load for checking if the user is still in the hibernation region and if so calculating the new hibernation time. Note that only hibernation region-based monitoring approach requires mobile side processing. For the comparison we use request dependent processing approach for checking if the user is in any of his privacy zone. In this experiment we used 30 sec as the frequency of location request. As expected even for low frequency request dependent approach is not viable solution for current mobile environments.

### *Privacy protection evaluation*

With this set of experiments we evaluate the effectiveness of attack models introduced in Section 5.2.6. Figure 5.18a shows entropy values calculated for different size of privacy

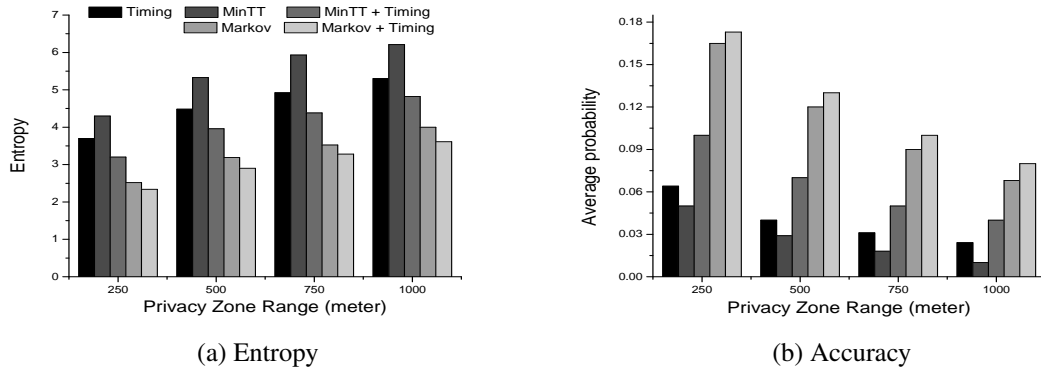
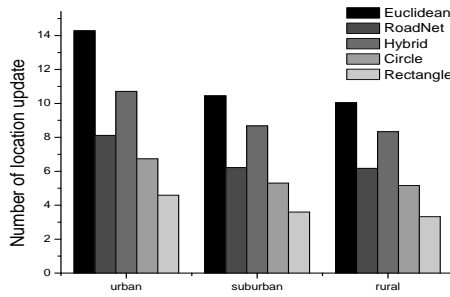


Figure 5.18: Privacy protection

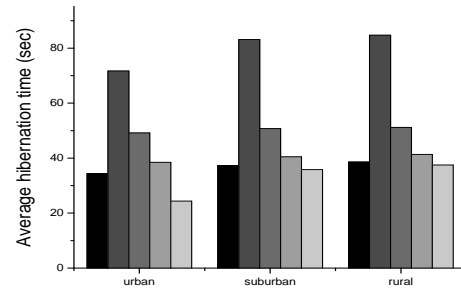
zones. As expected, the strength of the privacy zone is directly proportional to the number of segment covered by privacy zone. Since the trajectory generator used in our experiments does not reflect mobile users stops or slowdowns, timing attack shows better attack strength than the minimum travel time distance-based attack model. Among there attack model Markov Model-based transition attack outperforms other attack models as it predicts user future locations more effectively. We propose to use timing attack as an additional check for other attack models in order to eliminate assigning probabilities for segments that user can not be reached. In order to validate the correctness of entropy values, we compare assigned probabilities of each segment on the original trajectory. Our accuracy test in Figure 5.18b shows correlated result with the entropy.

### *Effect of different road networks*

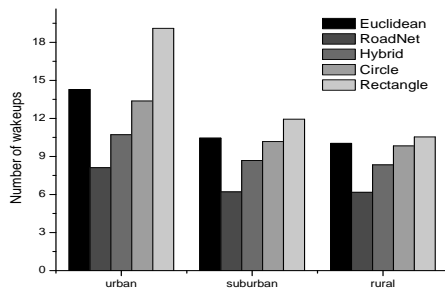
This set of experiments measure the performance of given privacy zone monitoring approaches using different types of road networks. Figure 5.19 shows the experimental results for the urban (Atlanta), suburban (Duluth) and rural (Helen) area road networks. The result confirms that as the number of shortest-path computation increase performance of IER approach gets worse. With respect to hibernation time and number of wake-ups the rectangle-based hibernation region monitoring approach is the most sensitive to road net-



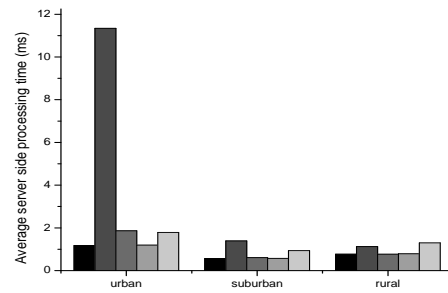
(a) Location update to the server



(b) Hibernation time



(c) Number of wake-ups



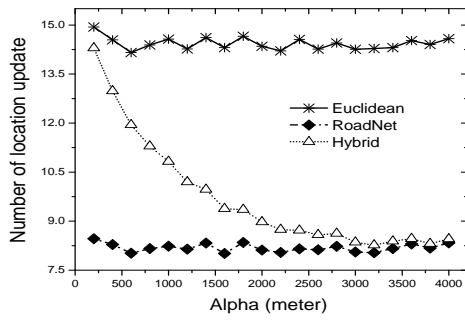
(d) Computation per update

Figure 5.19: Effect of different road networks

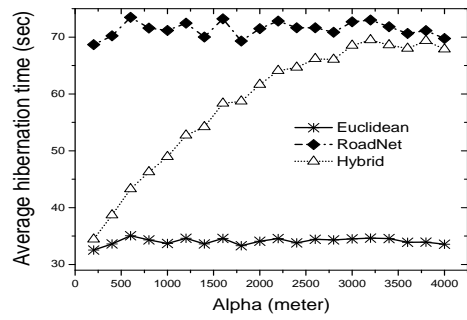
work structure. As the distance increase between roads with rectangle-based approach the chance of staying in the hibernation region increase more than other approaches.

### *Deciding the right alpha value*

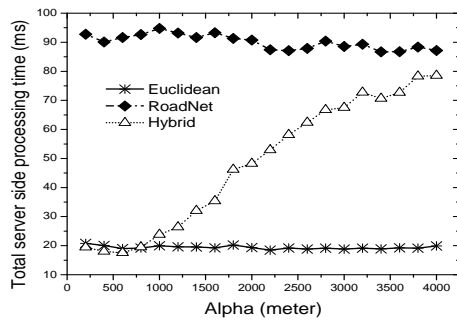
In hybrid hibernation-period monitoring approach we aim to provide optimal balance between Euclidean and IER models. With this set of experiments we showed the effect of selected alpha value. Figure 5.20 shows that for increasing alpha for low values (250 m to 1000 m) has significant effect on hibernation time but does not increase server side processing time. Based on this results we used 1000 meter as a default alpha values for our experiments.



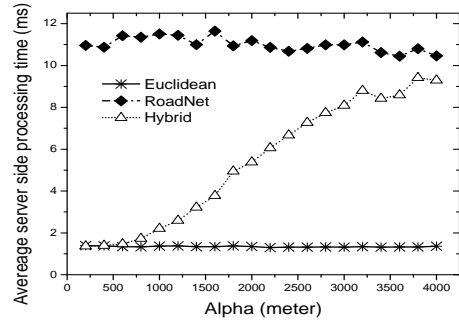
(a) Location update to the server



(b) Hibernation time



(c) Total computation



(d) Computation per request

Figure 5.20: Deciding right alpha value

## CHAPTER 6

### CONCLUSION

This thesis center around designing, building, and deployment privacy-aware services in the Internet of Things environment. We have presented orchestration framework, called ISYMPHONY, aiming at scaling real-time and on-demand IoT services provisioning in large scale IoT systems. A main idea behind our distributed IoT orchestration architecture is the intelligent partition of a real time IoT services provisioning task into an optimal coordination of server-side processing at IoT gateway and client-side process at edge clients running on edge devices. We also develop a road network-based partitioning mechanism that considers the density of each segment in order to minimize the number of inter-partition changes. In addition, we employ a set of optimizations to limit the amount of unnecessary processing on edge gateway side (object side) to enhance the overall performance and resource utilization of the ISYMPHONY system. Our initial experiments show that ISYMPHONY offers significant savings in terms of server load and thus improving server throughput by leveraging and coordinating the object side processing capabilities on edge gateways, compared to the solutions relying solely on server level processing for real time IoT services provisioning.

We have presented Foggy framework that offers an automated IoT application deployment and update model in order to provide optimal resource management in heterogeneous IoT environments. We identified several requirements in common IoT applications. Latency-sensitivity, dynamic workload, resource constraints and heterogeneity of the IoT devices, bandwidth consumption, and privacy preservation are the main constraints that we have taken into consideration during the design of our orchestration framework. As a proof of concept we implemented a prototype of Foggy and deployed it to a cluster of Raspberry Pi boards. Note that this work is the initial development of our IoT application deployment

framework. There are several research work that we plan to further improve our system. First of all, we are planning to extend our prototype for a large-scale real-world IoT application. Another research direction is to improve our policy-based resource management procedure with optimization techniques. The goal of such optimization techniques is not only optimal container placement on the nodes but also to support scalability for thousands of nodes and several applications with multiple container sequences. As a future direction also, we are planning to put in place a machine learning component that can learn from the behavior of the different deployed applications and recommend some requirements adjustment in certain context changes.

In this dissertation research we also provide privacy-preserving models and algorithms for IoT systems. First we have presented StarCloak, a query utility-aware and attack-resilient location anonymization technique for mobile users travels on the road network. StarCloak generate cloak regions with strong privacy-preserving grantee while minimize additional costs (latency and communication overhead) introduced by anonymization. Second, we propose new privacy preserving system, PrivacyZone, for the current mobile applications. The goal of the project is to hide user location in the user specified quarantine areas with minimal overhead and privacy guarantee.

## **6.1 Future Directions**

There are many interesting open research problems for the context of Internet of Things from various perspectives.

First of all, in ISymphony we considered a random way-point model for the range queries. However, in real world mobile objects are mostly interested in the road network distance of other objects. As the mobile object has limited capacity to process road network based queries, it is important to develop novel approach to minimize energy consumption on the mobile devices. We are also planning to develop advance road network partitioning mechanism that can further improve our overall system performance.

There are also several research work that we plan to further improve our IoT application deployment framework Foggy. We are planning to extend our prototype for a large-scale real-world IoT application. Another research direction is to improve our policy-based resource management procedure with optimization techniques. The goal of such optimization techniques is not only optimal container placement on the nodes but also to support scalability for thousands of nodes and several applications with multiple container sequences. As a future direction also, we are planning to put in place a machine learning component that can learn from the behavior of the different deployed applications and recommend some requirements adjustment in certain context changes.

Finally, for PrivacyZone in order to minimize server/mobile processing loads we are planning to develop motion-aware hibernation region calculation mechanisms that considers the direction of the mobile object to maximize the time user stay in the hibernation region. In addition, we are planning to develop dynamic privacy zone creation techniques that can generate more compact privacy zones on fly with the same privacy guarantee. Also, since PrivacyZone is a new concept for location privacy protection, it is worth to investigate more sophisticated attack models.

## REFERENCES

- [1] “Global sensors in internet of things (iot) devices market, analysis & forecast: 2016 to 2022,” Research and Markets, Tech. Rep., 2017.
- [2] “Unlocking the potential of internet of things, 20152020,” McKinsey Global Institute Report, Tech. Rep., 2015.
- [3] “Cisco global cloud index: forecast and methodology, 20152020,” Tech. Rep., 2016.
- [4] J. Research, *Mobile Context and Location Services*, <http://www.juniperresearch.com/press-release/context-and-location-based-services-pr2>, [Online; accessed 05-March-2018], 2014.
- [5] P. R. Center, *Location-based Services*, <http://www.pewinternet.org/2013/09/12/location-based-services>, [Online; accessed 05-March-2018], 2013.
- [6] L. Liu, “Privacy and location anonymization in location-based services,” *SIGSPATIAL Special*, vol. 1, no. 2, pp. 15–22, 2009.
- [7] B. Gedik and L. Liu, “Protecting location privacy with personalized k-anonymity: architecture and algorithms,” *TMC*, vol. 7, pp. 1–18, 1 2008.
- [8] B. Bamba, L. Liu, P. Pesti, and T. Wang, “Supporting anonymous location queries in mobile environments with privacygrid,” in *WWW*, 2008, pp. 237–246.
- [9] G. Ghinita, P. Kalnis, and S. Skiadopoulos, “Prive: anonymous location-based queries in distributed mobile systems,” in *WWW*, 2007, pp. 371–380.
- [10] M. F. Mokbel, C.-Y. Chow, and W. G. Aref, “The new casper: query processing for location services without compromising privacy,” in *VLDB*, 2006, pp. 763–774.
- [11] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, “A performance comparison of multi-hop wireless ad hoc network routing protocols,” in *MobiCom*, 1998, pp. 85–97.
- [12] E. Hytiä and J. Virtamo, “Random waypoint mobility model in cellular networks,” *Wireless Networks*, vol. 13, pp. 177–188, 2 2007.

- [13] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, “Fog computing: a platform for internet of things and analytics,” in *Big Data and Internet of Things: A Roadmap for Smart Environments*, Springer, 2014, pp. 169–186.
- [14] B. Gedik and L. Liu, “Mobieyes: distributed processing of continuously moving queries on moving objects in a mobile system,” Springer.
- [15] D. L. Mills, “Internet time synchronization: the network time protocol,” *IEEE Transactions on communications*, vol. 39, no. 10, pp. 1482–1493, 1991.
- [16] B. Bamba, L. Liu, A. Iyengar, and S. Y. Philip, “Distributed processing of spatial alarms: a safe region-based approach,” in *Distributed Computing Systems, 2009. ICDCS’09. 29th IEEE International Conference on*, IEEE, 2009, pp. 207–214.
- [17] K. Lee, L. Liu, B. Palanisamy, and E. Yigitoglu, “Road network-aware spatial alarms,” *IEEE Transactions on Mobile Computing*, vol. 15, no. 1, pp. 188–201, 2016.
- [18] P. Pesti, *GTMobiSIM*, <http://code.google.com/p/gt-mobisim/>, [Online; accessed 05-March-2018], 2009.
- [19] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, “The  $r^*$ -tree: an efficient and robust access method for points and rectangles,” in *Acm Sigmod Record*, ACM, vol. 19, 1990, pp. 322–331.
- [20] M. Vögler, J. Schleicher, C. Inzinger, S. Nastic, S. Sehic, and S. Dustdar, “Leonore—large-scale provisioning of resource-constrained iot deployments,” in *Service-Oriented System Engineering (SOSE), 2015 IEEE Symposium on*, IEEE, 2015, pp. 78–87.
- [21] B. Cheng, A. Papageorgiou, and M. Bauer, “Geelytics: enabling on-demand edge analytics over scoped data sources,” in *Big Data (BigData Congress), 2016 IEEE International Congress on*, IEEE, 2016, pp. 101–108.
- [22] S. Distefano, G. Merlino, and A. Puliafito, “Application deployment for iot: an infrastructure approach,” in *Global Communications Conference (GLOBECOM), 2013 IEEE*, IEEE, 2013, pp. 2798–2803.
- [23] F. Li, M. Vögler, M. Claeßens, and S. Dustdar, “Efficient and scalable iot service delivery on cloud,” in *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on*, IEEE, 2013, pp. 740–747.
- [24] A. Papageorgiou, M. Zahn, and E. Kovacs, “Auto-configuration system and algorithms for big data-enabled internet-of-things platforms,” in *Big Data (BigData Congress), 2014 IEEE International Congress on*, IEEE, 2014, pp. 490–497.

- [25] E. Theodoridis, G. Mylonas, and I. Chatzigiannakis, “Developing an iot smart city framework,” in *Information, intelligence, systems and applications (iisa), 2013 fourth international conference on*, IEEE, 2013, pp. 1–6.
- [26] L. Sanchez, L. Muñoz, J. A. Galache, P. Sotres, J. R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis, *et al.*, “Smartsantander: iot experimentation over a smart city testbed,” *Computer Networks*, vol. 61, pp. 217–238, 2014.
- [27] L. Ling, M. Loper, Y. Ozkaya, A. Yasar, and E. Yigitoglu, “Machine to machine trust in the iot era,” in *The 18th International workshop on Trust in Agent Societies*, Singapore, 2016.
- [28] “Internet of things (iot) 2013 to 2020 market analysis: billions of things, trillions of dollars,” International Data Corporation, Tech. Rep., 2013.
- [29] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, “Face recognition: a literature survey,” *ACM computing surveys (CSUR)*, vol. 35, no. 4, pp. 399–458, 2003.
- [30] E. Auvinet, E. Grossmann, C. Rougier, M. Dahmane, and J. Meunier, “Left-luggage detection using homographies and simple heuristics,” in *In Proc. 9th IEEE International Workshop on Performance Evaluation in Tracking and Surveillance (PETS06)*, 2006, pp. 51–58.
- [31] D. Merkel, “Docker: lightweight linux containers for consistent development and deployment,” *Linux Journal*, vol. 2014, no. 239, p. 2, 2014.
- [32] S. Li, L. Da Xu, and S. Zhao, “The internet of things: a survey,” *Information Systems Frontiers*, vol. 17, no. 2, pp. 243–259, 2015.
- [33] S. K. Mohalik, N. C. Narendra, R. Badrinath, M. B. Jayaraman, and C. Padala, “Dynamic semantic interoperability of control in iot-based systems: need for adaptive middleware,” in *IEEE 3rd World Forum on Internet of Things (WF-IoT)*, 2016, pp. 199–203.
- [34] M. Satyanarayanan, P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, W. Hu, and B. Amos, “Edge analytics in the internet of things,” *IEEE Pervasive Computing*, vol. 14, no. 2, pp. 24–31, 2015.
- [35] B. Cheng, A. Papageorgiou, F. Cirillo, and E. Kovacs, “Geelytics: geo-distributed edge analytics for large scale iot systems based on dynamic topology,” in *IEEE 2nd World Forum on Internet of Things (WF-IoT)*, 2015, pp. 565–570.
- [36] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwalder, and B. Koldehofe, “Mobile fog: a programming model for large-scale applications on the internet of things,”

in *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing*, ACM, 2013, pp. 15–20.

- [37] M. Aazam and E.-N. Huh, “Dynamic resource provisioning through fog micro data-center,” in *Pervasive Computing and Communication Workshops (PerCom Workshops)*, 2015 IEEE International Conference on, IEEE, 2015, pp. 105–110.
- [38] T. Kirkham, A. Sinha, N. Parlavantzas, B. Kryza, P. Fremantle, K. Kritikos, and B. Aziz, “Privacy aware on-demand resource provisioning for iot data processing,” in *Internet of Things. IoT Infrastructures: Second International Summit, IoT 360 2015, Rome, Italy, October 27-29, 2015, Revised Selected Papers, Part II*, Springer, 2016, pp. 87–95.
- [39] A. R. Beresford and F. Stajano, “Mix zones: user privacy in location-aware services,” in *PERCOMW*, 2004, pp. 127–131.
- [40] B. Palanisamy and L. Liu, “Mobimix: protecting location privacy with mix-zones over road networks,” in *ICDE*, 2011, pp. 494–505.
- [41] X. He, G. Cormode, A. Machanavajjhala, C. M. Procopiuc, and D. Srivastava, “Dpt: differentially private trajectory synthesis using hierarchical reference systems,” *VLDB*, vol. 8, no. 11, pp. 1154–1165, 2015.
- [42] W. Qardaji, W. Yang, and N. Li, “Differentially private grids for geospatial data,” in *ICDE*, 2013, pp. 757–768.
- [43] G. Cormode, C. Procopiuc, D. Srivastava, E. Shen, and T. Yu, “Differentially private spatial decompositions,” in *ICDE*, 2012, pp. 20–31.
- [44] C. Dwork, “Differential privacy,” in *ICALP*, 2006, pp. 1–12.
- [45] T. Wang and L. Liu, “Privacy-aware mobile services over road networks,” *VLDB*, vol. 2, pp. 1042–1053, 1 2009.
- [46] M. Gruteser and D. Grunwald, “Anonymous usage of location-based services through spatial and temporal cloaking,” in *MobiSys*, 2003, pp. 31–42.
- [47] H.-J. Cho and C.-W. Chung, “An efficient and scalable approach to cnn queries in a road network,” in *VLDB*, 2005, pp. 865–876.
- [48] H. Hu, J. Xu, and D. L. Lee, “A generic framework for monitoring continuous spatial queries over moving objects,” in *SIGMOD*, 2005, pp. 479–490.
- [49] M. Kolahdouzan and C. Shahabi, “Voronoi-based k nearest neighbor search for spatial network databases,” in *VLDB*, 2004, pp. 840–851.

- [50] K. Mouratidis, M. L. Yiu, D. Papadias, and N. Mamoulis, “Continuous nearest neighbor monitoring in road networks,” in *VLDB*, 2006, pp. 43–54.
- [51] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao, “Query processing in spatial network databases,” in *VLDB*, 2003, pp. 802–813.
- [52] C.-Y. Chow and M. F. Mokbel, “Enabling private continuous queries for revealed user locations,” in *International Symposium on Spatial and Temporal Databases*, Springer, 2007, pp. 258–275.
- [53] C. Díaz, S. Seys, J. Claessens, and B. Preneel, “Towards measuring anonymity,” in *PET*, 2003, pp. 54–68.
- [54] C.-Y. Chow, M. F. Mokbel, J. Bao, and X. Liu, “Query-aware location anonymization for road networks,” *GeoInformatica*, vol. 15, no. 3, pp. 571–607, 2011.
- [55] K. Olejnik, I. I. Dacosta Petrocelli, J. C. Soares Machado, K. Huguenin, M. E. Khan, and J.-P. Hubaux, “Smarper: context-aware and automatic runtime-permissions for mobile devices,” in *Proceedings of the 38th IEEE Symposium on Security and Privacy (SP)*, IEEE, 2017.
- [56] *LBE Privacy Guard*, <http://forum.xda-developers.com/showthread.php?t=1091065>, [Online; accessed 05-March-2018], 2011.
- [57] H. Almuhammedi, F. Schaub, N. Sadeh, I. Adjerid, A. Acquisti, J. Gluck, L. F. Cranor, and Y. Agarwal, “Your location has been shared 5,398 times!: a field study on mobile app privacy nudging,” in *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, ACM, 2015, pp. 787–796.
- [58] B. Liu, J. Lin, and N. Sadeh, “Reconciling mobile app privacy and usability on smartphones: could user privacy profiles help?” In *Proceedings of the 23rd international conference on World wide web*, ACM, 2014, pp. 201–212.
- [59] M. Götz, S. Nath, and J. Gehrke, “Maskit: privately releasing user context streams for personalized mobile applications,” in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, ACM, 2012, pp. 289–300.
- [60] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.
- [61] D. Delling, A. V. Goldberg, M. Goldszmidt, J. Krumm, K. Talwar, and R. F. Werneck, “Navigation made personal: inferring driving preferences from gps traces,” in *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ACM, 2015, p. 31.

- [62] P. Pesti, L. Liu, B. Bamba, A. Iyengar, and M. Weber, “Roadtrack: scaling location updates for mobile clients on road networks with query awareness,” *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 1493–1504, 2010.