Caching in VANETs for Social Networking

by

Sara Abdelhamid Elsayed

A thesis submitted to the
School of Computing
in conformity with the requirements for
the degree of Doctor of Philosophy

Queen's University
Kingston, Ontario, Canada
December 2020

Dedication

To Dad, Mom, and My Brother Karim.

with all my love.

Abstract

Social media traffic constitutes the highest percentage of Internet traffic. This social media traffic is largely facilitated by mobile devices, which imposes a huge traffic load on backhaul links in 5G networks, and can in turn affect the quality of service. This traffic load can be alleviated by using vehicular networks as a traffic offloading platform. In particular, vehicles can act as a resourceful asset for edge caching, which enables data acquisition from nearby caching nodes rather than the remote backhaul servers. However, caching in vehicular networks encounters many challenging issues. These include the highly dynamic nature of vehicles, which can lead to instability in caching decisions. Also, despite being equipped with storage resources, relying on static roadside units for caching might not always be feasible. This is due to the large investments that their wide deployment requires. Another challenge is prompted by the high delay and low packet delivery ratio often associated with accessing data from distant content providers in vehicular networks, which makes it imperative to take the quality of service into consideration during caching.

In this thesis, we propose a caching framework that aims at maximizing cache hit ratio, as well as improving the quality of VANET-based Internet services. To do so, we consider two types of users; users who exhibit a predictable behavior resulting from their daily routine during driving from one place to another, and those who do not

have such a routine. We exploit the predictable behavior of the first type of users in order to pre-cache the data at roadside parked vehicles for requesters to proactively acquire as they pass by. In order to promote informed proactive cache placement decisions that consider the spatiotemporal availability of replicas, we propose a travel time prediction scheme to be incorporated into the caching process.

For the second type of users, we use the static and mobile nature of parked and moving vehicles, respectively, to leverage the use of cooperative caching within the context of VANETs. This solution involves a content discovery module and a cache placement module. The former enables the nodes to dynamically locate replicas that are close to the requester during the request-forwarding process. The latter facilitates making informed caching decisions to decide where and what to cache. Our cooperative caching solution involves extending the search space and the cooperation range beyond the neighborhood scope in order to increase cache hits. We also cope with the dynamic nature of vehicles by proposing a vehicles' trajectory prediction scheme. Extensive simulations demonstrate the ability of the proposed approaches to achieve significant improvements in their targeted objectives compared to other prominent caching and prediction schemes in VANETs.

Co-Authorship

Journal Articles

1. **S. A. Elsayed**, S. Abdelhamid, and H. S. Hassanein, "LSTM-based Travel Time Prediction in VANETs", IEEE Transactions on Vehicular Technology. (in preparation)

2. **S. A. Elsayed**, S. Abdelhamid, and H. S. Hassanein, "Prediction-Assisted Cooperative Cache Discovery in VANETs for Social Networking", IEEE Transactions on Vehicular Technology. (under review)

3. **S. A. Elsayed**, S. Abdelhamid, and H. S. Hassanein, "Predictive Proactive Caching in VANETs for Social Networking", IEEE Transactions on Intelligent Transportation Systems. (under review)

Conference Publications

1. **S. A. Elsayed**, S. Abdelhamid, and H. S. Hassanein, "Optimal Proactive Caching in VANETs for Social Networking", in Proceedings of the IEEE Global Communications Conference (GLOBECOM '19), December 2019.

2. **S. A. Elsayed**, S. Abdelhamid, and H. S. Hassanein, "Probabilistic Cooperative Caching in VANETs for Social Networking", in Proceedings of the IEEE

Global Communications Conference (GLOBECOM '18), December 2018.

3. **S. A. Elsayed**, S. Abdelhamid, A. M. Nagib, and H. S. Hassanein, "Tracking-based Cooperative Caching in VANETs for Social Networking", in Proceedings of the IEEE LCN Workshop On User MObility and VEhicular Networks (IEEE LCN On-MOVE 2018), October 2018.

4. **S. A. Elsayed**, S. Abdelhamid, and H. S. Hassanein, "Proactive Caching at Parked Vehicles for Social Networking", in Proceedings of the IEEE International Conference on Communications (ICC '18), Kansas, MO, USA, May 2018.

5. S. Abdelhamid, **S. A. Elsayed**, and H. S. Hassanein, "Caching and Forwarding Assistance for Vehicular Information Services with Mobile Requesters", in Proceedings of the IEEE LCN Workshop On User MObility and VEhicular Networks (LCN ON-MOVE 2015 Workshop), Florida, USA, October 2015.

Acknowledgments

Praises and thanks to God the most merciful for blessing me with the opportunity to pursue this journey that enabled me to learn and explore new opportunities, and for helping me every step of the way.

A special thanks to my supervisor and mentor, Prof. Hossam Hassanein. Thank you Prof. Hossam for guiding and supporting me throughout this process with your patience, kindness, and vast knowledge, and for doing this while granting me enough room to work in my own way. I am forever grateful and heartily thankful for your continuous encouragement, unfailing advices, and insightful visions, and for always caring about my well-being. It was a great privilege working with you and learning from you in every aspect. Observing your diligence and commitment to your work, as well as your sincere investment in the success and prosperity of your students has been such an inspiration. Thank you for all your valuable insights on how to create and entertain new research ideas, and how to demonstrate an exceptional level of leadership and understanding towards multiple students who have different characters, potentials, and needs. I have a great respect for you, and I hope that if I ever get the opportunity to pursue a career in academia to be able to develop the same level of professional excellence, while maintaining that much decency and grace in treating my students as you treat yours.

I would like to express my appreciation towards my co-supervisor and dear friend, Dr. Sherin Abdelhamid. I cannot thank you enough Sherin for your constant help and support throughout this journey. You have always been there for me, and for that I am eternally grateful. The level of knowledge and expertise you have in this field has always been a valuable asset, and a source of appreciation and admiration. I am extremely proud of you and of all the success you have achieved in your career so far, and I am thankful that I got to work with you. Many thanks for being gracious and generous enough to help me cope with the stress of graduate studies, and for extending your endless encouragement and understanding. Thank you for knowing how to balance between the two roles of being a great friend when I needed one, and an insightful advisor when your professional guidance was sought.

To my supervisory committee, Dr. Farhana Zulkernine, and Dr. Abdelhamid Taha, thank you for your valuable comments and feedback throughout this process, and particularly during my COMP exam.

I would also like to express my gratitude and appreciation towards Mrs. Basia Palmer, who provided me with all the help and support I needed during her work in the department, and was extremely kind towards me. Many thanks to Mrs. Debby Robertson as well for always being there to help make my transition to Queen's easier since day one.

I was also fortunate enough to share this experience with a magnificent group of colleagues and friends in the Telecommunications Research Lab (TRL). I am extremely grateful to TRL alumni, Hesham Farahat and Ramy Atawia for teaching me how to use the network simulator NS-3. Special thanks to Hesham for overseeing my first project using NS-3, and for bearing with me during all those times I sought his

help without once turning me down. Many thanks to TRL alumnus, Wenjie Li for providing me with the resources I needed to learn Gurobi and for helping me integrate it with NS-3. Thank you Wenjie for all the stimulating discussions we had, and for always being there to help. I am also very thankful to Dr. Sharief Oteafy for his helpful and cheerful attitude during the time he was working as a postdoctoral fellow in TRL, and for so graciously extending words of comfort and encouragement when I needed them. Words cannot express my gratitude towards my TRL friends Faria, Mary, Basma, Habiba, and Rawan for the enormous amount of support they provided me throughout this journey, and for all the fruitful discussions and memorable time we had. Many thanks for the help and support of Amir Ibrahim and Ahmad Nagib, whom I have known for years before we even joined TRL. It was great having familiar faces that linked me to beautiful memories back home. Thank you Ahmad for always being so thoughtful, gracious, and kind that I don't remember seeking your help with anything without you trying everything you possibly can to do it. Thank you Amir for your kind support, and for the fact that you and your beautiful wife Samiha have welcomed me several times to your home with such warmth that made me feel like family. I would also like to thank Mohamed Adel, who is such a decent and helpful colleague in every possible way. Many thanks to Dr. Sameh, Dr. Hazem, Dr. Ahmed Elbery, Abdalla Abdelrahman, Galal Hassan, Mohannad, Amany, Mohamed Anas, Amr Abutuleb, Sherif Azmy, and Ayah Abusara for all the presentations and discussions we had during our regular TRL meetings.

Special thanks to my friend Shady Khalifa for guiding me during the process of applying to Queen's and for offering his help throughout every step till I came to Kingston. I am forever grateful for your help Shady. I am really thankful for having

the privilege of knowing you and your wonderful wife, my dear friend Omneya, who has also been there for me during difficult times. Thank you to my wonderful friend Gehan Selim who welcomed and helped me with everything I needed when I arrived to Kingston. Much appreciation to the wonderful community I encountered in Kingston as well.

Finally, I would like to extend my eternal gratitude towards my beloved parents and brother Karim. Thank you for your unconditional love and unfailing support. I will always be indebted to you for believing in me, and for always having my back. You have always managed to make me feel better throughout every difficulty and challenge, and have helped me stay calm in the most stressful of times. It has been your love, compassion, wisdom, patience, and kindness that got me to this point. Thank you for putting up with me, and for being my backbone and the most precious blessing I have ever had. Without you, this thesis would not have been possible.

Statement of Originality

I hereby certify that all the work presented in this PhD thesis is original, and all notions and/or techniques attributed to others have been fully acknowledged in accordance with the proper referencing practices.

Contents

List of Tables

List of Figures

List of Acronyms

5G Fifth Generation.

AIB Availability Information Base.

ANN Artificial Neural Network.

ARCA Any Relation Clustering Algorithm.

ARIMA Autoregressive Integrated Moving Average.

BP Backpropagation.

BPC Broadcast Proactive Caching.

C-V2X Cellular Vehicle-to-Everything.

CADD Caching-Assisted Data Delivery.

CADPC Caching-Assisted Data Delivery and Distributed Probabilistic Caching.

CCD Cooperative Content Discovery.

CCDLC Cooperative Content Discovery with List of Clusters.

CH Cluster Head.

DPC Distributed Probabilistic Caching.

DSC Dice Similarity Coefficient.

DSRC Dedicated Short Range Communications.

DTW Dynamic Time Warping.

FCM Fuzzy C-Means.

GC GroupCaching.

GD Gradient Descent.

GD-LSTM Gradient Descent-Long Short-Term Memory.

GD-LSTM-NW Gradient Descent-Long Short-Term Memory-No Weather.

GPS Global Positioning System.

ICN Mobile Ad Hoc Network.

ILP Integer Linear Programming.

INFORM INterest FORwarding Mechanism.

ITS Intelligent Transportation Systems.

LC List of Clusters.

LCD List of Cached Data.

LPR List of Proactive Requesters.

LSTM Long Short-Term Memory.

LUV Least Unified Value.

MANET Mobile Ad Hoc Network.

MED Minimum Edit Distance.

MTD-Probit Mixture Transition Distribution-Probit.

OBU On-Board Units.

PACD Prediction-Assisted Cooperative Content Discovery.

PCCMPV Probabilistic Cooperative Caching at Moving and Parked Vehicles.

PCPV Proactive Caching at Parked Vehicles.

PCPV-GD Proactive Caching at Parked Vehicles-Gradient Descent.

PCPV-PSO Proactive Caching at Parked Vehicles-Particle Swarm Optimization.

PPCF PPCF.

PSO Particle Swarm Optimization.

PSO-LSTM Particle Swarm Optimization-Long Short-Term Memory.

QoS Quality of Service.

RCS Road Caching Spot.

RMM Regional Markov Model.

RNN Recurrent Neural Network.

RSU Road Side Unit.

SVM Support Vector Machine.

TPP Table of Possible Providers.

TTL Time-To-Live.

V2I Vehicle-to-Infrastructure.

V2V Vehicle-to-Vehicle Communication.

VANETs Vehicular Ad Hoc Networks.

VLMC Variable-Length Markov Chain.

VOPC Vehicular Optimal Proactive Caching.

VOPC-GD Vehicular Optimal Proactive Caching-Gradient Descent.

VOPC-PSO Vehicular Optimal Proactive Caching-Particle Swarm Optimization.

WAVE Wireless Access in Vehicular Environments.

XXDSC XXDice Similarity Coefficient.

Chapter 1

Introduction

1.1 Motivation

The penetration rate of social media has been significantly increasing worldwide. In 2019, 79% of Internet users were social media users [1]. This substantial growth is expected to escalate even further in the future, with an anticipated augmentation in the number of social media users reaching 3.1 billion in 2021 [2]. Mobile devices account for more than 60% of such excessive usage [3]. This causes backhaul links in Fifth Generation (5G) networks to incur a huge traffic influx [4]. Note that 5G promises several attractive prospects, including significantly high bandwidth, fast connectivity, and low latency [4]. However, the limited transmission capacity affiliated with the wireless backhaul links makes it difficult to cope with the explosively growing traffic [4]. Thus, it is imperative to reduce the load at backhaul links in order for 5G to keep its promises, and prevent the deterioration of the Quality of Service (QoS).

One promising solution to alleviate the aforementioned problem is to use Vehicular Ad Hoc Networks (VANETs) as a traffic offloading platform, where the pervasive availability, mobility, and abundant storage resources of vehicles can be exploited to

perform edge caching [4]. In edge caching, content access occurs via intermediate nodes at the edge of the network rather than the far-away original data provider at backhaul links (i.e., the data center/server) [4]. Note that caching social media traffic can be particularly beneficial since there is a high correlation in the data accessed by many users on social media. This is attributed to the fact that the most followed accounts on social media platforms (e.g., Instagram) are those of public figures, such as politicians, actors, musicians, etc. Thus, increasing cache hits of such contents in VANETs (i.e., data acquisition from caching vehicles) can reduce the backhaul traffic load and bring the data closer to the requester [4].

VANETs have emerged as a communication paradigm that fosters inter-vehicle communication on the road. They serve as an enabling technology for an extensive range of applications in Intelligent Transportation Systems (ITS), including Internet access [5]. Internet access in VANETs can take place via Vehicle-to-Vehicle (V2V) communication, as well as communication between vehicles and roadside access points, commonly referred to as Road Side Units (RSUs) [5, 6]. Along with their communication capabilities, vehicles and RSUs are also equipped with storage resources, which enable them to be used as caching units [7]. The static nature of RSUs can help stabilize the caching decisions. However, sheer deployment of RSUs might not always be plausible, since they usually require large investments [5, 8]. Thus, relying on RSUs for caching or communication might not always be feasible.

As a result of the infeasibility of sheer deployment of RSUs, V2V communication tends to be the predominant type of communication relied upon by Internet users to reach the closest RSU [5]. However, V2V communications intended for remote content providers are often coupled with high delay and low packet delivery ratio [5]. This is

due to the intermittent connectivity and highly dynamic topology of VANETs [5, 7]. In addition, a request-response data access model is typically adopted in Internet services in VANETs [7]. In such a model, the user sends a content request to the data center, accessed via an RSU, and the latter issues a response back to the requester [7]. This could further exacerbate the QoS. Thus, it is imperative to take the QoS into consideration, by selecting nodes closer to the requester for instance, when making caching decisions for social media access in VANETs.

Existing caching schemes are classified into two categories; proactive and reactive. In the former, data is prefetched and stored ahead of time, while in the latter, caching occurs as the data propagates back to the requester in response to a previous request [9, 10]. Proactive caching can significantly improve the QoS compared to reactive caching [10]. However, most existing proactive caching schemes in VANETs either employ a broadcast-based approach or focus on large-sized content download where each content is too large to be acquired at once, thus triggering the need for multiple transmissions [11]. The former are rendered unsuitable for social media access applications due to the huge amount of overhead incurred [10]. The latter are not suitable for the targeted social media platforms that do not offer large-sized contents, thus requiring single transmissions, such as Instagram. In contrast, most reactive caching schemes in VANETs are suitable for the targeted applications. However, they mostly adopt a non-cooperative caching approach, where caching decisions are made in an autonomous manner without any form of coordination between the nodes [6]. Such a non-cooperative caching approach tends to reduce cache hits compared to cooperative caching [12].

Cooperative caching has been recognized as an advantageous technique for enhancing the performance of content access in different network paradigms, including Mobile Ad Hoc Networks (MANETs) [12] and Information-Centric Networks (ICNs) [13]. In cooperative caching, caching decisions are made in a collaborative manner between the nodes [12]. Cooperative caching has been shown to bring the data closer to the requester, create increased data diversity, and achieve efficient utilization of the nodes' cache capacity [12, 13], which in turn increases cache hits. However, despite its demonstrated leverage in MANETs and ICNs, cooperative caching within VANETs has been mostly overlooked [12]. This is due to the highly dynamic nature of vehicles. This dynamic nature can shorten the lifetime of the exchanged cached content information and lead to unstable caching decisions. Another challenging issue in caching, that is typically overlooked in most caching schemes in VANETs, is the design of a rigorous policy for making efficient cache discovery decisions [12]. A cache discovery decision involves determining where the requested replica is located, and it is the first step that should be executed when a requester generates a content request [12].

The work presented in this thesis proposes a caching framework that addresses the aforementioned challenges. Note that although we focus on social media access applications, this work can be applicable to any type of applications involving the delivery of contents that possess the following characteristics: 1) non-real-time contents, 2) delay-tolerant contents that do not have a high priority, and 3) highly popular contents that tend to overload the backhaul links in 5G networks, thus caching them would be beneficial. In the remainder of this chapter, we present the research statement and objectives in Section 1.2, as well as the thesis main contributions in Section

1.3. Finally, the thesis outline is highlighted in Section 1.4.

1.2 Research Statement

In this thesis, we investigate the use of edge caching in VANETs to curtail the traffic load at backhaul links in cellular networks. Our main objective is to maximize cache hits and improve the quality of VANET-based Internet services, particularly in terms of delay and packet delivery ratio. Towards that end, we address the following research questions:

1. Can we exploit the static nature of roadside parked vehicles to reinforce the use of caching within VANETs?

2. Can the fact that some users tend to establish a rather predictable daily driving routine and social media access behavior be used to pre-cache the data at roadside parked vehicles for users to proactively procure as they pass by? Can we achieve a certain QoS demanded by users?

3. How can we predict the spatiotemporal availability of replicas given the various influential factors affecting the prediction accuracy, such as the weather and traffic conditions?

4. For users that do not exhibit such a predictable behavior, can cooperative reactive caching be leveraged within the context of VANETs? Can we devise an effective cooperative cache placement technique given the extreme dynamic nature of VANETs?

5. Can we design a cooperative cache discovery technique that helps expand the search space of the requested replicas within the context of VANETs? Can the

trajectory of caching vehicles be predicted to increase the search space even further?

6. How to reduce the overhead associated with cooperative caching due to the exchange of cached content information?

We believe that predictive proactive caching can serve users who have a rather consistent and predictable behavior in a way that can significantly increase cache hits and improve the quality of VANET-based Internet services. We also believe that cooperative reactive caching can be an extremely promising alternative for users who do not demonstrate such a predictable behavior. We claim that the use of parked vehicles can positively contribute to the caching process.

Note that in contrast to RSUs, roadside parked vehicles are natural infrastructures that exist in substantial number. Hence, they can serve as a source of profuse and cost-effective caching resources. In fact, it has been substantiated in a study that examined roadside parking spaces in Ann Arbor city in the US that their occupancy ratio can mount up to 93% and 80% during on and off-peaks, respectively [7]. Rechargeable batteries, especially in electric vehicles can highly facilitate the utilization of roadside parked vehicles for caching. It has been demonstrated that in such vehicles, the average time beyond which the battery gets drained is 160 hours [7].

Our objectives include the following: (1) designing a prediction module that can increase the prediction accuracy of the travel time of requesters on each road segment along their trajectory, in order to make informed proactive caching decisions, (2) proposing a prediction-based proactive cache placement module that can allocate replicas to roadside parked vehicles so as to maximize cache hits, while maintaining the QoS demanded by users, (3) introducing a benchmark that can quantify the potential gains of predictive proactive caching in improving the quality of Internet

services in vehicular networks, (4) proposing a cooperative cache discovery module that can expand the search space to detect data holders that are within closer proximity to the requester, (5) presenting a cooperative cache placement module that can extend the cooperation range among vehicles to cache more diverse data and increase cache hits, and (6) reducing the overhead incurred due to the exchange of cached content information in cooperative caching.

1.3 Thesis Contributions

The contributions of this thesis are as follows:

1) *Travel Time Prediction Module:* This module addresses Objective 1. The data center has a prediction module that enables it to predict the period of encounter of the requesting vehicles with each road segment along their trajectory. We mainly focus on the prediction procedure of this period due to the influential effect of various factors, including weather and traffic conditions on the prediction accuracy. Thus, we propose a novel prediction scheme that incorporates the use of a Long Short-Term Memory (LSTM) network, trained using particle swarm optimization (PSO-LSTM) to improve the training process. We also take the weather conditions, the time of the day, and the day of the week into consideration when training the model in order to increase the prediction accuracy.

2) *Predictive Proactive Cache Placement Module:* Objectives 2 and 3 are handled by designing the Vehicular Optimal Proactive Caching (VOPC) benchmark and the Proactive Caching at Parked Vehicles (PCPV) greedy heuristic scheme. In both VOPC and PCPV, the data center exploits the daily driving routine and predictable behavior of users. It does so to pre-cache the data at parked vehicles for the requesters

to proactively acquire either before the time of their requests or within a specified time frame. The objective is to maximize cache hits by assigning replicas to caching spots that yield maximum certainty in their spatiotemporal availability for requesters within certain time restrictions. This is while sustaining a cache capacity limit. VOPC formulates the caching problem as an integer linear programming (ILP) optimization problem and can thus act as an upper bound on reachable potential.

3) *Cooperative Cache Discovery Module:* In order to address Objective 4, we propose two tracking-based cooperative cache discovery schemes; the Cooperative Content Discovery (CCD) scheme, and the Prediction-Assisted Cooperative Content Discovery (PACD) scheme. Tracking-based schemes involve some form of information exchange among nodes to track the cached contents, which can thus expand the search space, and consequently increase cache hits [12]. This is since such cached content information can be used to navigate request packets towards nearby caching nodes rather than blindly directing them towards the far-away data center. In CCD, we expand the search space beyond the typical neighborhood scope by relying on beacon messages that are exchanged periodically between neighboring vehicles, as well as the mobile and static nature of moving and parked vehicles, respectively, to diffuse cached content information within the network. CCD then uses location-based breadcrumbs to track the next road segments to which caching nodes are planning to traverse, and dynamically detect closer caching nodes to the requester. In order to further expand the search space, we propose PACD, where a novel vehicles trajectory prediction scheme is designed to predict the location of mobile caching nodes. To the best of our knowledge, CCD is the first tracking-based cache discovery scheme within VANETs that is extended beyond the neighborhood scope, and PACD is the first one to use

prediction, thus expanding the search space beyond restrictions.

4) *Cooperative Cache Placement Module:* In order to achieve objective 5, we propose the Probabilistic Cooperative Caching at Moving and Parked Vehicles (PCCMPV) scheme. To the best of our knowledge, PCCMPV is the first cooperative cache placement scheme in VANETs that caches the data at both parked and moving vehicles, and enables the extension of the cooperation range, thus making more informed caching decisions. Such an extension is facilitated by sending cached content information of different nodes from parked to moving vehicles, and vice versa, via beacon messages. In PCCMPV, we populate valuable road segments with diverse cached data to increase cache hits. In addition, we exploit the trajectory of moving vehicles to also apply an implicit form of off-path caching, thus increasing the range of potential caching nodes.

5) Information Exchange with Reduced Overhead: In order to address objective 6, we adopt the use of bloom filters. A bloom filter is a simple and spatially-efficient randomized data structure [14]. It is typically used to parsimoniously represent a set of elements by catering for membership queries without taking too much space [14]. We use bloom filters during the cached content information exchange process associated with cache discovery and placement, which significantly reduces overhead.

1.4 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 presents some background and related work. We provide a review of the existing reactive and proactive caching schemes. Also, since most reactive caching schemes in VANETs follow a non-cooperative caching approach, we review the existing cooperative caching techniques

in other network paradigms, such as MANETs and ICNs. We provide a detailed discussion of both cooperative cache placement and cache discovery schemes to understand which characteristics need to be leveraged and which should be overlooked when designing cooperative caching schemes within the context of VANETs. We present our predictive proactive caching framework in Chapter 3. In this chapter, we introduce our proposed prediction module, as well as our proactive cache placement module, including the ILP formulation and greedy heuristic scheme proposed to solve the proactive cache placement problem. Furthermore, the performance evaluation results comparing the optimal and heuristic solutions are provided in this chapter. In Chapter 4, we discuss the cooperative cache discovery module, including the two proposed schemes CCD and PACD. The proposed vehicles trajectory prediction mechanism used in PACD is also presented in this chapter. Chapter 5 discusses the proposed cooperative cache placement module. We also provide a detailed discussion of the performance evaluation of the proposed cooperative caching solution. This involves each of the two cache discovery schemes presented in Chapter 4, along with the cooperative cache placement scheme. Finally, we highlight the conclusions of our work and outline some potential future directions in Chapter 6.

Chapter 2

Background and Overview

2.1 VANETs Overview and Characteristics

In VANETs, vehicles can exchange information with each other and/or with roadside units (RSUs) [15]-[20]. The former is referred to as Vehicle-to-Vehicle communication (V2V), while the latter is referred to as Vehicle-to-Infrastructure communication (V2I) [17, 21]. In an attempt to reinforce V2V and V2I communications, standards related to wireless access technology in vehicular environments have been developed [15, 16]. One of the most important of such wireless access technologies is the Dedicated Short Range Communications (DSRC) [17, 18]. The DSRC standardization effort has been amended by the IEEE 802.11 and the IEEE 1609 standard groups, which changed its name to IEEE 802.11p Wireless Access in Vehicular Environments (WAVE) [17, 18]. Recently, Cellular Vehicle-to-Everything (C-V2X) technology has been standardized to support V2X services [19]. It is worth mentioning that the work proposed in this thesis is applicable regardless of the underlying vehicular communication technology.

The notion of supporting wireless communication in smart vehicles has captured researchers' attention since the 80's [15]. Note that smart vehicles are the main

facilitators of Information Transportation Systems (ITS) [15]-[18]. ITS can streamline a wide range of applications, including traffic safety (e.g., sending warning messages from the scene of an accident to other vehicles on the road), traffic management (e.g., providing real-time traffic conditions), and infotainment applications (e.g., Internet access) [17, 19, 20]. In the last decade, research, standardization, and development in the area of wireless communication in smart vehicles have gained vigorous momentum. This can be attributed to the extensive endorsement of IEEE 802.11 technologies, as well as the acknowledgment of various governments, industries, and institutes of the crucial role that vehicular technology can play in enabling ITS [15]-[18].

VANETs are composed of two intrinsic components, namely on-board units (OBUs) and RSUs [17, 18, 22]. An OBU is a hardware device that resides on-board smart vehicles to enable them to communicate with other vehicles [18]. A smart vehicle is typically equipped with an OBU and a tremendous number of sensors for capturing and processing information about the surroundings [17, 18]. An OBU can resemble a personal computer in terms of its storage and computational capabilities [17, 18]. In addition, an OBU consists of a user interface and a communication module [18]. An RSU is a stationary device that is typically located along road sides, near parking spaces, or at intersections [18]. RSUs are equipped with a communication module for communicating with vehicles as they pass by. In addition, RSUs can be connected to the Internet or other servers, thus facilitating Internet access to vehicles. For this purpose, RSUs are also equipped with communication modules based on different radio technologies to enable communication within the infrastructure network [17, 18].

VANETs are characterized by a number of unique characteristics that distinguish them from other networks. Such characteristics are summarized as follows [18]:

- Predictable Mobility: The movement of vehicles is restricted by road topology and layout, as well as the necessity to abide to road signs, traffic lights, and traffic rules. Thus, the next position of a vehicle can be more easily predicted compared to MANETs, where nodes' movements are arbitrary, making it harder to predict their future positions.

- Highly Dynamic Topology: The fast and rapidly changing movement of vehicles makes the network topology more apt to extremely rapid and frequent changes.

- Intermittent Connectivity: In VANETs, network connectivity and link lifetime between vehicles are highly affected by the dynamic changes in the topology. In addition, network connectivity can be rigorously affected by the scarce or low vehicular density. The fast variations in link connectivity can cause many routes to be disconnected before they can even be used.

- Sufficient Power, Storage, and Computational Resources: Nodes in VANETs are vehicles rather than handheld devices, which is why power is not a critical issue, since energy is continuously provided to vehicles via the long life battery. In addition, vehicles tend to have abundant resources in terms of storage and computational capacity.

- Bandwidth Limitations: Due to the short range of bandwidth frequency (10-20 MHz) used in vehicular scenarios, data dissemination latency is affected by the fair use of bandwidth. This is particularly true when the density of vehicles is high, and thus more nodes contend for the same resources.

Along with the aforementioned characteristics, another important ingredient in VANETs is the way in which data access occurs. In particular, in social media access

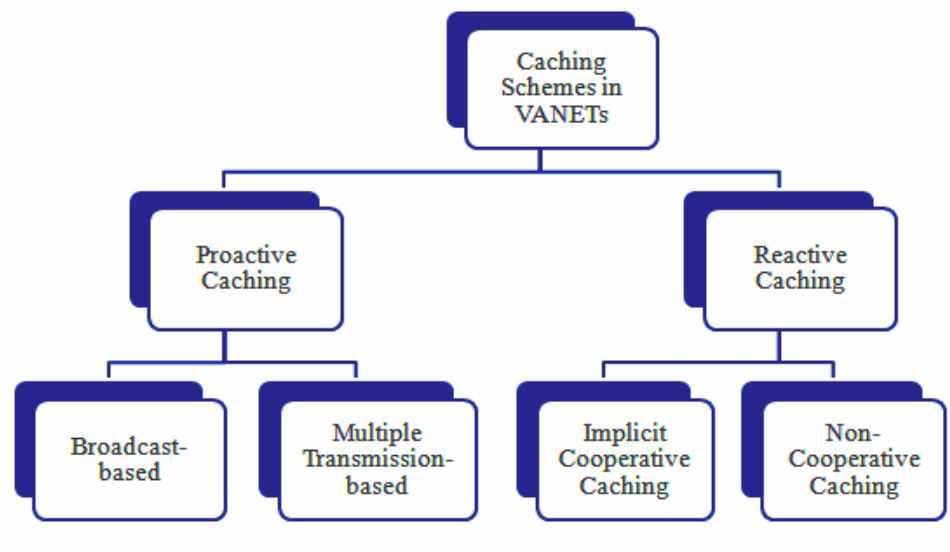


Figure 2.1: Taxonomy of caching schemes in VANETs.

applications in VANETs, data access is pull-based [17]. That is, a request targeting the data center is sent to an access point (i.e., RSU), and a response is sent back to the requester. This process is often associated with high delay and low packet delivery ratio due to the highly dynamic topology and intermittent connectivity of VANETs [5, 10]. By the time the data is issued back to the requester, its position might be significantly changed [5]. Thus, the packet might be dropped if the requester cannot be tracked [5]. This further affects the QoS. One way that can help improve the QoS and reduce the load at backhaul links is to incorporate the use of caching. In the next sections, we provide a review of the existing caching schemes in VANETs. A taxonomy of such schemes is depicted in Figure 2.1.

2.2 Caching in VANETs

Caching schemes in VANETs can be classified into proactive caching schemes and reactive caching schemes.

2.2.1 Proactive Caching in VANETs

In proactive caching, the data center caches the data at vehicles by prefetching the requesters contents of interest to be cached ahead of time [23]. This helps reduce latency and alleviate the traffic load at backhaul links [23]. An example of proactive caching schemes in VANETs is proposed in [24], where the data provider periodically broadcasts the data to all the vehicles within its transmission range to be cached. Each vehicle can then exchange the cached data with other vehicles upon encounter. Thus, users can acquire the data they are interested in only if they encounter another vehicle that happens to have it. To increase data availability, the authors in [10] propose a proactive caching scheme where the data provider periodically broadcasts the data to all the vehicles in highly dense roads and at intersections.

These proactive caching schemes eliminate the need for an infrastructure, which makes them cost-effective and suitable for infrastructureless vehicular environments. However, severe congestions can occur due to excessive transmissions, particularly in dense networks [10]. In addition, they are often considered unsuitable for user-specific applications, such as social media access applications. This is since in such applications, as opposed to safety applications, the requested content varies from one user to another, so the flooded contents are not of interest to all users [10]. Furthermore, they trigger broadcast storms and lead to bandwidth inefficiency [25]. The aforementioned reasons lead to their infeasibility for applications that have huge amounts of content,

such as social media applications. The scheme in [23] has been used as a baseline by other broadcast-based proactive caching schemes, such as [25] and [26], with more emphasis on resolving the issues of broadcast storms and bandwidth inefficiency.

Proactive caching has also been investigated for downloading large contents in VANETs. Such contents cannot always be obtained all at once because of the short connectivity period between vehicles and infrastructures [27]. Thus, such schemes divide the data into smaller chunks and aim at minimizing the download time of all the chunks constituting the entire content. Note that in large contents, multiple requests are typically issued by the requesters to fully retrieve the data. In [27], rather than sending a request for each chunk, a vehicle only sends a request for the first chunk, associated with information about its location, speed, and frequency of interests to the data provider. The data provider then uses a mobility prediction scheme to cache the remaining data chunks at specific RSUs that are expected to be in close proximity to the requesting vehicle. Thus, the number of data transmissions is reduced. However, the incorporated mobility prediction scheme in [27] is highly unreliable as it predicts the future position of vehicles based on their current position and speed. Thus, it assumes that vehicles are moving in one direction and at a constant velocity [27]. In addition, RSUs require very large investments, so they might not be densely deployed [7].

Recently, some schemes, such as [11, 28, 29], have proposed the use of roadside parked vehicles for downloading large contents. These schemes divide the data into smaller chunks and use the sequential nature of parked vehicles to store these chunks. In [11, 28, 29], only an initial interest is issued and the remaining data is prefetched to reduce the download delay. However, most of the schemes that adopt the use of parked

vehicles are more concerned with content downloading, rather than caching itself [11]. In [11], a requesting vehicle sends a content download request to the nearest parking cluster. The parked vehicle located at the entry of the road segment is selected as the cluster head (CH). The CH communicates with other parking clusters to download the remaining parts and have them ready for the requester. The authors assume that parked vehicles are connected to the content provider via the Internet. This incurs additional costs for parked vehicles. In [28], each moving vehicle sequentially uploads its content chunks as it passes by parked vehicles that form line clusters. This is done without considering the possible trajectories or needs of the requesters. Thus, acquiring the data relies on opportunistic encounter with line clusters that happen to have the data. In [30] and [31], RSUs have been used for storing content chunks, while modeling the mobility patterns of vehicles. In [30], such mobility patterns have been modeled as a single highway, while a Manhattan mobility model following a more realistic mobility assumption has been adopted in [31]. However, the authors assume a constant velocity of vehicles and disregard the unstable conditions that could affect their mobility.

Most existing proactive caching schemes in VANETs adopt a broadcast-based approach, which renders them unsuitable for social media access applications [5, 10]. Other schemes are designed to accommodate the download process of large-sized contents that require multiple transmissions, with the necessity to wait for an initial request to be issued before an action is taken. In this thesis, we propose a predictive proactive caching module that takes the trajectory of vehicles, as well as their request patterns into consideration in order to enable caching to occur ahead of time. The proposed module is suitable for social media access applications that require single transmissions.

2.2.2 Reactive Caching in VANETs

In reactive caching, the data is cached as it propagates back to the requester in response to a previous request. In general, there are two types of reactive caching techniques, namely cooperative and non-cooperative caching. In non-cooperative caching, caching decisions are made in an autonomous manner that does not involve any kind of coordination between the nodes [32]. In cooperative caching, two or more nodes either form a cooperative caching environment or make caching decisions in a collaborative manner [32]. This enables the sharing and coordination of cached data between multiple nodes in order to improve data availability and enhance the overall caching performance [5, 13, 32].

Most existing reactive caching schemes in VANETs typically employ the use of non-cooperative caching. In [33], the authors propose a non-cooperative caching scheme, where all moving vehicles along the data forwarding path cache the data. This significantly increases wasted cache space and reduces data diversity, which may lead to reduced cache hit ratio, particularly with sparse cache space and frequent cache replacements [9]. In [9], the authors strive to improve caching efficiency and content diversity in the network by proposing the Distributed Probabilistic Caching (DPC) scheme. In DPC, caching occurs at moving vehicles by considering three factors; data popularity, the degree and betweenness centrality of the vehicle in the ego network, as well as the relative direction of movement between the data provider and consumer. DPC has been shown to improve network performance compared to the scheme in [33]. However, the highly dynamic nature of vehicles can cause the calculated centrality to be largely unstable, which can affect the efficiency of caching decisions. In addition, the lack of any form of cooperation between the nodes can

lead to inefficient utilization of the nodes cache capacity.

In [34], a caching scheme that relies on global information of the network topology is proposed. In such a scheme, caching occurs at moving vehicles by applying the use of the minimum vertex cover set theory. The minimum vertex cover set theory is used to cover the entire network with the minimum number of caching nodes, such that each node can acquire the data in one hop. However, this requires acquiring the topology information of the entire network. Thus, caching decisions based on minimum coverage set are more suitable for networks with a static topology so as to avoid frequent updates of the global topology information. In networks with a highly dynamic topology, such as VANETs, this global cache node selection scheme tends to trigger significantly high maintenance costs.

In [35], the authors propose the Caching-Assisted Data Delivery (CADD) scheme, which implicitly inherits some features of cooperative caching. In CADD, caching occurs only at static nodes, called Road Caching Spots (RCSs), deployed at intersections. Using an implicit form of collaboration between on-path RCSs, the RCS that receives the highest number of requests is dynamically selected for caching. Such an implicit cooperation improves cache hit ratio. However, since caching decisions occur without any exchange of cached content information between RCSs, the problem of increased wasted cache capacity still persists, thus cache hit ratio can still be affected.

The lack of an explicit exchange of cached content information in the aforementioned schemes can lead to increased redundancy (i.e., caching the same content at nodes that are within close proximity), and inefficient utilization of the available caching resources, which can reduce cache hits [13]. In [36], an explicit cooperative caching scheme where cache placement decisions are made at RSUs, is employed.

However, the cooperation range is restricted to the neighborhood scope only, which limits the extent of data diversity. In addition, the authors ignore the caching resources of vehicles and rely solely on RSUs, which are not densely deployed.

Most existing reactive caching schemes in VANETs are susceptible to increased data redundancy and wasted cache space, thus reduced cache hits. This is due to the lack of explicit cooperation and cached content information exchange between nodes. In this thesis, we propose to take advantage of the benefits of explicit cooperative caching within the context of VANETs, with an extended cooperation range that goes beyond the neighborhood scope. Also, as opposed to most existing caching schemes in VANETs that exploit the caching resources of either moving vehicles only or RSUs only, we leverage the caching resources of both parked and moving vehicles.

Cooperative caching has been recognized as an advantageous technique for enhancing the performance of content access in different network paradigms, including MANETs (Appendix A) [12] and ICNs (Appendix B) [13]. However, despite its substantiated leverage [12, 13], cooperative caching has been rarely investigated within the context of VANETs. This can be attributed to the extremely dynamic nature of vehicles, which shortens the lifetime of the exchanged cached content information, since they tend to rapidly become obsolete as vehicles quickly change their positions. We believe that it is possible to take advantage of the benefits of cooperative caching within the context of VANETs. To do so, it is important to study existing cooperative caching techniques in other network paradigms, such as MANETs and ICNs. This can help identify the features of cooperative caching that can be leveraged and the ones that should be completely avoided when applied within the context of VANETs.

The state of the art cooperative caching techniques in literature handle one or both of two cooperative caching components, namely cache discovery and cache placement [12, 13]. A taxonomy of such cooperative caching techniques in MANETs and ICNs

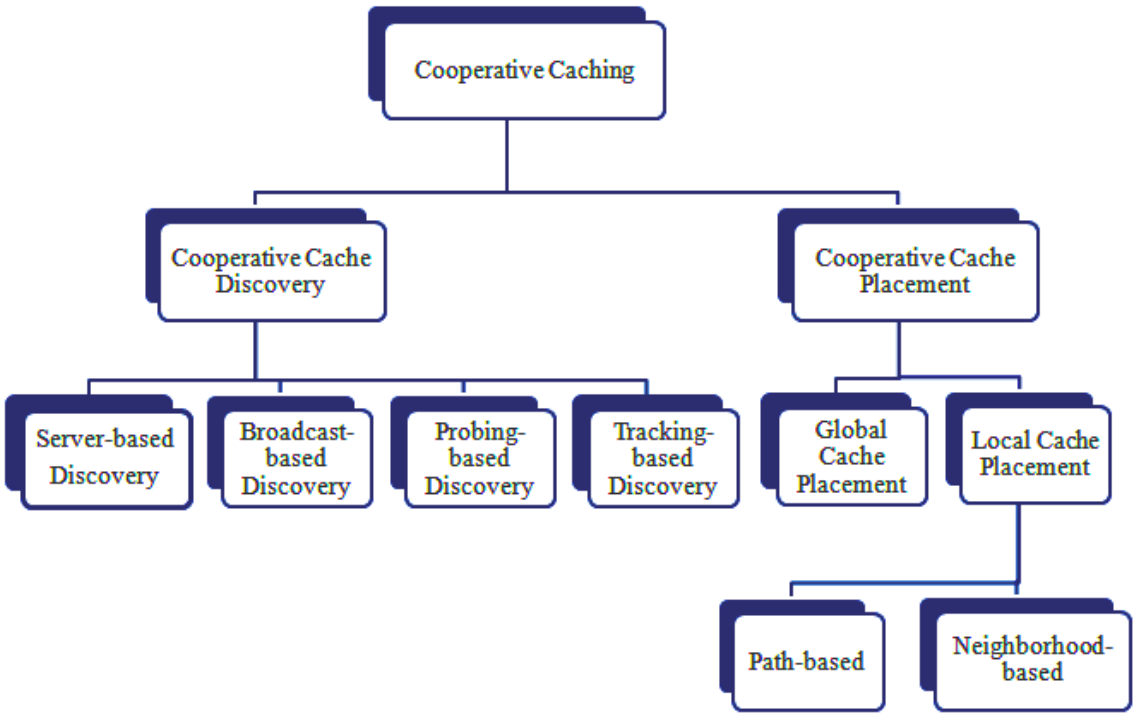


Figure 2.2: Taxonomy of cooperative caching schemes in MANETs and ICNs.

is illustrated in Figure 2.2.

Note that the cache discovery component is concerned with locating the nearest copy of the cached data. Discovering where the data is cached is the first procedure that needs to be performed when a node issues a request [12]. In the cache placement component, decisions pertaining to where to cache the data and which data to cache are made [12, 13]. In the following sections, we provide a detailed discussion of each of these components in MANETs and ICNs.

2.3 Cooperative Cache Discovery

Various cooperative cache discovery schemes have been designed for MANETs and ICNs in the literature. Such schemes can be classified into broadcast-based, probing-based, server-based, and tracking-based schemes [12, 13]. In the next sections, we provide a detailed discussion of some of the existing schemes in each category.

2.3.1 Server-based Cache Discovery

In this type of schemes, requests are directed to the original data provider (i.e., the server) and the caches of intermediate nodes are consulted for a matching replica. For example, as depicted in Figure 2.3, node A generates a request for the data item d_1. It first checks its own local cache for a matching replica. Upon local cache miss, A sends the request to the server. As the request packet propagates along the request forwarding path, each intermediate node checks whether it has the data in its cache. If not, it forwards the packet to the next node along the path. This is repeated until the server or a caching node is reached. When node E receives the request, it checks its cache, finds a match for d_1, and issues a reply back to node A. This approach does not yield additional overhead as it does not require the transmission of extra messages [12]. However, it significantly limits the search space since it relies on finding the cached data at intermediate nodes encountered by the request packet en route to the server [12]. It offers no guarantee that any of those nodes has the cached data. Thus, the request could eventually be satisfied by the far-away server, which results in increased delay [13].

Cache Data [37], which is proposed in MANETs, represents the basic server-based approach. Requests for contents are directed from the requesting node towards the

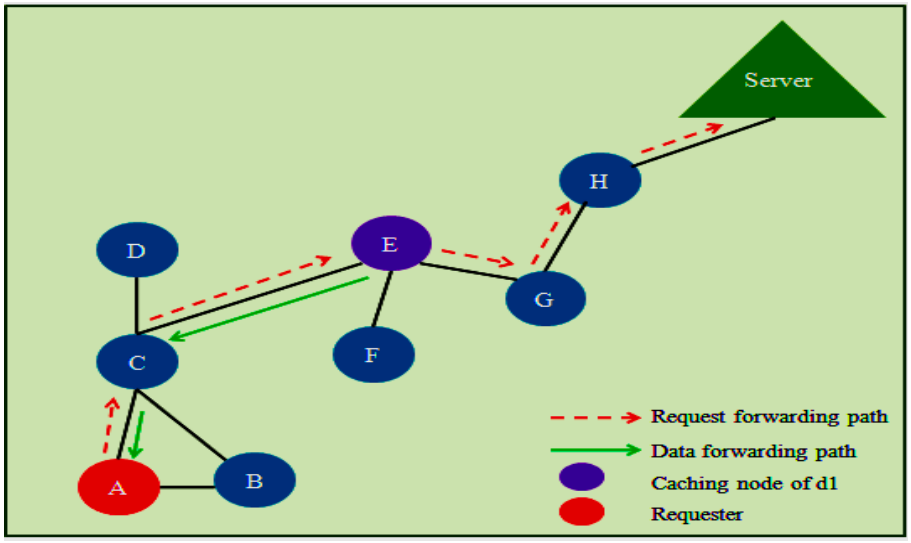


Figure 2.3: Server-based cache discovery.

server. As the data packet propagates back to the requester, intermediate nodes cache the data that are frequently requested. Subsequent requests for the same data can then be satisfied by intermediate caching nodes. However, due to the mobility of nodes, the probability of opportunistic encounter with caching nodes might be significantly affected, causing requests to be served by the far-away server [37, 38].

Cache Path [37] extends the Cache Data approach in MANETs by enabling data forwarding nodes to cache the path to caching nodes. When a forwarding node receives a data packet, if the packet destination is closer to it than the original data provider (i.e., the server), it caches the path to the destination. Subsequently, if this forwarding node requests the data itself or receives a request for the data from another node, it can simply navigate the request along the stored path of caching nodes. This saves cache capacity compared to the Cache Data approach [37, 39]. However, due to the dynamic nature of MANETs, the stored paths could frequently become obsolete

[39].

In Hybrid Cache [37], also proposed for MANETs, intermediate nodes en-route to the server selectively use Cache Data or Cache Path in order to leverage the benefits and avoid the weaknesses of each scheme. For a forwarding node to decide which scheme to apply, it uses the data size, the Time-to-Live (TTL) of the data, and the potential savings yielded from caching the path. Cache Data is used if the data size or its TTL is small. If TTL is small, caching the path would not be efficient, since the data would not remain valid for a long time, thus the path to the data would soon be invalidated. In contrast, if the data size or its TTL is not small, Cache Path is used. Also, if a large number of hops is saved by using Cache Path, it might be better to use it. The problem with these schemes is that the caching information of a node cannot be explored if it is not on the forwarding route to the server [37]-[41].

The aforementioned basic server-based cache discovery approach is used in many schemes in ICNs, including [42]-[46]. In ICNs, the server-based approach is referred to as the opportunistic forwarding mechanism. Such an approach is considered the default strategy used by most caching schemes in ICNs [13].

The dynamic INterest FORwarding Mechanism (INFORM) [47] is another server-based cache discovery scheme used in ICNs. INORM applies a dynamic forwarding mechanism based on the Q-learning algorithm [48]. In INFORM, when a forwarding node receives a request packet, and if the requested content is locally unavailable, it forwards the packet over the shortest path towards the original data provider and the cache of each intermediate node is consulted in a hop-by-hop basis. In addition, the forwarding node forwards a copy of the request on a randomly selected interface. This is done during an exploration phase to discover content replicas, as well as the

Q values of interfaces in order to determine the best interface yielding the minimum delay. Adding the interest random walk feature to the basic server-based approach enables the exploration of a wider search space. However, such a random walk may eventually end without encountering a caching node, particularly since it is limited by a certain number of hops.

2.3.2 Broadcast-based Cache Discovery

In this type of schemes, requests are flooded to locate the cached data. Once the recipients of the broadcasted messages receive the request, they check to determine whether the requested data is stored in their local cache. If it is, the data is sent back to the requester. If not, the request is re-broadcasted. For example, as depicted in Figure 2.4, node A is interested in data item d_1. It first checks its own local cache for a matching replica. Upon local cache miss, A broadcasts the request. Since neither B nor C has d_1 in the local cache, they rebroadcast the request. When C does that, the request reaches all of its neighbors. Since both D and E have d_1 in their cache, they both issue a data reply back to A.

This approach expands the search area compared to the server-based approach, and thus expedites the discovery process. However, it can be cost effective in terms of bandwidth and can have a severe impact on the overall traffic load in the network [12], particulary in cases when there is a huge amount of contents, thus prompting sheer number of requests. Consequently, most schemes utilizing this approach tend to use limited flooding [12, 13]. A typical form of limited flooding restricts the propagation of the request packet to a specific number of hops [12, 13].

The Split Cache scheme proposed in [49] employs the basic form of limited flooding

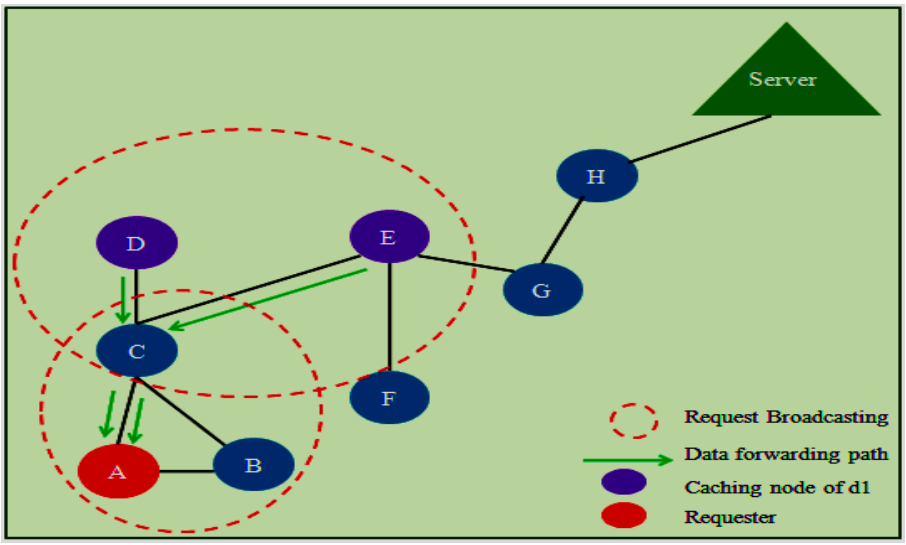


Figure 2.4: Broadcast-based cache discovery.

in MANETs. In Split Cache, a hop count parameter is included in the request packet when it is broadcast so as to limit the range of the request. This alleviates the aforementioned problems. However, limited flooding still has several drawbacks [12]. For instance, if none of the nodes receiving the request has the requested data in its cache, either the request is dropped or the requester waits for the timeout before it resends the request to the server [12]. This causes an increased latency.

Two broadcast-based schemes in MANETs are investigated in Hamlet [50], both of which adopt the limited flooding approach. These schemes are referred to as Mitigated Flooding and Eureka [50]. In the former, the basic form of limited flooding, which incorporates the use of a hop count parameter, is adopted. In Eureka [50], a network steering capability is added. Such a capability enables routing the data towards parts of the network where there is a higher probability that the data will be found. To do that, requests are steered within the network using the concept of information

density. This reduces delay and increases packet delivery ratio compared to Mitigated Flooding [12, 50].

Another MANET-based scheme that relies upon limited flooding is the Shared Cache [51]. In this scheme, nodes are organized into clusters. When a node requests a data packet, it first checks its local cache. In case of a cache hit, the data is returned. Otherwise, the node broadcasts the request within the cluster. If the data is available in the cache of a cluster member, it is sent back to the requester. If not, upon overhearing the cache miss, the cluster head forwards the request packet to the server. This reduces the delay encountered in the basic form of limited flooding when a requesting node is forced to wait for a timeout before resending the request. However, due to the dynamic topology of the network, maintaining these clusters can also lead to increased overhead, particularly in dense networks and in cases when the cluster head gets disconnected or move away [12, 51].

In [52], an ICN-based approach that uses limited-flooding for cache discovery is proposed. In this scheme, rather than the hop-count parameter used in various ICN-based schemes, including [53], the receiver of a request packet determines whether or not to rebroadcast the packet by determining the redundancy of broadcasting. This occurs by identifying the sender's neighbors and determining if the neighbors of the receiving node are a subset of those of the sender. If so, the node will not rebroadcast the request packet since it has already been received by all of its neighbors. Thus, this approach reduces the number of packets sent in the network by avoiding sending redundant packets. However, it does not have the capability to direct the requests where contents are more likely to be found.

In [54], an ICN-based scheme that uses limited flooding based on a link-weight

parameter is introduced. In this scheme, the link-weight parameter serves the same purpose as the hop-count parameter in basic limited flooding. However, it avoids searching for the data where it is less likely to be found. This is done by taking into consideration that the caching policy identifies paths with low delay rates by avoiding congested links, and requires the data to be cached on the downstream end of a congested link. Accordingly, in the cache discovery employed in [54], the link-weight parameter used for limited flooding is the reverse of the link bandwidth. Thus, a request is flooded until a link with a bandwidth that is as low as the bandwidth specified is found.

2.3.3 Probing-based Cache Discovery

In this type of schemes, a node first determines the location of the requested data within the collective cache before sending a direct request to the discovered caching node [12, 13]. Thus, in case of a local cache miss, a node sends a broadcast message requesting to be notified by all the nodes that have the data. The requester then selects one of the replying nodes and sends a request directly to the selected node. Typically, the requester chooses the first node from which it receives a reply as it is considered the closest node [12]. For example, as shown in Figure 2.5, node A is interested in data item d_1. Upon local cache miss, it broadcasts a probing message to request the location of d_1. When B and C receive the probing message, they rebroadcast it. Accordingly, the probing message reaches the caching nodes D and E. They both respond by sending their location to node A. Once this information is received, A selects the closer node (i.e., D), and issues the request to it. Upon receiving the request, node D sends a reply back to A.

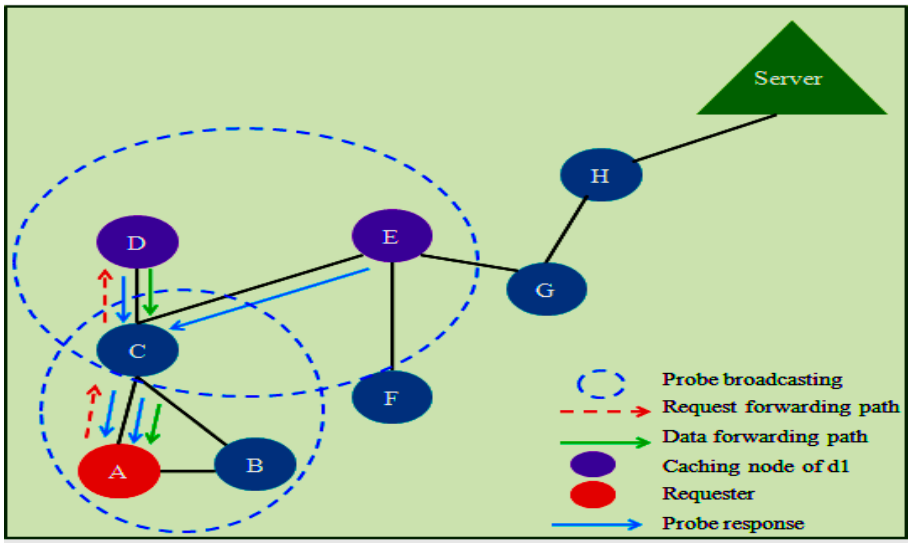


Figure 2.5: Probing-based cache discovery.

Probing-based schemes benefit from the expanded search space provided by the broadcast-based approach while minimizing the communication overhead typically incurred due to the multiple data copies received by the requester [12]. However, such schemes can cause increased delay compared to the broadcast-based approach [12].

Route Stability [55] is a probing-based scheme proposed in MANETs. In Route Stability, a query for the location of the requested data is broadcasted in the network. Upon receiving the query, the nodes check their local cache to determine if there is a match for the requested data. If so, a reply indicating the nodes address is sent back to the requester. Otherwise, the node rebroadcasts the query. In order to limit the range of the query, Route Stability calculates certain probabilities that indicate route stability. In each forwarding node, the probability of a particular route is checked. If the probability exceeds a certain threshold, the query is rebroadcasted. Otherwise,

the query is not rebroadcasted. This helps reduce the overhead associated with the probing process. However, there is no guarantee that the requested replica can be found along the routes that have higher stability.

Another probing-based scheme in MANETs is proposed in [56]. In this scheme, rather than broadcasting query messages to all neighbors, the nodes send them to the K most relevant neighbors that are considered more stable and less loaded, thus more suitable to satisfy the request. The rationale behind this scheme is that caching nodes may move away during the time between sending the probe reply and the requester issuing the direct request. Consequently, both the requester and responder may lose connectivity. In this case, it may be required to repeat the entire probing process, thus wasting more resources and causing higher delay. To alleviate this problem, the query is sent to certain neighbors that are not significantly loaded, and are stable enough to be able to satisfy the request. This is done based on a scoring function. However, the scoring function does not consider the probability that the selected neighbors actually have the data in their cache.

Stateful forwarding [57] is a probing-based scheme proposed in ICNs. In Stateful forwarding, nodes broadcast periodical probing messages on available interfaces in order to retrieve requested contents via the best route. Such periodic probing reduces the amount of traffic induced during the content discovery process. The best route is determined by ranking different paths based on whether or not delivery failures or congestions have been previously observed on them. Restricting the probing process to certain routes helps reduce the amount of overhead and delay associated with the probing process. However, it does not guarantee that the probing messages are directed towards the nodes that actually have the data.

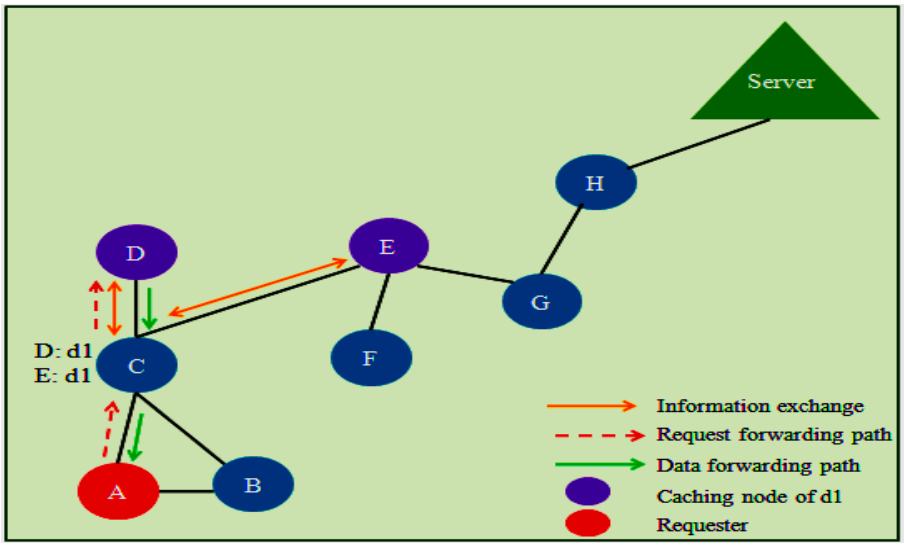


Figure 2.6: Tracking-based cache discovery.

2.3.4 Tracking-based Cache Discovery

Many existing cooperative cache discovery schemes depend on tracking cached contents [12, 13]. Such tracking information can be used to navigate requests towards nearby caching nodes rather than directing them towards the far-away server or blindly sending broadcast or probing messages. This type of schemes is referred to as tracking-based schemes [12]. For example, as depicted in Figure 2.6, all neighboring nodes periodically exchange their cached content information. Thus, node C receives information from D indicating that it has the data item d_1 in its cache. It also receives similar information from E. Thus, C now knows that d_1 is available at D and E. It saves such information for subsequent use. At a later time, node A becomes interested in d_1. Upon local cache miss, node A checks if it has any cached content information (tracking information) pertaining to d_1. If not, it sends the request along the request forwarding path to the server. Each intermediate node checks its own

local cache for a matching replica. If it does not have the data in its cache, it checks if it has any tracking information regarding the data. When the request reaches node C, and based on its knowledge that both D and E have the data, it directs the request to D, which responds by issuing a data reply back to A.

In Neighbor Caching [58], a tracking-based scheme is proposed within the context of MANETs. In Neighbor Caching, when a node encounters the need for a cached data item to be replaced, it coordinates with its neighbors to determine the best node to which the replaced data can be migrated. Once the migration occurs, the node keeps a reference to the neighboring node to which the data has migrated. In the future, if this node initiates or receives a request for the migrated data item, it forwards the request to the referenced neighboring node. Such a scheme can reduce the overhead associated with cached content information exchange. However, it restricts the tracking process to migrated contents only, which limits the benefits of the tracking process.

Another set of tracking-based schemes in MANETs depend on some form of information exchange for tracking cached contents [12, 59, 60]. In particular, they rely on nodes actively forming a group and periodically exchanging the status of their caches [12]. For example, in GroupCaching [59], one-hop neighbors are organized into a group. The members of each group periodically exchange information pertaining to their cached contents. Two tables are maintained by each group member; a self table and a group table. In the self table, a node maintains a list of its own cached data items. In the group table, a node maintains a list of the cached data items in each member of its group. Upon initiating or receiving a request for a data item, a node first checks its self table for a matching replica. In case of a cache miss, the group table is consulted. If the requested data is cached at one of its neighbors, the node

forwards the request to that neighbor. Otherwise, the request is directed towards the original data provider (i.e., the server). The exchange of cached content information increases the benefits gained from the tracking process. This leads to reduced delay compared to passive tracking schemes, such as Neighbor Caching [12]. However, limiting the search space to one-hop neighbors can be disadvantageous. This is since the dynamic nature of nodes causes the tracking information to be invalidated once neighboring nodes move out of range.

Among the tracking-based schemes that rely on information exchange between a group of nodes in MANETs is the scheme in [60]. In this scheme, cached content information is periodically exchanged between neighboring nodes that lie within a certain zone specified by a given number of hops. In order to resolve a request, a node first checks its local cache. In case of a cache miss, it determines whether a matching replica is available at one of the nodes in the specified zone. Otherwise, the request is directed to the server. This scheme expands the search space compared to schemes restricting the search space to one-hop neighbors only [12, 60]. However, this comes at the expense of increased communication overhead [12].

In ICNs, tracking-based schemes are also referred to as Index-based schemes [13]. Among such schemes is the one in [61], where a data forwarding node decides whether or not to reference the replica holder (i.e., data source) based on a probability. This probability is calculated based on the node's distance to the data holder. This tracking information can then be consulted to satisfy future requests. However, the lack of explicit information exchange between the nodes restricts the range of tracking information, thus limiting the search space and reducing cache hits [12, 13].

The schemes proposed in [62]-[93] are examples of tracking-based schemes in ICNs,

where one-hop neighbors exchange their cached content information for tracking purposes. Such schemes employ the same request resolution mechanism previously illustrated in MANETs. Due to the static nature of caching nodes in ICNs, maintaining up-to-date tracking information does not yield as much overhead as the case in MANETs in such schemes [12].

Most existing tracking-based schemes that rely on cached content information exchange between a group of nodes, can achieve reduced overhead and delay compared to broadcast-based schemes and probing-based schemes [12, 13]. They also extend the search space compared to server-based schemes [12]. However, in dynamic networks, intensive number of messages might need to be exchanged to maintain up-to-date tracking information, which could lead to huge overhead [12]. To reduce overhead, most existing tracking-based schemes restrict information exchange to neighboring nodes only [12]. However, this limits the search space, and can thus cause failure to locate caching nodes [12]. In this thesis, we propose two tracking-based cache discovery schemes that expand the search space beyond the neighborhood scope, and we employ the use of vehicles trajectory prediction to predict the location of mobile caching nodes and search for replicas beyond restriction.

2.4 Cooperative Cache Placement

Several cooperative cache placement schemes have been proposed in MANETs and ICNs. Such schemes can be categorized into global cooperative caching schemes and local cooperative caching schemes [12, 13]. In the former, each node sends its cached content information to all other nodes in the network. This helps make more informed caching decisions, leading to significant increase in data diversity and efficient utilization of the nodes storage resources, which can significantly yield higher cache hits. However, global cooperative caching schemes typically induce huge communication overhead, particularly in highly dynamic networks [12]. Thus, global cooperative

caching schemes are not typically adopted within the context of MANETs, whereas there are few existing schemes in ICNs [13]. In contrast to global cooperative caching, there are local cooperative cache placement schemes. In such schemes, cooperation occurs between a group of nodes only. This reduces the amount of communication overhead associated with the collaboration process. However, it limits the cooperation range, causing caching decisions to get quickly invalidated, particularly in dynamic networks. Local cooperative caching schemes can be further categorized into path-based and neighborhood-based schemes. We discuss each of these categories in the next subsections.

2.4.1 Global Cooperative Cache Placement

A global cooperative cache placement scheme in ICNs is proposed in [94]. This scheme dynamically assigns data items to ICN-based caching nodes. In this scheme, distributed managers that reside in caching nodes make caching decisions based on a global view of the status of all caches, as well as the relevant request patterns of data items. Each distributed cache manager exchanges information pertaining to its own cached contents, observed request rates, and popularity of data items with all other cache managers in the network. This information exchange takes place periodically or in an event-based manner when changes in the cache status occur. Upon receiving a data item, a distributed cache manager decides whether or not to cache the data by coordinating with all other managers in the network in order to minimize the overall network traffic. Towards this purpose, an optimization technique is implemented to make optimal caching decisions. The global cooperation among cache managers yields significant performance leverages and significantly reduces convergence time

[94]. However, this comes at the expense of substantial amount of overhead [94].

In [95], global cooperation takes place between ICN-based caching nodes, and an attempt to alleviate the costs associated with cache collaboration is made through the use of content popularity ranking. Note that popularity is calculated in a distributive manner by content routers. Each router sustains an Availability Information Base (AIB), estimating which content could be possessed by which router. A popularity ranking sequence is generated by each router via local measurements and is announced to all other routers. Accordingly, each router updates its AIB. In order to handle the inconsistency of content popularity measured by different routers, a self-adaptive dual segment cache design algorithm is employed. The pairwise link cost and cooperative forwarding/caching-related metrics are periodically announced by each router. The main objective of the scheme is to optimally assign contents to caching routers in a way that minimizes the average content access cost and cache miss rate. This scheme shows significant improvements compared to hierarchical caching schemes that can only exploit the storage resources of en-route caching nodes. However, the accuracy of the cached content information can be highly affected by the correctness of the popularity-based estimation process.

Despite the use of global cooperative caching in some schemes in ICNs, the extreme amount of communication overhead associated with the collaboration process renders these schemes disadvantageous in ICNs [13, 94, 95]. This is particularly the case when there is a substantial amount of different contents, such as social media traffic.

In contrast to global cooperative caching, there are local cooperative cache placement schemes. In such schemes, cooperation occurs between a group of nodes only.

This reduces the amount of communication overhead associated with the collaboration process. However, it limits the cooperation range, causing caching decisions to get quickly invalidated, particularly in dynamic networks. Local cooperative caching schemes can be further categorized into path-based and neighborhood-based schemes. We discuss each category in the following subsections.

2.4.2 Path-based Local Cooperative Cache Placement

In this type of schemes, cooperation is held between nodes along the delivery path [13]. Typically, such cooperation lacks the explicit exchange of cached content information between the nodes. This avoids incurring extra communication overhead. However, it narrows the collaboration scope and leads to increased redundancy and wasted cache space. For example, in Hamlet [96], which is a path-based cache placement scheme that is applied within the context of MANETs, nodes along the delivery path make caching decisions based on the diversity of information. In particular, each intermediate node records the distance between itself and the request initiator (i.e., where a replica is likely to be cached), as well as the distance to the data provider. The node then calculates an index of information presence for each data item. Upon receiving a data item, each node uses the presence index of such data item to determine whether a replica should be cached. This can reduce redundancy and wasted cache space along the delivery path. However, due to the dynamic nature of MANETs, the calculated information presence index may not be correctly reflected. In addition, requests patterns and data popularity are not considered in the caching decision.

Split Cache [97] improves upon Hamlet [96] by enabling intermediate nodes to

make caching decisions based on both the diversity of information and the frequency of content requests. Split Cache adopts a cache split policy, where the cache is split into two partitions, one caches popular data contents and the other caches less popular contents. Split cache reduces delay compared to Hamlet, since the former takes data popularity into consideration, which enables queries for frequently requested contents to be satisfied from nearby caching nodes, rather than having to propagate farther away. However, as previously mentioned, due to the frequent movement of nodes, both the presence and request patterns information can get quickly revoked [97]. Such rapid invalidation of this information further narrows the already limited collaboration scope in path-based schemes.

In ICNs, the static nature of caching nodes makes the aforementioned problem less of a concern. For example, in [98], caching decisions are made based on the number of copies to be cached on a forwarding path. This is estimated based on the cache capacity of the caching nodes along the delivery path. The main objective of such a policy is to enable fairness pertaining to the distribution of contents along the delivery path. For this purpose, content routers strive to probabilistically cache data items to enable fair multiplexing of the path cache capacity among various content flows per unit of time while sustaining considerable reductions in cache evictions and efficient utilization of the nodes storage resources. In addition, data items targeting requesters that are located further away from the data provider are less likely to be cached in nodes that are near this data provider. This is since such data items have a higher chance to be cached deep along the delivery path than contents targeting requesters that are located closer to a source. This scheme has been shown to yield significant reductions in terms of server hit ratio, delay, and cache evictions. However,

this comes at the expense of high computational cost [13, 98]. Also, the scheme does not consider the request patterns of data items in the caching decision [13].

The Least Unified Value-path scheme (LUV-path) [99] addresses the drawbacks of other schemes, such as [98], where there is a high probability of caching unpopular contents [13]. In addition to taking contents popularity into consideration, LUV-path enables caching routers along the delivery path to implicitly collaborate together to increase data diversity and reduce caching redundancy. Towards this purpose, LUV-path allocates different weights to passing-by contents between upstream routers and downstream routers. Such weights are assigned in a coordinated way based on the content popularity and the number of hops from the caching node to the data provider. LUV-path addresses the fact that duplicated contents in upstream routers are less likely to be accessed, leading to much lower cache hits. Thus, in LUV-path, the caching probability for new contents increases at upstream routers compared to downstream routers. As it is less likely for downstream routers to cache new data items, popular contents, whose weight increases every time they get accessed, would reside in downstream routers, thus closer to consumers. Thus, LUV-Path has the capability of directing popular contents towards the network edge to be located closer to consumers while increasing data diversity. However, the limited scope of the cooperation range significantly restricts the benefits gained from cooperative caching [13].

2.4.3 Neighborhood-based Local Cooperative Cache Placement

In this type of schemes, cooperation occurs between a node and its neighbors [12, 13]. Such schemes typically involve an explicit exchange of cached content information

among neighboring nodes in order to make more informed caching decisions. This slightly expands the cooperation range compared to path-based caching schemes [13]. However, this comes at the expense of increased communication overhead, particularly in dynamic networks [12, 13]. Consequently, many neighborhood-based schemes restrict the cooperation range to one-hop neighbors only [59, 100, 101, 102].

GroupCaching [59] is a neighborhood-based cooperative caching scheme proposed in MANETs. In GroupCaching, one-hop neighbors are organized into a group. Each node periodically exchanges its cached content information with its remaining group members. Caching decisions are then made collaboratively within each group in order to increase data diversity and avoid wasted cache space. Thus, upon receiving a data item, a node checks whether any of its group member already has the data in its cache. If not, the data is cached. If the node does not have enough cache space, it randomly selects one of its neighbors that have enough cache space and sends the data to it for caching. If there is no enough cache space in the group, the node selects the group member that has the least recently used data item and sends the data to it for replacement. Several other schemes in MANETs adopt the same approach, including [100]-[103]. In these schemes, the increased data diversity within the neighborhood of nodes increases data accessibility, and thus reduces access delay compared to path-based schemes [12, 13]. However, due to the mobility of nodes, the exchanged cached content information is characterized by a short lifetime, which results in unstable caching decisions [12].

Another neighborhood cooperative caching scheme in MANETs, where one-hop neighbors form a collective cache is proposed in [103]. In this scheme, the cache admission policy is based on data diversity within the group, as well as the distance

between the node and the data provider. If the distance between the former and the latter is too small, the data is not cached. This further avoids wasting the nodes storage resources. In addition, if the data is not already cached within the group and the node does not have enough cache space, a replacement policy that is based on a utility function is adopted. In such a policy, a certain utility function that is based on four factors is used. These factors include the popularity of the data item, its time-to-live, its size, and the distance between the data provider and consumer. This improves data accessibility and cache hit ratio [103]. However, the aforementioned problems are still predominant [12].

The scheme in [60] extends the cooperation range to include neighbors within a certain number of hops. Such neighboring nodes form a zone and in each zone, the nodes periodically exchange their cached content information and energy levels. Caching decisions are made based on both data diversity and content popularity to optimize cache placement within zones in order to efficiently utilize the available cache space and improve data availability. Thus, when a node receives a frequently accessed data item, it checks whether the data is already cached within its zone. If not, it caches the data. If the node does not have enough cache space, the least frequently accessed data item is selected for replacement and is migrated to the most valuable zone member. The most valuable zone member is the one with the highest cache space and residual energy level. This scheme tends to improve data availability and reduce access latency. However, as the number of hops specifying the zone range increases, the amount of communication overhead incurred due to the exchange of cached content information and energy levels significantly increases. Since such information can get quickly invalidated in networks with highly dynamic nature, the

additional communication overhead may be incurred without much profit.

As previously mentioned, in ICNs, the caching nodes are static. Thus, the lifetime of the exchanged cached content information is not as short as in dynamic networks. In [104], a neighborhood-based caching scheme is proposed to push popular contents closer to consumers, while reducing data redundancy and wasted cache space. In this scheme, one-hop neighbors form a group and their cached content information is periodically exchanged. The cache admission policy enables the caching of only one replica along the data delivery path. The caching node is determined based on two conditions. First, in order to avoid data redundancy, the data item should not be already cached at any of its neighbors. Second, if the incoming content is not the least frequently accessed compared to the already cached data items. In order to enable popular contents to reside near the consumers, caching decisions begin during the request forwarding process. As the request propagates from the requester to the data provider, the forwarding node compares the access frequency of the requested data with that of the least frequently accessed data item in its cache. If the aforementioned data diversity and content popularity conditions are satisfied, the node is designated as the content caching node. If not, the same process is repeated by the next request forwarding node. Note that when the data propagates back, it is cached at the designated node. This scheme brings the data closer to the requester, thus reduces access delay. However, the cooperation range is still limited. A multi-hop neighborhood collaborative caching scheme is proposed in [105] to better utilize the storage resources within several hops in ICNs.

Rather than determining whether or not to cache a data item, the neighborhood-based caching scheme in [63] enables nearby nodes to eliminate the redundancy of

their cached contents. Thus, neighboring nodes periodically evict duplicate contents from their caches in a collaborative manner. This enables a significant amount of the nodes storage resources to be released and further used to store more fresh and popular contents. To do so, each node periodically exchanges its cached content information with its one-hop neighbors. Whenever there is a potential gain yielded from redundancy elimination, a new round of the cooperative redundancy elimination process is triggered. This gain is calculated based on the amount of released caching slots.

The drawback of path-based and neighborhood-based schemes is that their cooperation range is limited [13]. We propose to expand this range by exploiting the mobility and static nature of moving and parked vehicles, respectively. These elements are used to acquire and distribute cached content information among a wider range of nodes, from parked to moving vehicles, and vice versa. As opposed to most existing techniques, along with on-path caching, we also apply an implicit form of off-path caching by exploiting the trajectory of vehicles.

2.5 Leveraging Cooperative Caching within VANETs

In this section, we highlight the type of approaches that can be leveraged and the ones that should be avoided when applying cooperative caching within VANETs. This is discussed from the perspective of the following characteristics:

- **Bandwidth Utilization:** VANETs are constrained in terms of bandwidth. Thus, due to the excessive bandwidth utilization associated with the broadcast-based and probing-based discovery schemes, they should be avoided within VANETs [12]. In contrast, tracking-based schemes can be leveraged [12]. However, the amount of overhead incurred during the cached content information

exchange in order to track the cached contents and the nodes holding them should be taken into consideration. This is particularly important when expanding the cooperation range beyond the neighborhood scope. Server-based discovery schemes typically involve the least amount of overhead among all the other discovery schemes [12, 13]. Such schemes are the ones that are typically used within VANETs. However, they extremely restrict the search space, which can lead to low cache hits [12, 13]. Since global-based cache placement schemes involve excessive bandwidth utilization due to the exchange of cached content information among all the nodes of the network [13], they should be avoided. In contrast, local-based cache placement schemes can be leveraged.

- **Dynamic Topology:** The schemes that can be affected by the rapidly changing topology in VANETs include the probing-based discovery, the tracking-based discovery, and the global-based cache placement schemes [12, 13]. In probing-based discovery schemes, prior to issuing a request, a node broadcasts a probing message to determine the location of the nodes that have the data. The closest node (i.e., the first one to respond) is then selected to be the target of the issued request. This selection is done under the assumption that by the time the request is issued, the caching node will still be the closest. In this case, a changing topology might lead to some increase in delay and bandwidth utilization [12]. This adds more delay to the already increased delay involved in the probing-based discovery approach compared to other discovery schemes [12, 13]. This is due to the probing process that needs to be performed before the request-response process could even begin. In tracking-based schemes, the tracking information pertaining to the nodes holding the cached contents and

their locations, could be quickly invalidated due to the highly dynamic topology in VANETs [12]. However, there is an opportunity in VANETs to exploit the fact that beacon messages are periodically exchanged among neighboring vehicles. Such messages could be leveraged by each vehicle to include its own cached content information and exchange it with its neighbors. In addition, the predictability of the movement of vehicles can be used to predict their future location in order to keep track of the position of caching nodes even when vehicles move out of range.

Chapter 3

Predictive Proactive Caching at Parked Vehicles

3.1 Introduction

In this chapter, we explore the use of predictive proactive caching by proposing a Predictive Proactive Caching Framework (PPCF) to reduce the load at backhaul links and provide a certain QoS to social media users that have a predictable behavior. PPCF relies on the fact that some users tend to maintain a daily routine in the route they follow, the time of navigating that route, and the type and time of requesting a certain content. For instance, as a vehicle is moving during a typical morning on the driver's daily route to work, he/she is used to dropping off their children at school first, who in turn are used to accessing their Instagram accounts to check the latest posts of public figures. Such a daily routine triggers a predictable behavior that PPCF utilizes to pre-cache the data at parked vehicles for the users to proactively acquire either before the time of their requests or within a specified time frame. Hence, PPCF aims to sustain a certain quality of service demanded by users.

PPCF consists of two basic modules executed by the data center. Note that the data center is the original data provider, and it is the centralized entity that makes

all vital decisions in these two modules. The first module is the prediction module that predicts all the information required to make informative caching decisions. This information includes the period of encounter of requesters with each road segment along their trajectory. We mainly focus on the prediction procedure of this period. This is due to the effect of different weather and traffic conditions on the prediction accuracy, which can in turn have a significant impact on the performance of the cache placement procedure. For this purpose, we propose a novel travel time prediction technique that uses the Long Short-Term Memory (LSTM) model, trained using Particle Swarm Optimization (PSO), while taking weather and traffic conditions into consideration. To the best of our knowledge, this is the first PSO-based LSTM model to be introduced in travel time prediction. This is as opposed to the typical LSTM model trained using the gradient descent (GD) backpropagation algorithm. In addition, we use the predicted average travel time to estimate a personalized travel time for each user by considering their different driving behaviors.

The second module is the proactive cache placement module that enables the data center to select the appropriate road segments for caching. In order to make the cache placement decision and tackle all the related time-sensitive constraints, we present an optimal solution, called the Vehicular Optimal Proactive Caching (VOPC). The objective of VOPC is to maximize cache hits by assigning replicas to caching spots that yield maximum certainty in their spatiotemporal availability for requesters. This is while sustaining a cache capacity limit and a predetermined quality of service. Once a road segment is selected, the data center chooses the parked vehicle that has the maximum available cache space to cache the replica.

VOPC is introduced as a benchmark that can be used to determine the upper

bound of the potential of predictive proactive caching in improving the quality of service in VANETs. It enables researchers to quantify the potential gains of predictive proactive caching schemes, assess their performance based on the gap to the optimal solution, and determine if there is a room for improvement. The caching problem in VOPC is formulated as an Integer Linear Programming (ILP) problem. Considering the known complexity of ILP, solving this problem is NP-Complete. Thus, a greedy heuristic scheme, called the Proactive Caching at Parked Vehicles (PCPV) scheme, is also proposed to cope with practical real-time requirements.

We implement VOPC and PCPV using the NS-3 network simulator [64] integrated with the Gurobi optimizer [65] in order to generate the optimal solution in VOPC. The performance of the optimal and heuristic solutions VOPC and PCPV are compared, and we use VOPC to quantify the potential gains of PCPV. In addition, we assess the performance of predictive proactive caching compared to the baseline broadcast-based approach. This is while neutralizing the issues that render the latter unsuitable for Internet access applications. Moreover, extensive experiments are conducted to evaluate the performance of PCPV compared to VOPC and the broadcast-based approach under varying conditions. Simulation results show that VOPC and PCPV significantly outperform the broadcast-based approach in terms of delay, packet delivery ratio, cache hit ratio, and satisfaction ratio. They also show that VOPC can act as an upper bound on the reachable potential of predictive caching, and that PCPV can perform very close to the optimal solution in many cases.

In order to assess the performance of the proposed prediction model, namely PSO-LSTM, we compare it to the GD-LSTM model. Simulation results show that the former can significantly outperform the latter in terms of prediction accuracy. We

also implement VOPC and PCPV using both PSO-LSTM and GD-LSTM in order to evaluate the effect of the level of prediction accuracy on the efficiency of the cache placement procedure. We refer to them as VOPC-PSO, VOPC-GD, PCPV-PSO, and PCPV-GD.

The remainder of this chapter is organized as follows. Section 3.2 outlines some related work. Section 3.3 presents the predictive proactive caching framework, including the prediction module, and the cache placement module. Section 3.4 illustrates the performance evaluation and simulation results. Section 3.5 summarizes and concludes the chapter.

3.2 Related Work

An overview of the related work in proactive caching has already been discussed in Section 2.2.1 in Chapter 2. Thus, in this section, we only highlight some of the related work in travel time prediction in VANETs.

Due to the fluctuations of the vehicles' travel time on roads, there is a strong need for mathematical models that are capable of predicting travel time with adequate accuracy. The most prominently used travel time prediction models can be classified into four categories; historical average models, regression models, filtering models, and machine learning models [66]-[68].

Historical average models [69]-[71] predict future travel time based on historical data collected from previously observed trips. On the one hand, the algorithms incorporated in historical average models are typically simple and involve relatively small computation time. On the other hand, they work under the assumption that traffic conditions remain stationary. Thus, these models are adequately accurate only

in cases when traffic conditions are somewhat stable. Some historical average models, such as [69] and [70], predict the arrival time of vehicles using the historical average travel time directly, or jointly with other inputs. Other models, such as [71], predict the travel time of vehicles using their average speed on road segments.

In regression models, a multivariate statistical technique is applied to determine the correlation that exists between a set of independent variables and a given dependent variable [66]. In contrast to historical average models, linear regression models can perform adequately under unstable traffic conditions [66]-[68]. Linear regression models have been lucubrated by many researchers to predict vehicles' arrival time [72, 73]. For example, in [72], a set of linear regression models have been proposed, where the distance over links, and weather parameters are defined as the set of independent variables. In [73], linear regression models have also been used, with different sets of independent variables defined. In [74], a linear model predicts the travel time of a given trip at the current time by integrating the most recent calculated travel time of the trip and the historical average travel time of the same trip departing at the same time. Linear regression models have the capability to determine the significance of independent variables in predicting the travel time [66]. However, the applicability of such models is typically limited [66]. This is due to the inter-correlation between variables in transportation systems [66]. Other regression models, such as the Autoregressive Integrated Moving Average (ARIMA) model [75] have been explored. ARIMA fits a time series model by exploiting the historical series of travel time and then estimates the future travel time one by one [75].

Filtering models, such as the Kalman filter model [76], exhibit the dynamic capability of updating their predictions according to new data that reflect the varying

features of the transit-operating environment. In particular, such models are able to continuously update the state variable (i.e., the travel time) as new observations manifest. However, data fluctuations might lead to difficulties in solving time lag [68, 76].

Machine learning models, such as Artificial Neural Networks (ANNs), are highly popular in predicting travel time as a result of their ability to solve difficult non-linear relationships. For example, the objected-oriented neural network approach used in [77] composes the network input of the amalgam of the current observed speed and the flow data of the upstream and downstream stations along the freeway section, and uses that to predict the next travel time on that section. Support Vector Machines (SVMs) have also been used for travel time prediction [78]. ANNs and SVMs have shown significant superiority over other models in terms of prediction accuracy, including the historical average and regression models, as demonstrated in the studies conducted in [72, 79, 80]. However, despite such superiority, they still lack the ability to dynamically modify the prediction results [67]. In other words, the prediction capability of such models is limited to estimating the travel time depending on historical data only with no regard to real-time information.

Recently, the Long Short-Term Memory (LSTM) neural network model has also been investigated for travel time prediction [81]. It has been shown in [81] that deep learning models, such as LSTMs, can render promising results in terms of accuracy in the travel time prediction problem. This is due to their ability to take sequential dependencies into consideration. However, the work in [78] can only provide a short-term prediction of the average travel time. Recently, LSTMs have been used to perform long-term travel time prediction [82]. The main goal in this type of prediction

is to predict the travel time at a specific day and time at least one week ahead. However, the LSTM models used do not consider the effect of the varying weather conditions. In addition, most of the existing ANNs and LSTMs developed in order to predict the travel time are based on the Backpropagation algorithm (BP) that relies on GD [72, 79, 80]. The most prominent drawback of the BP algorithm is local minima entrapment [72, 79].

In order to improve the training process, we propose the use of Particle Swarm Optimization (PSO) rather than BP to train the LSTM model. Note that PSO has been previously used with ANNs in some engineering problems, such as predicting voltage stability [83], and solving power flow problems [84], and it has been shown to provide high prediction accuracy. The proposed model also enables long-term predictions, while taking into consideration the varying weather conditions. Furthermore, while most existing predictions focus on the average travel time, we also provide personalized estimates of the travel time of each user.

3.3 Predictive Proactive Caching Framework

In the proposed Predictive Proactive Caching Framework (PPCF), we assume that requesters subscribing to the service grant the data center access to their trajectories, as well as their social media profiles. This information is granted in exchange for the service and according to a privacy agreement. As previously mentioned, the data center is the original data provider accessed via an RSU. Each user indicates his/her maximum acceptable delay upon subscribing to the service. Thus, the deadline of each user's request is determined. The data center is responsible for choosing the optimal road segments for caching. The data center is equipped with a prediction

module that allows it to predict three pieces of information. First, it predicts the request time of each user using request patterns predictors, such as [91]. Second, it estimates the posting frequency of a given public figure. This information indicates the Time-to-Live (TTL) of the data. Third, it predicts the period of encounter of the requester with each road segment along its specified trajectory.

In the proposed prediction module, we focus on the prediction of the period of encounter only. We do so due to the fact that the prediction accuracy can be significantly affected by various factors, including dynamic weather conditions. In contrast, the remaining pieces of information are not affected by many factors, so they can be predicted with significantly high accuracy using many exiting predictors [91]. Note that the posting frequency of a given public figure specifies the TTL of the data. This TTL indicates the expected time before a public figure publishes a new post, rendering the previous data obsolete.

After a road segment is selected, the data center sends the replica to the parked vehicle that has the maximum available cache capacity at the selected road segment to be cached. Parked vehicles that are willing to dedicate part of their storage capacity to the caching service are solicited by the data center in exchange for some incentives. In the following subsection, we present the system model.

3.3.1 System Model

Let U be the set of users subscribing to the service. Every user $u \in U$ can request a certain data $d \in D$, where D is the set of recent contents published by public figures followed by the users. Each data item d has a time-to-live TTL_d, after which a new post is generated by the corresponding public figure. Such a new post is considered

a new version of d. Each version of a data item d, denoted l^d, is associated with a generation time g_l^d, and an expiry time t_{lexp}^d. The trajectory of every user $u \in U$ is denoted T_u. Note that T_u= r_1 , r_2, ..r_k represents the sequence of road segments along the vehicle's trajectory, where r_j denotes the road segment encountered by the vehicle. The set of all road segments is denoted R. A road segment $r_k \in R$ represents a directed edge e_{ij} between two different intersections I_i and I_j, where $e_{ij} \neq e_{ji}$. The time at which a given l^d is cached at a road segment r_j is denoted t_{lcach}^{jd}. The maximum cache capacity at a road segment r_j is denoted C_j.

The time of request of user u for data d is denoted τ_d^u, and the deadline of the request is denoted η_d^u. A time period, denoted ϵ_j^u, is assigned to each road segment $r_j \in T_u$ along the trajectory of u. This period indicates the start time $t_{u,j,s}$ and the end time $t_{u,j,e}$ during which it is estimated that user u will encounter r_j (i.e., its time of arrival and departure to and from r_j). A probability of encounter, denoted P_j^u, is associated with each period of encounter. Such a probability represents the level of certainty of the estimated period. We only focus on the level of certainty of the period of encounter due to the profound effect of varying weather and traffic conditions on it, which can in turn impact the spatiotemporal availability of the cached replicas.

The problem of estimating the period of encounter can be considered as a travel time prediction problem considering that the users can provide the time at which they begin their daily trips, or such time can be estimated using historical average models [71]. This indicates the time of arrival to the first road segment along their trajectory. The time of departure from any road segment r_j along the user's trajectory can be calculated as given by Eq. 3.1, where $\gamma_j^{t_{u,j,s}}$ is the estimated travel time along r_j at time $t_{u,j,s}$. The time of arrival at any road segment r_j is equal to the departure

time from the previous road segment along the user's trajectory (i.e., $t_{u,j,s} = t_{u,j-1,e}$). Thus, once the travel time is estimated, the period of encounter of each vehicle with each road segment along its trajectory can be determined.

$$t_{u,j,e} = t_{u,j,s} + \gamma_j^{t_{u,j,s}} \tag{3.1}$$

It can be deduced from the way the period of encounter is estimated that the effect of the travel time prediction error tends to accumulate as users move further along their trajectory. This accumulation occurs since the arrival time of any vehicle at any road segment r_j along its trajectory $t_{u,j,s}$ relies on the departure time from its previous road segment r_{j-1}, which in turn depends on the estimated travel time at r_{j-1}. Thus, the first road segment along a user's trajectory tends to render the least error compared to subsequent road segments, whereas the last one renders the highest error. Accordingly, the closer a road segment is to the first road segment in the sequential order of a user's trajectory, the lower the prediction error, and thus the higher the certainty. Thus, the probability of encounter P_j^u is calculated as given by Eq. 3.2, where $\Lambda_{r_j}^u$ is the position of r_j in the sequential order of trajectory T_u of user u.

$$P_j^u = 1 - \frac{\Lambda_{r_j}^u - 1}{\sum_{i=1}^{|T_u|} \Lambda_{r_i}^u - 1} \tag{3.2}$$

Note that the data center can have access to a plethora of traffic monitoring services that provide the most recent traffic updates. This includes the actual average travel time at every road segment. Parked vehicles can also periodically report information pertaining to the actual average travel time at the road segments where they reside. Such updates can be used to adjust the cache placement decisions if the

actual values of the average travel time significantly differ from the predicted values.

3.3.2 Prediction Module

In this section, we present the proposed PSO-LSTM prediction module. To do that, we first provide a detailed discussion of the data preparation stage. The model prediction methodology and training procedure are then illustrated.

A) **Data Preparation**

The dataset used is generated using the Simulation of Urban MObility (SUMO) tool [86], which is a microscopic traffic simulator that can accommodate large road networks, import real-life road maps, and create realistic mobility traces. Using SUMO, we create a road topography that consists of 120 different edges, and emulate the traffic behavior of users that have a repetitive pattern in terms of the daily route they follow. This is done over a period of 6 months, where the traffic of one day is simulated over a period of 2000 seconds, and only two types of days, triggering two different types of traffic distribution are considered per week, namely weekdays and weekends. Typically, changes in traffic does not occur every second, and thus the travel time values tend to remain relatively stable for some period of time [76, 81, 82]. Accordingly, we divide the total period of 2000 seconds per day into a number of consecutive time intervals, each of which is composed of 30 seconds. Thus, for each edge, there is a total of 66 rows representing the travel time values per day at that edge, and 528 rows per month. The 6-month period involves 3,168 rows per edge, and a total of 380,160 for all edges.

Each of the aforementioned rows includes the following information: the weather condition, the day of the week, the time of the day, the road segment ID (i.e., edge ID),

and the average travel time. We consider three types of weather conditions (sunny, moderately snowing, and heavily snowing), as well as two types of days (weekdays, and weekends). In order to create realistic traffic, we adopted the analysis used in [89] that studies the variations in traffic conditions over one week to detect the changes in traffic volume that occur over the day during weekdays and weekends. Note that aside from the time of the day and the average travel time, the remaining aforementioned features are categorical features. Thus, we represent each of them as a vector of 1's and 0's using one-hot encoding [90].

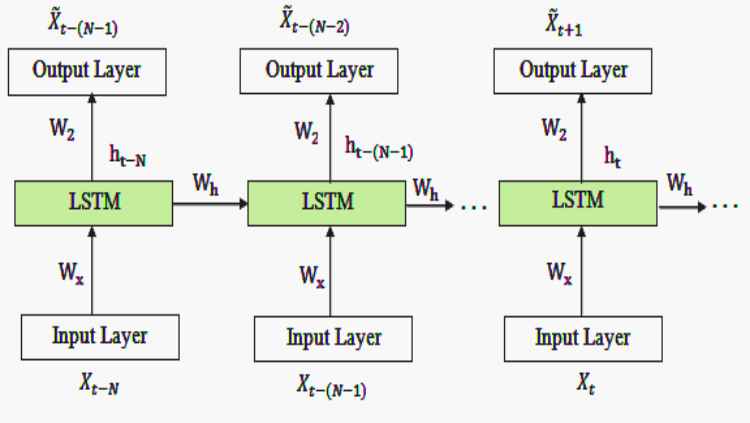
We take different weather conditions into consideration since they can have a profound impact on the daily traffic. For instance, in case of snow, the road surfaces tend to become rather slippery, which forces drivers to become more cautious as they adopt a higher following distance rate and lower speed in order to avoid accidents [87]. Some studies have been conducted to reproduce that same effect in traffic simulators in a realistic manner, including SUMO [87, 88]. In [87], real data sets were used in order to facilitate the calibration of SUMO to different weather conditions. In [87], it has been shown that the effect of weather conditions can be simulated by varying three parameters in SUMO. Such parameters are the acceleration, desired speed, and the car-following distance [87]. Note that the latter can be controlled using the parameter bx in the Wiedemann car-following model incorporated into SUMO [87]. In alignment with the findings of [87], different weather conditions can be reflected by varying the acceleration of vehicles in SUMO between 0.1 and 1.0, with 0.05 increments, the desired speed between 15 km/h and 55 km/h, with increments of 5 km/h, and the following distance between 0.25 and 6 with 0.25 increments.

In long-term travel time predictions, as the case in our proposed model, the prediction of future travel time along any given road segment at a certain time period on Saturday for example requires exploiting the data representing the travel time values at the same time period in the N previous Saturdays of the past N weeks. Accordingly, the data preparation stage converts the aforementioned rows of the dataset into a sequential structure, where each sample is composed of N time steps, each of which consists of the previously discussed features [82].

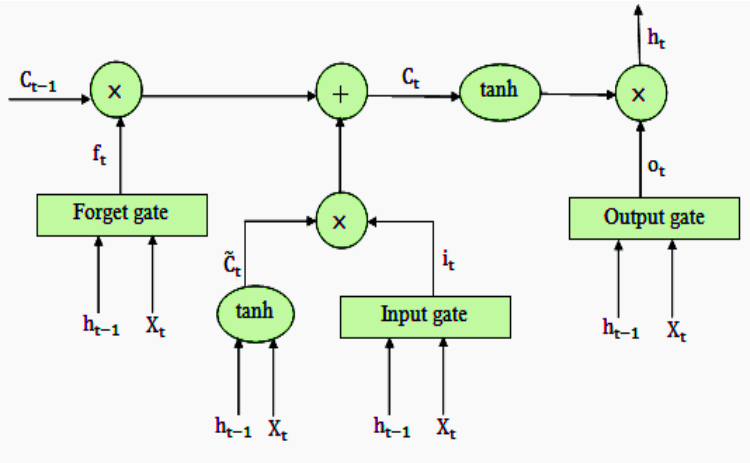
B) **Long Short-Term Memory Model (LSTM)**

LSTMs are a special type of Recurrent Neural Networks (RNNs) that have the ability to learn long-term dependencies [81]. The ability to remember information that goes back to a long period of time is the default behavior of LSTMs. It is not a behavior that they struggle to learn. As a result, LSTMs have been shown to work well in problems related to time-series prediction [81].

An LSTM has a chain-like structure that is composed of a number of cells in its hidden layer, referred to as LSTM cells. As depicted in Figure 3.1(a), each LSTM cell relays a message to its successor. An LSTM cell consists of three gates, referred to as input gate, forget gate, and output gate. Such gates control the flow of information to the cell state [81, 82]. As depicted in Figure 3.1(b), the cell state is the horizontal line that runs down through the entire chain at the top of the diagram. The gates enable regulating the information that need to be removed and/or added to the cell state. At time t, the input is denoted x_t. Such an input can be a single value or a multivariate input. In our case, x_t is a multivariate input. It is a matrix of size $X \times B$, where X is the size of the input features, and B is the batch size [81, 82]. The batch size is used in the gradient descent training process to indicate the number of training



(a) LSTM neural network.



(b) LSTM cell structure.

Figure 3.1: LSTM neural network architecture. Redrawn from [81]

samples that the model goes through before updating the internal parameters. In our LSTM model, $B = 1$. The cell input state is denoted $\tilde{C}_t$, the cell output state is denoted C_t, and its preceding state is denoted C_{t-1}. The hidden layer output and its previous output are denoted h_t and h_{t-1}, respectively. The size of $\tilde{C}_t$, C_t, h_t, and h_{t-1} is $H \times B$, where H is the number of hidden units [81]. The input, forget, and output gates are denoted i_t, f_t, and o_t, respectively, and are all of size $H \times B$. The

forget gate is used to determine the information to forget or to keep, the input gate specifies which values to update, and the output gate decides which parts of the cell state to output [81, 82].

As demonstrated in Figure 3.1(b), the LSTM cell structure shows that both C_t and h_t are passed on to the next cell [81]. In order to calculate C_t and h_t, the following procedure, referred to as the forward propagation procedure, is performed [81].

1) The input, forget, and output gates are first calculated using Eq. 3.3, Eq. 3.4, and Eq. 3.5, respectively. The cell input state is then calculated as given by Eq. 3.6. The matrices W_x^i, W_x^f, W_x^o, and W_x^C are the weight matrices of size $H \times X$ that connect the input x_t to the input gate, forget gate, output gate, and the cell input state, respectively. In contrast, the matrices W_h^i, W_h^f, W_h^o, and W_h^C are the weight matrices of size $H \times H$ that connect h_{t-1} to the input gate, forget gate, output gate, and the cell input state, respectively. The matrices b_i, b_f, b_o, and b_C are the bias matrices of size $H \times B$ of the input gate, the forget gate, the output gate, and the cell input state, respectively. Note that σ is the sigmoid function, which is given by Eq. 3.7, and tanh is the hyperbolic tangent function.

$$i_t = \sigma(W_x^i.x_t + W_h^i.h_{t-1} + b_i) \tag{3.3}$$

$$f_t = \sigma(W_x^f.x_t + W_h^f.h_{t-1} + b_f) \tag{3.4}$$

$$o_t = \sigma(W_x^o.x_t + W_h^o.h_{t-1} + b_o) \tag{3.5}$$

$$\tilde{C}_t = \tanh(W_x^C.x_t + W_h^C.h_{t-1} + b_C) \tag{3.6}$$

$$\sigma(z) = \frac{1}{1 + \exp(-z)} \tag{3.7}$$

2) The cell output state C_t is calculated using Eq. 3.8.

$$C_t = i_t * \tilde{C}_t + f_t * C_{t-1} \tag{3.8}$$

3) The hidden layer output h_t is calculated using Eq. 3.9.

$$h_t = o_t * tanh(C_t) \tag{3.9}$$

4) In order to predict the travel time, we create an LSTM neural network as depicted in Figure 3.1(a) [81]. At time t, the input x_t is the historical data that includes information about the weather, the day of the week, the time of the day, the road segment, as well as the historical average time. The final output of the network, denoted $\tilde{x}_{t+1}$, is the predicted future average time on the same day along that road segment. Using the aforementioned calculated value of h_t, we calculate the predicted value as given by Eq. 3.10, where W_2 is the weight matrix connecting the hidden layer and the output layer, and b is the bias term of the latter. Note that the series LSTM-based prediction shown in Figure 3.1(a) is based on N historical data.

$$\tilde{x}_{t+1} = W_2.h_t + b \tag{3.10}$$

The aforementioned steps require learning the weight and bias matrices that would render the minimum prediction error. Thus, the LSTM model needs to be trained in order for these matrices to be learned. We provide a detailed discussion of the proposed training procedure below.

C) **LSTM Training**

The functional relationship between dependent and independent variables is not enforced in LSTMs [83]. Rather, such a relationship is based on the training process. Thus, the training process is a pivotal component that has a profound impact on the prediction accuracy [83]. In most existing LSTM-based models, the training process is typically based on the gradient descent Backpropagation algorithm (BP) [81, 82]. This algorithm can lead to local minima entrapment. Thus, we propose to exploit the use of PSO during the training process since it has the capability of significantly expanding the search space compared to the gradient descent algorithm, thus increasing the chance of finding the global minima [83, 84].

The PSO algorithm was proposed by Kennedy and Eberhart as a metaheuristic algorithm that relies on the concept of swarm intelligence, which has the ability to solve complex mathematical problems in engineering [85]. In PSO, a number of particles move in the search space based on certain dynamics until they reach the best solution [83, 84]. It can be successfully incorporated into the optimization of nonlinear continuous functions. PSO is also easy to implement without the need for any gradient information [83, 84].

In PSO, every solution of a given problem is represented by a particle that can iteratively navigate in the search landscape for the purpose of reaching the best solution. The total number of particles is denoted S, the total number of iterations is denoted K, and the total number of components is denoted J. The position of each component $j \in J$ of each particle $p_i \in L$ at iteration $k \in K$ is iteratively updated by considering two vectors, namely the position vector and the velocity vector, denoted Z_{ij}^{k+1} and V_{ij}^{k+1}, respectively. The former specifies the position of the particle in the search landscape, while the latter indicates the direction and intensity of movement.

In the proposed PSO-LSTM, the position and velocity of each particle is expressed in the form of a matrix rather than a vector. In particular, the components of each particle are the aforementioned weight and bias matrices, whose optimal values (i.e., position) need to be learned. Note that the position of each particle is evaluated based on a certain fitness function. In the proposed model, this function is the Root Mean Square Error (RMSE), which is given by Eq. 3.11, where Q is the set of observations/samples, and $|Q|$ is the number of observations, $\tilde{x}_q$ is the q^{th} predicted value of the travel time and x_q is its corresponding q^{th} observed value.

$$RMSE = \sqrt{\frac{\sum_{q=1}^{|Q|}(\tilde{x}_q - x_q)^2}{|Q|}} \tag{3.11}$$

In each iteration k, the matrices Z_{ij}^{k+1} and V_{ij}^{k+1} of the component j of each particle p_i are calculated using Eq. 3.12 and Eq. 3.13, respectively [85]. In Eq. 3.12, P_{ij}^k is the best local position of the j^{th} component of p_i, and G_j^k is the best global position of the j^{th} component among all particles [85].

$$V_{ij}^{k+1} = \omega V_{ij}^k + c_1 r_1 (P_{ij}^k - Z_{ij}^k) + c_2 r_2 (G_j^k - Z_{ij}^k) \tag{3.12}$$

$$Z_{ij}^{k+1} = Z_{ij}^k + V_{ij}^{k+1} \tag{3.13}$$

Note that the best position indicates the solution rendering the minimum RMSE. ω is the inertia weight that lies in the range $[0, 1]$ and controls how much the particle's movement is influenced by its preceding motion [85]. c_1 and c_2 are the learning coefficients of the local and global solution, respectively [85]. They help weigh the importance of the previous experiences of the particles [85]. r_1 and r_2 are two random

numbers in the range $[0, 1]$ that are used to control the randomness of the search [84]. Such random parameters can help avoid premature convergence, thus increasing the chance of finding the global optima [85].

Based on the aforementioned discussion, the PSO-LSTM training procedure works as follows (Algorithm 1):

1) Set the number of components J to 10, where the components represent the weight and bias matrices (line 12). Initialize the the local minimum error of each particle, denoted RMSE_i^{Lmin}, as well as the global minimum error, denoted RMSE^{Gmin} (lines 13 & 14). For each particle $p_i \in M$, and for each component $j \in J$, initialize the position and velocity matrices, Z_{ij}^k and V_{ij}^k, and set the best local position P_{ij}^k, as well as the weight and bias matrices to the corresponding Z_{ij}^k (lines 15-19).

2) In each iteration $k \in K$, do the following for each particle $p_i \in M$ (lines 20 & 21):

a) Perform the LSTM forward propagation procedure to calculate the predicted value of each sample $q \in Q$. This is done using equations Eq. 3.3- Eq. 3.10 (lines 22-24).

b) Based on the predicted travel time values acquired, calculate the error $RMSE_i^k$ using Eq. 3.11.

c) If RMSE_i^k is less than the particle's local minimum error, set the latter to RMSE_i^k, and its best local position to Z_{ij}^k (lines 26-29). Otherwise, they remain the same.

d) Determine the particle, denoted p_b, that renders the minimum error among all particles (i.e., the one that has the minimum local RMSE). If the particle's local minimum error is less than the global minimum error, set RMSE^{Gmin} to the particle's local minimum RMSE, and set p_b to p_i (lines 30-32). Otherwise, RMSE^{Gmin} remains the same (lines 33 & 34).

3) Once all the particles are processed, if RMSE^{Gmin} has changed, the best global

Algorithm 1 : PSO-LSTM Training

Input:
Number of historical time steps N
Set of training data Q
Number of iterations K
Number of particles S
Number of components J
Learning coefficients c_1 & c_2
Inertia weight ω

LSTM_TRAINING(N)
Begin
  $J \leftarrow 10$
  $\text{RMSE}^{Gmin} \leftarrow \infty$
  $\text{RMSE}_i^{Lmin} \leftarrow \infty$ $//\forall i \in S$
  for all $i \in S$ **do**
    for all $j \in J$ **do**
      Initialize Z_{ij}^0 and V_{ij}^0
      $P_{ij}^0 \leftarrow Z_{ij}^0$
      Set the weight & bias matrices to the corresponding Z_{ij}^0
  for all $k \in K$ **do**
    for all $i \in S$ **do**
      for all $q \in Q$ **do** // each sample in Q
        for all $t \in N$ **do** // each time step t
          Calculate the predicted value $\tilde{x}_q$ // Eq. 3.3 - Eq. 3.10
      Calculate the error $RMSE_i^k$ // Eq. 3.11.
      if $RMSE_i^k < RMSE_i^{Lmin}$ **then**
        $RMSE_i^{Lmin} \leftarrow RMSE_i^k$
        for all $j \in J$ **do**
          $P_{ij}^k \leftarrow Z_{ij}^k$
      if $RMSE^{Gmin} < RMSE_i^{Lmin}$ **then**
        $RMSE^{Gmin} \leftarrow RMSE_i^{Lmin}$
        $p_b = p_i$
      else
        $RMSE^{Gmin}$ remains the same
    if $RMSE^{Gmin}$ is not the same **then**
      for all $j \in J$ **do**
        $G_j = Z_{bj}^k$
    for all $i \in S$ **do**
      for all $j \in J$ **do**
        Update V_{ij}^{k+1} // Eq. 3.12.
        Update Z_{ij}^k // Eq. 3.13.
  return G_j $//\forall j \in J$
End

position of each component G_j^k is set to that of p_b (i.e., G_j^k=Z_{bj}^k) (lines 35-37).

4) For each particle, update the position of the weight and bias matrices using Eq. 3.12 and Eq. 3.13 (lines 38-41).

5) After finishing the last iteration, the global best position G_j is acquired $\forall j \in J$ (line 42). The final weight and bias matrices are set accordingly.

Once the training procedure is terminated, and all hyperparameters are determined, we perform the LSTM forward propagation procedure on the testing data to predict the average travel time on each road segment at each time interval. We then estimate the travel time of each individual user. In order for this to occur, we perform the following procedure, detailed below, after training the model.

D) **Travel Time Estimation of Individual Users**

In our proposed prediction scheme, we consider the fact that the personalized travel time of different drivers can deviate from the estimated average travel time based on their driving behavior. For instance, more cautious drivers tend to move relatively slower than most average drivers, while expert drivers can move a little faster. Thus, we classify the users in our dataset into three classes; class 1: cautious, class 2: average, and class 3: expert. This is done by calculating the degree of their membership to each class, denoted ϑ_{ij}. The degree of membership of user i to class 1, ϑ_{i1}, is calculated as the ratio of the number of times that the individual travel time of user i subceeds the average travel time by a certain threshold, denoted th_{dm}, to the total number of samples. In contrast, ϑ_{i3} is the ratio of the number of times that the individual travel time of user i exceeds the average travel time by th_{dm}, to the total number of samples, while ϑ_{i2} is the ratio reflecting the number of times that neither the first case nor the second one applies. The class rendering the highest membership

is the one to which user i belongs.

(1) We cluster the training data into Y clusters, so that those belonging to users moving along the same road segment on the same type of day, as well as at the same time interval, and under the same weather conditions over the available months, are grouped together.

(2) Each cluster $y \in Y$ is then divided into B_y sub-clusters enclosing users in y whose movement is associated with the same month and the same week.

(3) For each cluster $b \in B_y$, $\forall y \in Y$, we calculate the standard deviation, $SD_{b,y}$ of the users' individual travel time. $SD_{b,y}$ is calculated based on Eq. 3.14, where $\acute{\gamma}_{b,y}$ is the average travel time of the users in cluster $b \in B_y$, $\gamma_{i,b,y}$ is the individual travel time of user $u_i \in b$, and $n_{b,y}$ is the number of users in $b \in B_y$.

$$SD_{b,y} = \sqrt{\frac{\sum_{i=1}^{n_{b,y}} (\gamma_{i,b,y} - \acute{\gamma}_{b,y})^2}{n_{b,y}}} \tag{3.14}$$

(4) We then calculate the pooled standard deviation for each cluster $y \in Y$, denoted SD_p^y to estimate the aggregated standard deviation pertaining to all the sub-clusters B_y belonging to y. Note that this will be used to estimate the final individual travel time of users. The value of SD_p^y is calculated using Eq. 3.15.

$$SD_p^y = \sqrt{\frac{\sum_{b=1}^{B_y} (n_{b,y} - 1) SD_{b,y}^2}{\sum_{b=1}^{B_y} (n_{b,y} - 1)}} \tag{3.15}$$

5) Once the pooled standard deviation is determined, the estimated individual travel time of each user u in the testing set satisfying the same conditions in y can be

calculated based on Eq. 3.16, where $\bar{\gamma}_y$ is the predicted average travel time.

$$\tilde{\gamma}_{u,y} = \begin{cases} \bar{\gamma}_y - SD_p^y & \text{if } u \text{ is a cautious driver} \\ \bar{\gamma}_y & \text{if } u \text{ is an average driver} \\ \bar{\gamma}_y + SD_p^y & \text{if } u \text{ is an expert driver} \end{cases} \tag{3.16}$$

3.3.3 Optimal-based Proactive Cache Placement

In this subsection, we present the optimal solution of the proactive cache placement module, called the Vehicular Optimal Proactive Caching (VOPC). VOPC is introduced as a benchmark that can act as the upper bound on the reachable potential of predictive proactive caching. We provide a detailed discussion of the ILP cache placement problem formulation.

Our objective is to maximize cache hits that ensure the spatiotemporal availability of the cached replicas for the users to proactively acquire within a certain time frame. In order to ensure such an availability, the following restrictions need to be taken into consideration when making a caching decision at road segment r_j to satisfy the request of user u for data item d:

1) r_j must be part of the user's trajectory ($r_j \in T_u$).

2) The deadline of the request should be after the user u starts its period of encounter with r_j ($\eta_d^u > t_{jstart}^u$). This is to ensure that the user does not acquire the cached data after the deadline of the request.

3) The cached replica should still be valid by the time the user starts passing by r_j ($t_{lexp}^d > t_{jstart}^u$). This is to guarantee that the data is acquired before its expiry time.

4) The cached replica must still be valid by the time the user requests it ($t_{lexp}^d >$

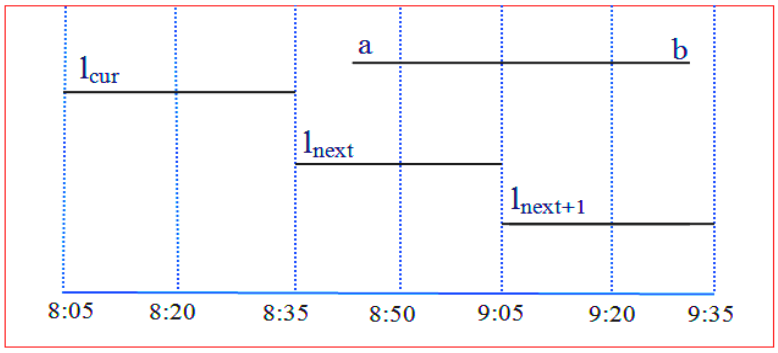
τ_d^u).

5) The replica must be already cached at r_j before the user is done passing by the road segment ($t_{jend}^u > t_{lcach}^{jd}$).

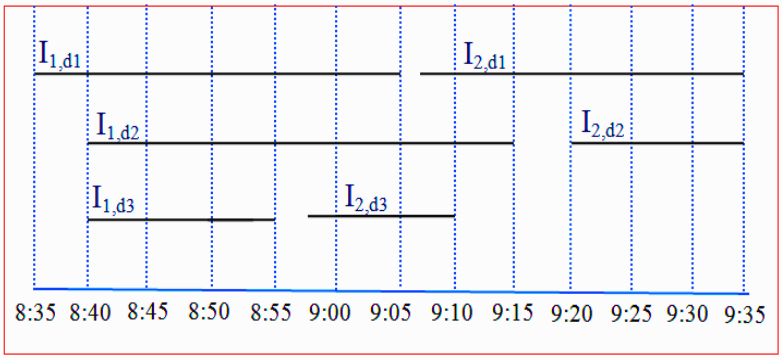
6) The replica should already be cached and available at r_j before the deadline ($\eta_d^u > t_{lcach}^{jd}$).

The set of requests satisfying the aforementioned restrictions for each road segment $r_j \in R$ can be predetermined. To do so, for each $r_j \in R$, we specify the set of users F_j^d who request d and abide to restrictions (1) and (2). In order to check the remaining restrictions, potential updated versions of each data item $d \in D$, as well as their caching and expiry time need to be determined. Thus, we specify the potential versions of d that can satisfy the requests of the users in F_j^d. Such versions are the ones that will be generated during the period of encounter of the requesters in F_j^d with r_j. This period begins from the start time of their earliest period of encounter with r_j, denoted a, until the end time of the latest period of encounter, denoted b.

As depicted in Figure 3.2(a), in order to specify the potential versions of a data item d, we use the version available at the current time, denoted l_c^d, whose generation time $g_{l_c}^d$ is known. Starting from l_c^d, we determine the subsequent versions. The first version that can satisfy the requests of F_j^d is the first one whose expiry time $> a$, whereas the last version is the one whose expiry time $> b$. For example, as shown in Figure 3.2(a), the candidate versions of d_1 that can be assigned to requesters satisfying restrictions (1) and (2), are determined. The start and end time of the earliest and latest periods of encounter of such requesters with r_j are given by a and b, respectively. The generation time of the current version, l_{cur}, is known to be at 8:05. The TTL of d_1 is 30 min. Thus, l_{cur} expires at 8:35, marking the generation of



(a) Candidate versions of d_1.



(b) Intervals for the updated versions of d_1, d_2, and d_3.

Figure 3.2: An example illustrating the process of determining the spatiotemporal caching slots at road segment r_j.

the next updated version l_{next} which will expire at 9:05. This replica is considered the first candidate version of d_1 that can be cached at r_j, since its expiry time is greater than a. The version l_{next+1} is the last candidate since its expiry time exceeds b.

Once the potential versions are determined, the lifetime of each version at the corresponding road segment is specified. Thus, for each road segment, we define a set of intervals A_j^d for every data item. Each interval starts from the time at which the corresponding version is to be cached at r_j, denoted t_{lcach}^{jd}, until the time it expires,

denoted t^d_{lexp}. Note that the data center sends each version for caching once it is generated. Thus, t^{jd}_{lcach} is equal to $g^d_l + \Delta_{jduration}$, where $\Delta_{jduration}$ is the amount of time required for the road segment (i.e., parked vehicle) to receive the replica from the data center. The value of $\Delta_{jduration}$ is calculated as the length of the shortest path to r_j divided by the average propagation speed. As depicted in Figure 3.2(b), once the versions of d_1 at r_j are determined, the set of their intervals are defined. The defined intervals represent the spatiotemporal caching slots to which the requests can be assigned. Since d_1 has two versions, the two intervals I_{1d_1} and I_{2d_1} are defined. For each interval $k \in A^d_j$, and based on the aforementioned restrictions (3-6), we define the set of users belonging to F^d_j that consider k a feasible spatiotemporal caching slot. This set is denoted B^d_{jk}.

Another important restriction is ensuring that the cache capacity limit at each road segment is not exceeded. The cache capacity of a road segment is only affected by the replicas of different data items assigned to it in parallel. For instance, in Figure 3.2(b), intervals I_{1d_1}, I_{1d_2}, and I_{1d_3} overlap. Thus, if each of them has at least one request assigned to it, the consumed cache capacity would be 3. Hence, for each road segment $r_j \in R$, we define O_j as the set of sets ψ_j, where ψ_j is the set of intervals that overlap in r_j. In Figure 3.2(b), this is given by O_j=$\{\{I_{1d_1}, I_{1d_2}, I_{1d_3}\}, \{I_{2d_1}, I_{1d_2}, I_{2d_3}\}, \{I_{1d_1}, I_{1d_2}, I_{2d_3}\}, \{I_{2d_1}, I_{2d_2}\}\}$.

The objective is to assign the requests to the proper spatiotemporal caching slots, such that the probability of cache hits that guarantee the acquisition of the data within the desired time frame is maximized. As previously mentioned, for each caching slot, the set of requests that consider it feasible for caching is denoted B^d_{jk}.

B^d_{jk} is predetermined based on the assignment restrictions (1-6). The problem is formulated as a 0-1 Integer Linear Program (0-1 ILP), where the decision variable x^d_{ijk} is set to 1 if the request of user i for data item d is assigned to interval k at road segment j, and 0 otherwise. In the objective function, the probability of cache hits is reflected using the probability of encounter of users with the road segments. The reason for this is that such a probability, P^u_j, determines the degree of accuracy of the estimated period of encounter, and thus plays a vital role in assuring the spatiotemporal availability of the assigned replicas. For example, if user u passes all the restrictions relative to a particular spatiotemporal caching slot at r_j, but its probability of encounter with the latter is small, then another caching slot might be a better alternative for u.

$$\max_{x^d_{ijk}} \sum_{d\in D}\sum_{j\in R}\sum_{k\in A^d_j}\sum_{i\in B^d_{jk}} P^i_j x^d_{ijk}$$

s.t.

$$\text{C1:} \quad \sum_{j\in R}\sum_{k\in A^d_j} x^d_{ijk} \leq 1 \qquad \forall i \in U, \forall d \in D$$

$$\text{C2:} \quad \sum_{k\in\psi_j} \delta_{jk} \leq C_j \qquad \forall j \in R, \forall \psi_j \in O_j$$

$$\text{C3:} \quad \sum_{i\in B^d_{jk}} x^d_{ijk} \geq \delta_{jk}(M+1) - M \qquad \forall j \in R, \forall k \in A^d_j$$

$$\text{C4:} \quad \sum_{i\in B^d_{jk}} x^d_{ijk} < \delta_{jk}(M+1) + 1 \quad \forall j \in R, \forall k \in A^d_j, \forall d \in D$$

The aforementioned objective is subject to the constraints C1-C4. Constraint C1 specifies that each request must be assigned to at most one interval. This is to make sure that each request is assigned only once. Constraint C2 indicates that for each $r_j \in R$, the total number of overlapping intervals at r_j that have at least one request

assigned to them should not exceed the maximum cache capacity C_j. This constraint is checked for each set of overlapping intervals $\psi_j \in O_j$ at r_j. Constraints C3 and C4 are artificial constraints designed to verify C2 (i.e., the cache capacity constraint). In order for C2 to be verified, we need to determine the number of overlapping intervals that have at least one request assigned to them. For this purpose, we define an artificial variable, δ_{jk}, which is set to 1 if at least one request is assigned to interval k at road segment j, and 0 otherwise. Thus, if $\sum_{i \in B^d_{jk}} x^d_{ijk} \geq 1$, then $\delta_{jk}=1$. Otherwise, $\delta_{jk}=0$. This can also be expressed as follows: if $\delta_{jk}=1$, then $\sum_{i \in B^d_{jk}} x^d_{ijk} \geq 1$. Otherwise, if $\delta_{jk}=0$, then $\sum_{i \in B^d_{jk}} x^d_{ijk} < 1$. Since it is not possible to include a conditional statement in the formulation problem, constraints C3 and C4 are constructed to serve the same purpose. The variable M in C3 and C4 is set to a large positive value to ensure that it is larger than the maximum possible value of the summation $\sum_{i \in B^d_{jk}} x^d_{ijk}$.

In order to verify that the conditional statement is satisfied by C3 and C4, consider that the value of δ_{jk} is set to zero in both constraints. This results in the following inequality: $-M \leq \sum_{i \in B^d_{jk}} x^d_{ijk} < 1$. Similarly, if δ_{jk} is set to 1 in C3 and C4, the inequality, $1 \leq \sum_{i \in B^d_{jk}} x^d_{ijk} < M + 2$, is obtained. Thus, C3 and C4 serve the desired purpose.

3.3.4 Heuristic-based Proactive Cache Placement

In order to deal with practical real-time requirements, we propose the greedy heuristic algorithm Proactive Caching at Parked Vehicles (PCPV) in order to solve the aforementioned cache placement problem. In PCPV, the data center selects the roads segments for caching based on a selection procedure that is composed of the following

three stages:

a) Grouping users based on data similarity

The data center groups users that request the same data items together. Let G be the set of groups generated by the data center. To determine the sequential order for processing the groups, every group $g \in G$ is ranked based on the popularity of the data requested by its users. The popularity of the data is determined by the number of users that are interested in it. Once the groups are ranked, they are sorted in a descending order based on their ranks. Then, the following stages (b and c) are performed for each group $g \in G$.

b) Determining the feasible road segments and intervals for caching

The same aforementioned time-constraint checkpoints (1-6) indicated in VOPC are tested to determine the road segments that are eligible for caching. A feasibility matrix, denoted $F^{g}_{m \times n \times o}$ is created accordingly. Such a feasibility matrix indicates the spatiotemporal caching slots that are eligible for caching for each user within the group, where $m = |g|$, $n = |R|$, and $o = |A^{d,g}_{j}|$. Note that, as previously mentioned, A^{d}_{j} is the set of intervals at which the potential versions of d can be cached in r_j. In order to do that, for each road segment $r_j \in R$, we perform the same aforementioned steps indicated in VOPC until we determine the set of users in g, denoted $B^{d,g}_{jk}$, that consider the interval $k \in A^{d,g}_{j}$ a feasible spatiotemporal caching slot. The feasibility matrix is then filled based on Eq. 3.17

$$f_{ijk} = \begin{cases} 1 & u_i \in B^{d,g}_{jk} \\ 0 & Otherwise \end{cases} \tag{3.17}$$

c) Assigning the requests to spatiotemporal caching slots

As previously mentioned, the goal is to maximize cache hits by maximizing the probability of encounter of the requests assigned to the designated replicas. This is while maintaining the cache capacity limit and the QoS demanded by users. Note that this QoS is maintained through the aforementioned restrictions that were used to construct the feasibility matrix.

In order to solve the cache placement problem, an iterative procedure is executed as detailed in Algorithm 2. A matrix called the selection matrix, denoted $S^g_{m\times n\times o}$ is created and iteratively updated to select the spatiotemporal caching slots for all users/requests within group $g \in G$.

The selection matrix is filled based on Eq. 3.18, where w is the iteration number, P^i_j is the probability of encounter of user i with road segment r_j. As previously mentioned in VOPC, O_j is the set of sets ψ_j, where ψ_j is the set of intervals that overlap in r_j. Also, $\delta_{jk} = 1$ if at least one request is assigned to a replica at interval k in road segment r_j. Otherwise, $\delta_{jk} = 0$.

$$s^w_{ijk} = \begin{cases} P^i_j & \text{if } s^{w-1}_{ijk} > 0, \text{and} \\ & \sum_{k\in\psi_j} \delta_{jk} \leq C_j, \forall \psi_j \in O_j \\ 0 & \text{Otherwise} \end{cases} \tag{3.18}$$

Let $U^{'}$ be the set of users in $g \in G$ who have not been assigned a replica location yet and $G^{'}$ be the set of users in $g \in G$ who have already been assigned a replica location. Initially, the iteration number w is set to 1 (line 10), and the selection matrix is set to the feasibility matrix (i.e., $s^0_{ijk} = f_{ijk}$) (line 11). Also, $U^{'}$ is set to $g \in G$, $G^{'}$ is empty, and the list of assigned replicas is empty (lines 12-15). For each interval k at each road segment r_j, δ_{jk} is set to 0 if $g = 1$. Otherwise, δ_{jk} is equal to

Algorithm 2 : PCPV-Assigning the requests to spatiotemporal caching slots

Input:
Set of All Road Segments R
Probability of Encounter P_j^i $\forall r_j \in T_i \; \forall i \in U$
Iteration Number w
Cache Capacity at Road Segment r_j, C_j $\forall r_j \in R$
Feasibility Matrix $F^g_{m\times n\times o} = f_{ijk}$ $m = |g|$, $n = |R|$, $0 = |A_j^{d,g}| \; \forall g \in G$

ASSIGN_REQUESTS(g)
Begin
  $w \leftarrow 1$
  $S^0_{m\times n\times o} \leftarrow \{s^0_{ijk}{=}f_{ijk}\}$ // Selection Matrix
  $U^{'} \leftarrow$ g //Set of users in g with no assigned replica location
  $g^{'} \leftarrow \emptyset$ //Set of users in g that have the same replica location
  $G^{'} \leftarrow \emptyset$ //Set of $g^{'}$ with an already assigned replica location
  LR[] $\leftarrow \emptyset$ // List of Assigned Replicas
  for all $j \in R$ **do**
    for all $k \in A_j^{d,g}$ **do**
      if $g = 1$ **then**
        $\delta_{jk} = 0$
      if $g \neq 1$ **then**
        $\delta_{jk} = \delta_{jk}^{prev}$ // Latest value from previous group
  while $U^{'}$ isNotEmpty **do**
    for all $j \in R$ **do** // Columns of Matrix
      for all $k \in A_j^{d,g}$ **do** // layer of Matrix
        for all $i \in U^{'}$ **do** // Rows of Matrix
          if $s^{w-1}_{ijk} \neq 0$ **then**
            Update s^w_{ijk} // Eq. 3.18
            DCH+ $= s_{ijk}$ //Degree of Cache Hits
            DCH_List[] $\leftarrow$ tuple (j, k, DCH)
    highest_DCH $\leftarrow$ the maximum value of DCH
    $r_{\hat{j}\hat{k}} \leftarrow$ the spatiotemporal caching slot with highest_DCH
    $\delta_{\hat{j}\hat{k}} \leftarrow 1$
    for all $i \in U^{'}$ **do**
      if $s^w_{ijk} \neq 0$ **then**
        $g^{'} \leftarrow g^{'} \cup i$
    $G^{'} \leftarrow G^{'} \cup g^{'}$
    $U^{'} = U^{'} \setminus g^{'}$
    LR[] $\leftarrow$ tuple ($g^{'}$, $r_{\hat{j}\hat{k}}$, d^g) // List of Assigned Replicas
    $w \leftarrow w + 1$
  return LR[]
End

its latest value that was assigned during processing the previous group (lines 16-21).

As long as $U^{'}$ is not empty, the following steps are iteratively repeated (line 22):

1) The selection matrix is updated by Eq. 3.18 (lines 23-27).

2) The degree of cache hits over each interval in each road segment, denoted DCH_{r_jk} is calculated as the sum of the selection matrix values for all users in $U^{'}$ over interval k in road segment r_j (lines 28 & 29). The spatiotemporal caching slot selected for caching, denoted $r_{\hat{j}\hat{k}}$, is the one rendering the maximum degree of cache hits (lines 30 & 31). The value of $\delta_{\hat{j}\hat{k}}$ is updated by setting it to one (line 32).

3) The selected spatiotemporal caching slot represents location of the replica for the users who consider it an eligible segment for caching. Thus, the set of users in $U^{'}$ who have non-zero entries in the selection matrix over the selected road segment are grouped together in $g^{'} \in G^{'}$ and removed from the set $U^{'}$ (lines 33-37). A tuple is added to the list of assigned replicas associating $g^{'} \in G^{'}$ to $r_{\hat{j}\hat{k}}$ and the data item to be sent, d^g (line 38). If $U^{'}$ is still not empty and the cache in all road segments is full, a Least Frequently Used replacement policy is used.

4) The iteration number is incremented by one (i.e., $w = w + 1$) (line 39), and the steps 1-4 are repeated. This is done until $U^{'}$ becomes empty. Once this occurs, the list of assigned replicas can be obtained (line 40).

3.3.5 Data Acquisition from Parked Vehicles

Once the data center makes all the cache placement decisions, it sends the replicas to the designated parked vehicles for caching. Note that moving vehicles are the ones that forward the data. When the data center sends a data packet to a parked vehicle to be cached, it also sends a list of the proactive requesters, denoted LPR, that should

be satisfied by the received replica. Upon receiving the data packet, the parked vehicle caches the data and extracts the LPR. Beacon messages are periodically exchanged among neighboring vehicles. Thus, when a parked vehicle receives a beacon message, it checks if the source ID of the message matches one of the IDs in LPR. If so, the parked vehicle sends the replica to the moving vehicle.

In case the requester arrives too early or too late to acquire the data (i.e., the data has not been cached yet or it has already expired), the data center strives to give the requester another chance for data acquisition. To do so, the cache placement procedure is repeated by the data center if new information that could affect the successful delivery of data has come to its knowledge. This includes two possibilities; the first is the actual content's expiry time, and the second is the most recent update on the actual average travel time of users. The latter is accessible by the data center via traffic update services. If any of this two possibilities occurs, the data center re-evaluates whether or not any of the critical time-constraints has been violated by any requester. For example, if the data center finds out that the average travel time it predicted contradicts with the actual travel time, it can determine whether some requesters have missed their designated replicas. Accordingly, it will re-examine the caching procedure to make an attempt to provide them with another replica while they are still on the move. Note that moving vehicles that forward the data to and from the data center keep a copy in their caches as well. Thus, they can also act as data providers if they are opportunistically encountered.

3.4 Performance Evaluation

In this section, we use VOPC to quantify the potential gains of the heuristic-based predictive caching approach PCPV. We also evaluate the performance of the predictive approach compared to a promising representative of the Broadcast-based Proactive Caching (BPC) approach. BPC is the scheme in [23]. It is implemented while ignoring the effects of the bandwidth issues that render the broadcast-based approach unsuitable for social media access applications. In addition, we compare our proposed PSO-LSTM prediction technique to the Gradient Descent-based LSTM model (GD-LSTM). Thus, we implement two versions of VOPC and PCPV; one using PSO-LSTM and another using GD-LSTM. These versions are referred to as VOPC-PSO and PCPV-PSO, as well as VOPC-GD and PCPV-GD. Furthermore, in order to assess the effect of taking varying weather conditions into consideration in the prediction model, we evaluate the prediction accuracy of both GD-LSTM and PSO-LSTM compared to the long-term GD-based LSTM model proposed in [82], which has not taken the weather into consideration. We refer to the latter as GD-LSTM-NW. Note that this LSTM model has been shown to outperform other prominent time series prediction techniques, including ARIMA [82].

We use the following performance metrics: 1) the average delay experienced starting from the time a request is generated until the data packet is received, 2) the packet delivery ratio, which is the ratio of data packets successfully acquired by users to the total number of generated data packets, 3) the cache hit ratio, which is the ratio of requests satisfied by caching nodes to the total number of received data packets, and 4) the satisfaction ratio, which is the ratio of data packets received before the specified deadline to the total number of data received.

In addition to the aforementioned metrics, two more metrics are used in order to assess the prediction accuracy of the prediction model. The first metric is the RMSE, given by Eq. 3.11, and the second is the coefficient of determination, referred to as R^2, which is another prominent statistical measure for evaluating the performance of prediction models. R^2 is defined as the percentage of variance in a dependent variable that is caused by its relationship with an independent variable [81]. It is given by Eq. 3.19, where $SS_{Regression}$, and SS_{Total} are the squared sum of the regression error, given by Eq. 3.20, and the squared sum of the total error, given by Eq. 3.21, respectively. Note that x_i, and $\hat{x}_i$ are, respectively, the actual and predicted values of the average travel time, while $\bar{x}$ is the mean value of the data points.

$$R^2 = 1 - \frac{SS_{Regression}}{SS_{Total}} \tag{3.19}$$

$$SS_{Regression} = \sum_{i=1}^{Q} (x_i - \hat{x}_i)^2 \tag{3.20}$$

$$SS_{Total} = \sum_{i=1}^{Q} (x_i - \bar{x})^2 \tag{3.21}$$

3.4.1 Simulation Setup

Simulations are conducted using the NS-3 network simulator [64]. NS-3 is integrated with Gurobi [65] to generate the optimal solution in VOPC. Table 3.1 summarizes the simulation parameters. Simulations are performed over a 6×6 road grid topography, comprised of 120 edges. Realistic vehicular mobility traces are created using the SUMO traffic simulator [86]. The network consists of $600/km^2$ moving vehicles, 200 of which are requesters. The simulation period is set to 2000 seconds. The transmission

Table 3.1: Simulation parameters of VOPC, PCPV, LSTM-GD, and LSTM-PSO.

Simulation Parameters	**Value**
Dimensions of the Road Topography	6×6
Number of Edges	120
Simulation Time	2000 sec
Communication Technology	IEEE 802.11p WAVE
Communication Range	200 m
Beacon Interval	1 sec
Number of Requesters	200
Number of Public Figures	50
TTL of Contents	5-7 minutes
Number of Parked Vehicles	240
Cache Capacity Percentage (θ)	60%
Percentage of Road Segments with Parked Vehicles	100%
Deadline Factor (β)	10 seconds
Number of Hidden Layers	1
Number of Hidden Units (H)	12
Number of Epochs in GD	50
Number of Epochs in PSO	30
Number of Particles	40
Learning Rate in GD	0.1
Inertia Weight (ω), c_1, and c_2	0.5, 1.5, and 1.5

range set to 200 m and the beacon interval set to 1 seconds. We use the IEEE 802.11p WAVE standard as a representative of vehicular communication technologies, but all the proposed schemes in this thesis can work regardless of the adopted underlying communication technology. Traffic updates are reported to the data center every 3 minutes. In BPC, data is broadcasted every 8 seconds.

Based on the Zipf-like distribution with a skewness factor=0.5, the interest generation is distributed among 50 public figures, each of which publishes a new post every $5 - 7$ minutes, rendering the previous one obsolete. The number of parked vehicles is 240, uniformly distributed among the road segments. Each parked vehicle

remains at its assigned parking space for the entire simulation period. We vary the dedicated cache capacity at roadside caching units at each intersection in BPC, as well as the total cache capacity of the parked vehicles at each road segment in VOPC and PCPV. This is expressed as the percentage, denoted θ, of the total number of contents that can be requested. The time of each request is assigned a random value that lies within the trip duration of the requester. The deadline of each request is set to the request time+β. Unless otherwise specified, θ and β are set to 60% and 10 seconds, respectively. The percentage of road segments that have parked vehicles residing at them is set to 100%. The threshold th_{dm} is set to 15 seconds.

The number of hidden units H in the three LSTM models is set to 12. The batch size is set to 1, the number of epochs in GD-LSTM and GD-LSTM-NW is set to 50, whereas that of PSO-LSTM is set to 30. The number of particles is set to 40, and the values of ω, c_1, and c_2 are set to 0.5, 1.5, and 1.5, respectively. The learning rate in gradient descent is 0.1, and the number of time steps in the three models is 6. We split the dataset into 80% training data and 20% testing data.

3.4.2 Results and Discussion

In our experiments, we evaluate the performance of BPC, PCPV-GD, PCPV-PSO, VOPC-GD, and VOPC-PSO under varying cache capacity percentage θ, percentage of road segments with parked vehicles κ, and deadline factor β. In addition, we evaluate the prediction accuracy of GD-LSTM-NW, GD-LSTM, and PSO-LSTM. The results acquired are discussed below. Simulation results are presented at a confidence level=90%.

1- Prediction Model Results

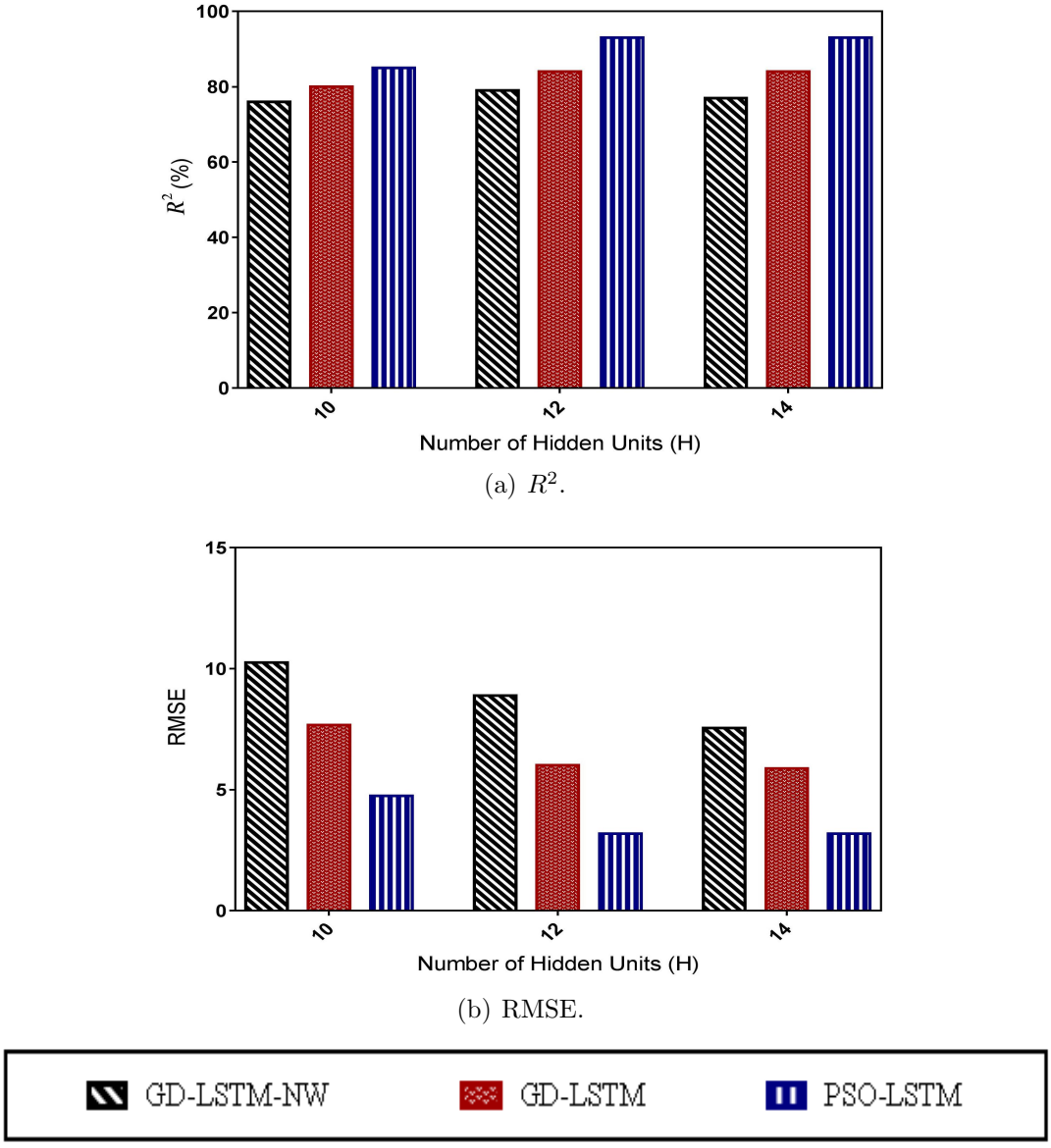
Table 3.2: Prediction performance results.

	R^2			RMSE		
Algorithm	$H = 10$	$H = 12$	$H = 14$	$H = 10$	$H = 12$	$H = 14$
GD-LSTM-NW	76	79	77	10.25	8.89	7.54
GD-LSTM	80	84	85	7.67	6.01	5.87
PSO-LSTM	85	93	93	4.74	3.18	3.18

We first present the results of the three prediction models GD-LSTM-NW, GD-LSTM, and PSO-LSTM in terms of R^2 and RMSE in order to evaluate their performance in terms of prediction accuracy. Table 3.2 shows the numerical values of R^2 and RMSE of each of the three schemes over different number of hidden units H. As shown in Figure 3.3(a), PSO-LSTM yields the highest R^2 among all schemes at $H = 10$, with a difference gain of 9%, and 5% compared to GD-LSTM-NW, and GD-LSTM, respectively. As depicted in Figure 3.3(b), PSO-LSTM also renders the lowest RMSE, with a reduction of 53% and 38% compared to GD-LSTM-NW, and GD-LSTM, respectively. Also, GD-LSTM outperforms GD-LSTM-NW, where it yields a difference gain of 4% in terms of R^2, and a reduction of 25% in terms of RMSE.

The leverage of PSO-LSTM further increases at $H = 12$, where it also renders the highest R^2 and lowest RMSE. In particular, it yields a difference gain of 14% and 9% in R^2, and a decrease of 64% and 47% in RMSE compared to GD-LSTM-NW, and GD-LSTM, respectively. In addition, at $H = 12$, GD-LSTM improves both R^2 and RMSE compared to GD-LSTM-NW, where it renders a difference gain of 5% in R^2, and a decrease of 32% in RMSE.

It is worth mentioning that the leverage gained in PSO-LSTM can be attributed to the fact that PSO has a higher chance of finding the global optima, due to its



(a) R^2.

(b) RMSE.

Figure 3.3: RMSE and R^2 over varying hidden units in GD-LSTM-NW, GD-LSTM, and PSO-LSTM.

ability to expand the search space. This is in contrast to GD, which tends to suffer from local optima entrapment. The disregard of the influential factor pertaining to weather conditions in GD-LSTM-NW further exacerbates the problem. This is since

the model fails to capture the relationship between different travel time occurring at various weather conditions.

When H is equal to 14, PSO-LSTM sustains the same performance in both R^2 and RMSE as those yielded at $H = 12$, while the corresponding values of GD-LSTM slightly increase, with a difference gain in R^2 of only 1%, and a reduction in RMSE of only 2% compared to its previous values at $H = 12$. In contrast, when H is equal to 14, the performance of GD-LSTM-NW slightly deteriorates compared to its previous values at $H = 12$. In particular, GD-LSTM-NW yields a 2% difference loss in its R^2, and a decrease of 15% in its RMSE.

In the remaining experiments presented in this section, we implement our proactive cache placement schemes VOPC and PCPV with both PSO-LSTM and GD-LSTM in order to study the effect of the level of prediction accuracy on the performance of the cache placement procedure. Since the performance of PSO-LSTM has not changed from $H = 12$ to $H = 14$, and that of GD-LSTM has only slightly improved, we set H to 12 in our experiments since the gain is not worth the time cost.

2- The Impact of Cache Capacity

In this experiment, the cache capacity percentage θ is varied from 20% to 100% to assess the performance of our proposed schemes under low, medium, and high cache capacity.

Figure 3.4 shows the performance of BPC, PCPV-GD, PCPV-PSO, VOPC-GD, and VOPC-PSO in terms of average delay over varying percentage of cache capacity. Note that in Figure 3.4, the curves of VOPC and PCPV that are implemented using the same prediction model are circled together. This is to indicate that the schemes

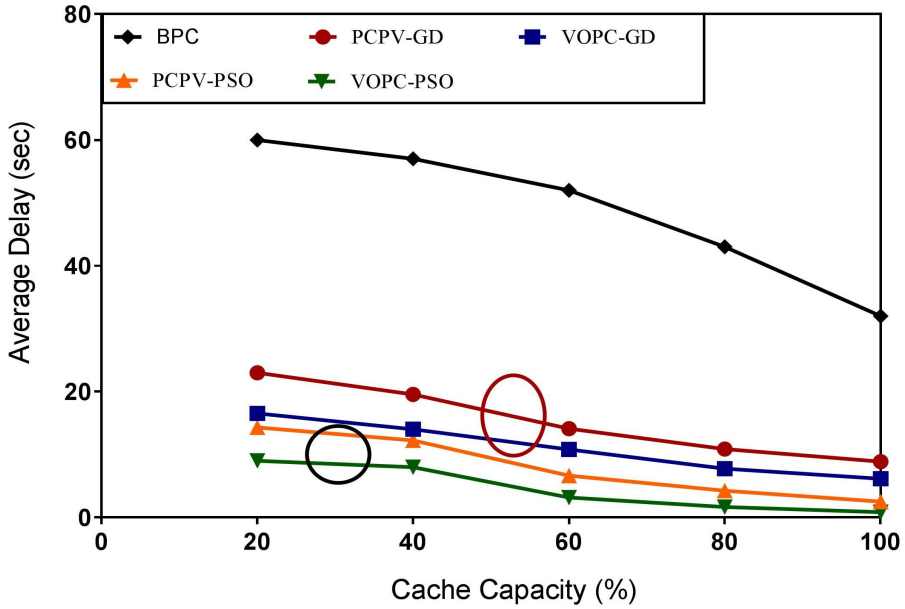


Figure 3.4: Average delay over varying cache capacity (θ).

within the same group are compared to each other. In addition, each circled group is compared to the other group. This applies to every graph in this section that has these circles.

As depicted in 3.4, the predictive approach demonstrates significant reduction in delay compared to BPC, where PCPV-GD, and PCPV-PSO improve the average delay by up to 72% and 81%, respectively, while VOPC-GD and VOPC-PSO improve it by up to 92% and 97%, respectively. This is since requesters in BPC rely on opportunistic encounter with vehicles that happen to have the data. Such an encounter might take some time to occur, which could delay the process of data acquisition. The possibility of this encounter is further prolonged as the cache capacity θ decreases. This is due to the reduction in data availability. In contrast, the predictive approach enables the data to be proactively acquired within a certain time frame specified by

users.

As demonstrated in Figure 3.4, PCPV approaches the optimal solution VOPC as θ increases. In contrast, the gap between the two increases as θ decreases, where the gap between PCPV-GD and VOPC-GD rises up to 28%, while that between PCPV-PSO and VOPC-PSO rises up to 24% when θ is equal to 20%. This indicates a room for improvement in the heuristic solution. The reason is that the lower the cache capacity, the higher the chance of assigning requests to road segments with which the users have more accumulated prediction error. This is since road segments yielding lower accumulated error might already be fully occupied. As a result of the higher prediction accuracy yielded by PSO-LSTM compared to GD-LSTM, the magnitude of the accumulated prediction error at road segments encountered further ahead along the users' trajectory is reduced in VOPC-PSO and PCPV-PSO compared to VOPC-GD and PCPV-GD, respectively. This leads to significant reduction in delay achieved by the former schemes compared to the latter. In particular, PCPV-PSO outperforms PCPV-GD by up to 38%, while VOPC-PSO outperforms VOPC-GD by up to 42%. Thus, VOPC-PSO yields the upper bound on the potential improvement in delay.

Figure 3.5 demonstrates the performance of the schemes in terms of packet delivery ratio over varying percentage of cache capacity. As shown in the Figure, VOPC yields the upper bound on the reachable packet delivery ratio. The predictive approach, including VOPC and PCPV, exhibits a significant improvement over BPC, where PCPV-GD, and PCPV-PSO increase the packet delivery ratio by up to 18% and 24%, respectively, while VOPC-GD and VOPC-PSO improve it by up to 31% and 33%, respectively. This is because the chance that the broadcasted data packets reach the requesters in BPC is largely dependent on the opportunistic encounter with the

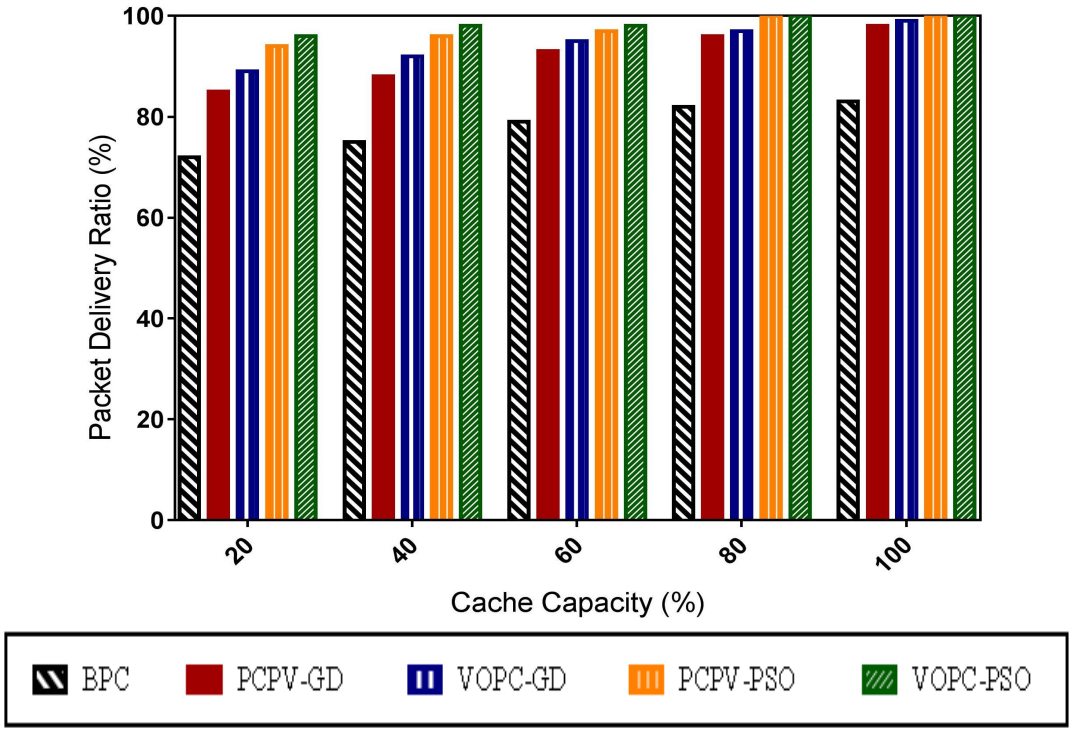


Figure 3.5: Packet delivery ratio over varying cache capacity (θ).

corresponding data holders. In contrast, in both VOPC and PCPV, the data is acquired by requesters as they deterministically pass by the caching nodes, thus significantly reducing the risk of packet loss.

As shown in Figure 3.5, the potential gain of PCPV-GD in terms of packet delivery ratio can reach up to 6% compared to VOPC-GD, while that of PCPV-PSO can reach up to 3% only. The gap between PCPV and VOPC increases as θ decreases. This is due to the increasing risk that requesters pass by the designated caching nodes either before the data has been cached or after it has expired. Such a risk manifests as the cache capacity is reduced due to the higher risk that replicas are placed at road segments with higher accumulated prediction error, since those with lower error

might already be fully occupied. In contrast, as θ increases, this risk is significantly reduced. Thus, PCPV gets closer to the optimal solution. Note that PCPV-PSO and VOPC-PSO outperform their GD counterparts, with an increase of up to 11%, and 8%, respectively. This is due to the higher prediction accuracy of PSO-LSTM compared to GD-LSTM, which in turn reduces the accumulated prediction error at road segments, thus reducing the risk of packet loss.

We conduct the same experiment to evaluate the performance in terms of cache hit ratio over varying (θ). As depicted in Figure 3.6, VOPC and PCPV significantly outperform BPC, where PCPV-GD, and PCPV-PSO improve the cache hit ratio by up to 23% and 29%, respectively, while VOPC-GD and VOPC-PSO improve it by up to 29% and 33%, respectively. This is due to the fact that as θ decreases, data availability at caching nodes decreases in BPC, which makes it harder for requesters to opportunistically encounter caching nodes that have the contents they desire. In contrast, VOPC and PCPV are designed such that contents are mainly acquired from caching nodes as requesters pass by. The only possibility for data acquisition from the data center is when requesters pass by the designated parked vehicle that is responsible for acting as their data provider, but the data is not found. This could occur either because they have departed the road segment before the data has been cached or arrived after it has already expired. In which case, the data center can give them another chance for data acquisition when it re-examines the cache placement procedure or it can directly send them a replica if there is no caching opportunity. The latter option could reduce the cache hit ratio. As previously mentioned, as θ increases, this risk is significantly reduced. Also, even if this risk occurs, there is a possibility that the data might be opportunistically acquired from nearby caching nodes, which

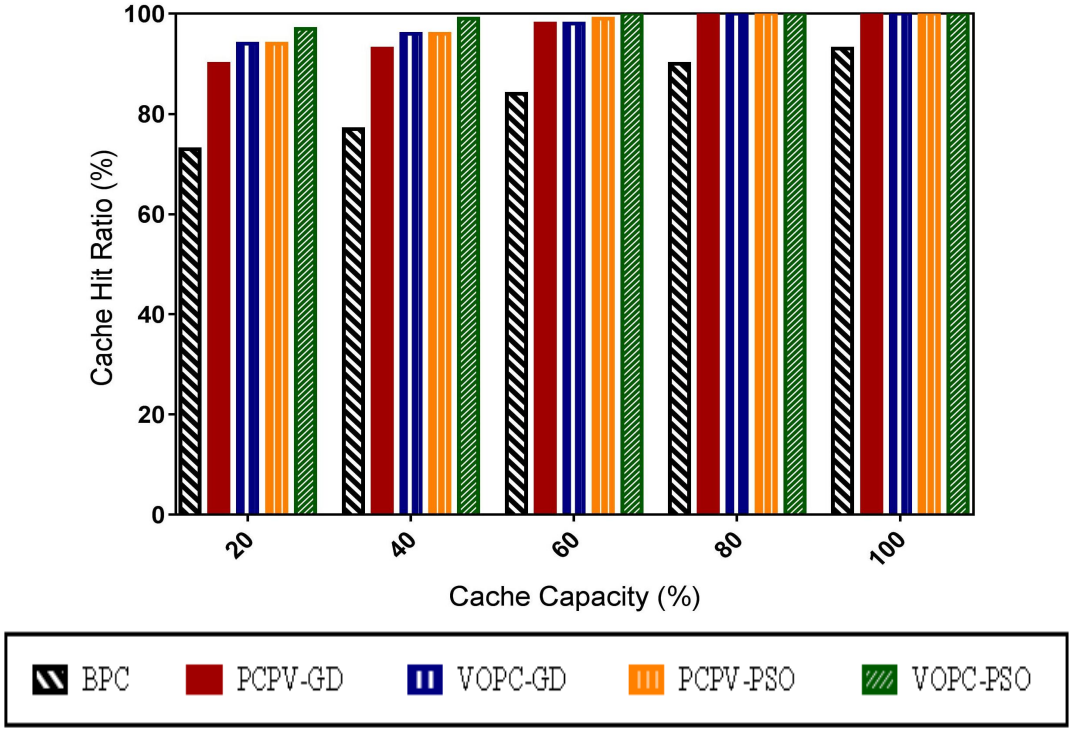


Figure 3.6: Cache hit ratio over varying cache capacity (θ).

could be parked or moving vehicles. Note that such a possibility increases as θ increases, due to the increased data availability at caching nodes.

As shown in Figure 3.6, VOPC provides the upper bound on the potential cache hit ratio increase under varying θ, where VOPC-PSO yields the highest cache hit ratio. Note that the gap between PCPV and the optimal solution is significantly small, where it can reach up to 4% between PCPV-PSO and VOPC-PSO, and 5% between PCPV-GD and VOPC-GD. This indicates that PCPV approaches the optimal solution even under low values of θ. This can be largely attributed to the aforementioned alternatives that enable the data to still be acquired from caching nodes even when requesters reach their designated caching slots too early or too late to be procured.

Since the aforementioned risk is reduced in VOPC-PSO and PCPV-PSO compared to their GD counterparts, they achieve a slightly higher cache hit ratio compared to VOPC-GD, and PCPV-GD, with an improvement of up to 4%, and 5%, respectively.

Figure 3.7 depicts the effect of varying θ on the satisfaction ratio of users. It shows that VOPC and PCPV yield significant improvement in terms of satisfaction ratio compared to BPC, where PCPV-GD, and PCPV-PSO improve it by up to 63% and 81%, respectively, while VOPC-GD and VOPC-PSO improve it by up to 86% and 98%, respectively. This can be attributed to the fact that in contrast to VOPC and PCPV, BPC does not attempt to abide by a specific QoS demanded by users. This makes users more susceptible to acquiring the data later than their desired deadline due to the much higher delay yielded by BPC.

As depicted in Figure 3.7, the satisfaction ratio of the predictive approach significantly increases as θ increases. This can be attributed to the significant reduction in delay yielded by VOPC and PCPV, due to the aforementioned reasons. In addition, as θ decreases, the chance of finding caching nodes that the requesters pass by before their deadline decreases, and if found, they might have a high accumulated prediction error, thus increasing the risk of low satisfaction ratio.

Note that VOPC provides the upper bound on the potential satisfaction ratio increase, where the potential gain of PCPV compared to the optimal solution increases as θ decreases, rising to up to 12% in PCPV-GD compared to VOPC-GD, and 10% in PCPV-PSO compared to VOPC-PSO. In contrast, as θ increases, PCPV gets closer to the optimal solution. The lower gap between PCPV-PSO and VOPC-PSO compared to that between PCPV-GC and VOPC-GC can be attributed to the lower prediction accuracy of LSTM-PSO, which helps reduce the accumulated error, and thus utilize

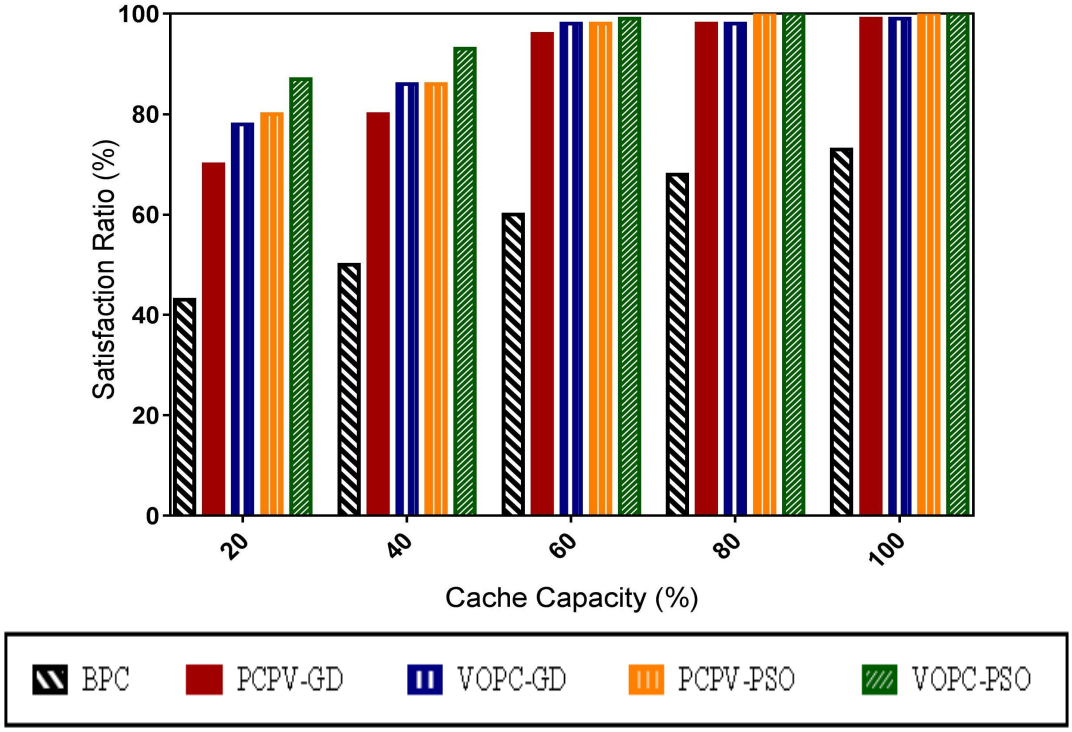


Figure 3.7: Satisfaction ratio over varying cache capacity (θ).

more caching nodes that could help deliver the data before the specified deadline. In fact, VOPC-PSO and PCPV-PSO outperform their GD counter parts, where the former improves the satisfaction ratio by up to 12% compared to VOPC-GD, and the latter improves it by up to 14% compared to PCPV-GD.

3- The Impact of the Percentage of Road Segments with Parked Vehicles

In this experiment, the percentage of road segments that have parked vehicles residing at them, denoted κ, is varied from 20% to 100% in order to study the effect of the scalability of road segments that are available for caching on the performance of the schemes.

Figure 3.8 demonstrates the effect of varying κ on the average delay. As shown in

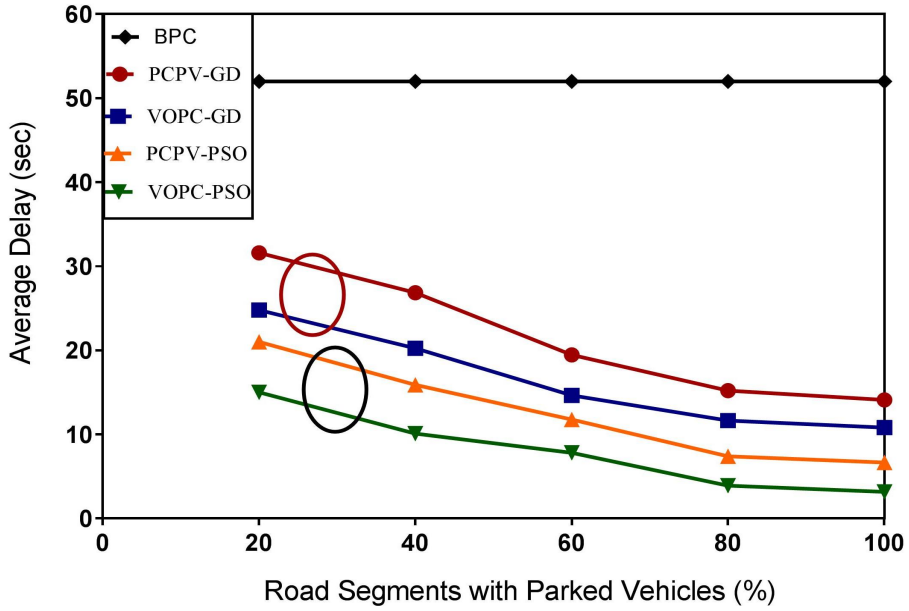


Figure 3.8: Average delay over varying percentage of road segments with parked vehicles (κ).

the Figure, varying κ has no effect on the performance of BPC. This is since BPC does not use parked vehicles for caching. Due to the same aforementioned reasons, the predictive approach significantly improves the delay compared to BPC, where PCPV-GD, and PCPV-PSO improve it by up to 73% and 79%, respectively, while VOPC-GD and VOPC-PSO improve it by up to 87% and 94%, respectively. Note that both VOPC and PCPV yield higher delay as κ decreases. This can be attributed to the fact that decreasing κ limits the number of road segments available for cache selection. This can force the data center to place replicas at road segments with which the requesters have low probability of encounter (i.e., high accumulative prediction error). This poses the risk of having the data cached at road segments that requesters encounter after their specified deadline. This risk further increases as the prediction

accuracy decreases. This explains the leverage gained by VOPC-PSO and PCPV-PSO compared to their GD counterparts, where the former improves the delay by up to 40% compared to VOPC-GD, while the latter improves it by up to 34% compared to PCPV-GD.

As shown in Figure 3.8, the gap between PCPV and the optimal solution VOPC increases as κ decreases, reaching up to 25% between PCPV-GD and VOPC-GD, and 22% between PCPV-PSO and VOPC-PSO, thus indicating a room for improving the delay in the heuristic solution. This can be attributed to the fact that the aforementioned risk is handled by PCPV on a group-by-group basis, where it divides the users into groups and process them sequentially to allocate the requests to spatiotemporal caching slots. Thus, a replica might miss a better caching slot merely because a different group of requests have been handled first, and that slot has already been occupied. In contrast, the optimal solution can handle all requests at once. Accordingly, it can optimally allocate the requests to reach the global solution. Note that the gap between PCPV and VOPC significantly decreases as κ increases, where the heuristic solution gets closer to the optimal.

Figure 3.9 shows the effect of varying κ on the packet delivery ratio, where BPC outperforms PCPV-GD, VOPC-GD, and PCPV-PSO by 11%, 5%, and 2%, while it renders a 3% lower packet delivery ratio compared to VOPC-PSO when κ is equal to 20%. As κ increases, the predictive approach starts to improve the packet delivery ratio compared to BPC. In particular, PCPV-GD, and PCPV-PSO improve it by up to 18% and 20%, respectively, while VOPC-GD and VOPC-PSO improve it by up to 23% and 26%, respectively. This is because as κ decreases in the predictive approach, the risk of having requesters that do not pass by any road segment that has parked

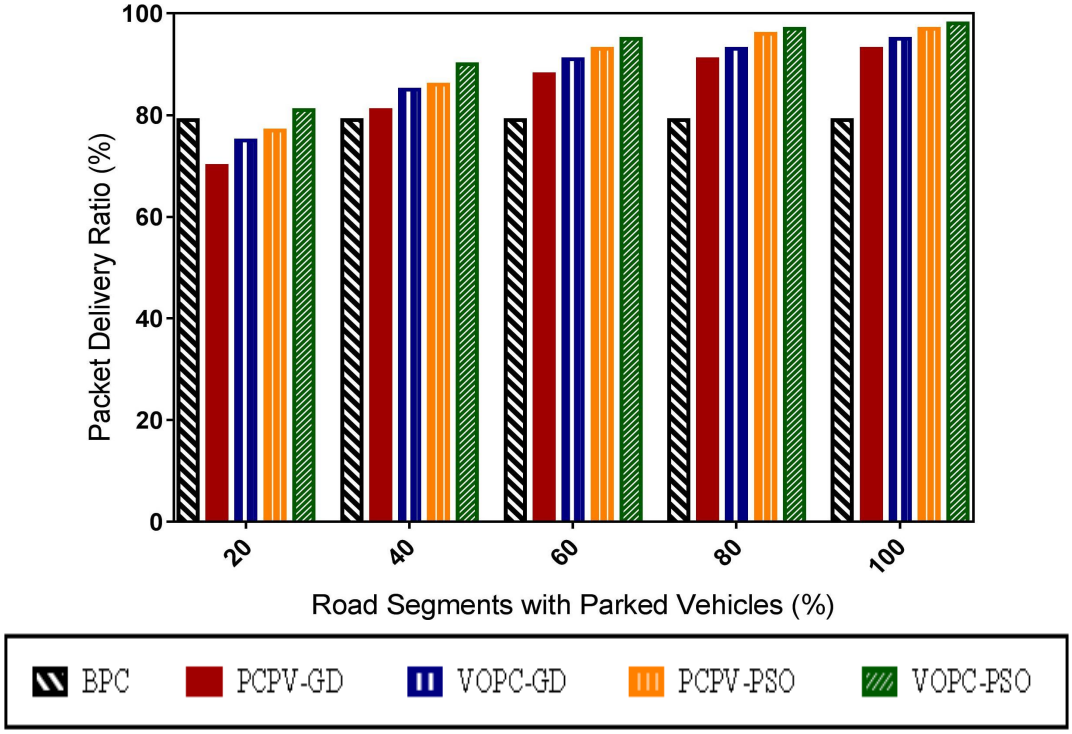


Figure 3.9: Packet delivery ratio over varying percentage of road segments with parked vehicles (κ).

vehicles increases. This forces data acquisition to occur either from the far-away data center or from caching nodes that are opportunistically encountered. Data acquisition from the data center suffers from the risk of dropping the packet due to failure to reach the requester. This could occur due to the movement of the requesters and the risk of erroneous estimations of their location, as well as the lack of a stable residence for the replicas to be cached along the route of the requesters. Note that, as the number of road segments available for cache selection decreases, the risk of allocating replicas to road segments that have high accumulative prediction error increases. This increases the risk of caching the data at road segments that requesters pass by either before the data has been cached or after it has already expired. Consequently,

more reliance on opportunistic encounter with caching nodes, or resorting to the data center for data acquisition takes place. Since the broadcast-based approach increases data availability at caching nodes by broadcasting the data, which increases the possibility of opportunistic encounter, it renders a higher packet delivery ratio at $\kappa = 20\%$. However, as κ increases, the aforementioned risks decrease, which enables the predictive approach to render a much higher packet delivery ratio than BPC.

As shown in Figure 3.9, VOPC-PSO yields the highest upper bound on the potential packet delivery ratio. Also, VOPC-PSO and PCPV-PSO outperform their GD counterparts, where the former improves the packet delivery ratio by up to 8% compared to VOPC-GD, while the latter improves it by up to 10%. This is due to the high prediction accuracy rendered by PSO-LSTM compared to GD-LSTM, which helps reduce the accumulated prediction error of the period of encounter at road segments. Note that PCPV tends to approach the corresponding optimal solution as κ increases, while the gap between the two solutions increases as κ decreases, reaching up to 7% between PCPV-PSO and VOPC-PSO, and up to 5% between PCPV-PSO and VOPC-PSO, indicating only a slight room for improvement in the heuristic solution.

In Figure 3.10, the impact of varying κ on the cache hit ratio is depicted. It is shown that as κ increases, the cache hit ratio increases. As previously mentioned, data acquisition in the predictive approach is mostly achieved by parked vehicles as requesters pass by the corresponding road segments. However, as κ decreases, the risk of resorting to the data center for data acquisition increases. This is due to the increased possibility of the aforementioned risks. When κ is equal to 20%, BPC slightly outperforms PCPV-GD, with an increase of 2%, while VOPC-GD and PCPV-

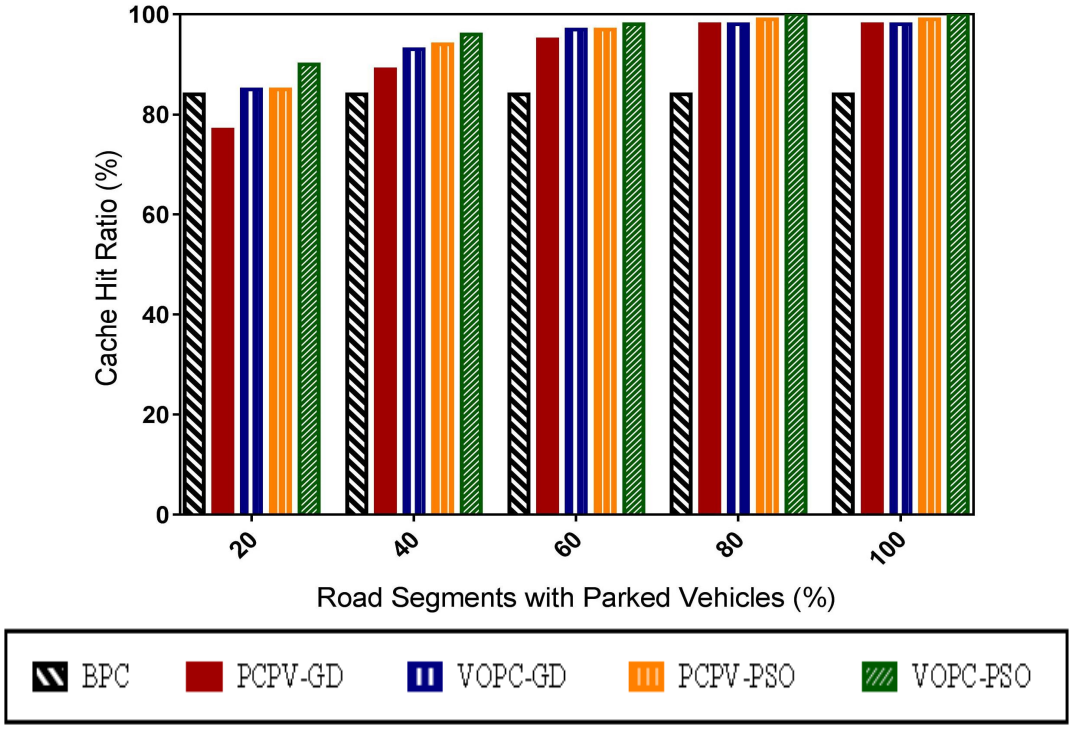


Figure 3.10: Cache hit ratio over varying percentage of road segments with parked vehicles (κ).

PSO improve the cache hit ratio by only 2%, and VOPC-PSO improves it by 7% compared to BPC. This is due to the fact that some requesters do not pass by any of the road segments that have parked vehicles for caching. The higher improvement rendered in VOPC-PSO is particularly attributed to its higher prediction accuracy compared to its GD counterpart, and the fact that it seeks the optimal solution. The higher prediction accuracy reduces the risk of requesters passing by their corresponding caching road segments when the data is not there, thus reducing the risk of having to resort to the far-away data center. As κ increases, all the predictive approach schemes start to significantly outperform BPC, reaching an improvement of up

to 17%, 17%, 18%, and 19% in PCPV-GD, VOPC-GD, PCPV-PSO, and VOPC-PSO, respectively.

As shown in Figure 3.10, PCPV-PSO improves the cache hit ratio by up to 10% compared to PCPV-GD, while VOPC-PSO improves it by up to 7% compared to VOPC-GD. This is due to the aforementioned reasons. In both cases, PCPV tends to approach the optimal solution as κ increases, with a gap of 0% at $\kappa = 100\%$. In contrast, this gap increases as κ decreases, reaching up to 7% between PCPV-GD and VOPC-GD, and 10% between PCPV-PSO and VOPC-PSO. This indicates a slight room for improving the heuristic solution.

In Figure 3.11, the effect of varying κ on the satisfaction ratio is demonstrated. As shown in the Figure, the satisfaction ratio in BPC is not affected by κ, since it does not involve the use of parked vehicles. In contrast, as κ increases, the satisfaction ratio increases in the predictive approach. This can be attributed to two reasons. The first reason is the fact that as κ decreases, the number of available road segments in the cache selection process decreases, which forces the data center to assign replicas to road segments with which users have high accumulative prediction error. This means that more requests are assigned to road segments with which the corresponding users have inaccurate estimation of their period of encounter. This increases the risk of data acquisition after the specified deadline, which reduces the satisfaction ratio. The second reason is the fact that lower values of κ increases the risk of the data center not being able to find caching slots for more users, since they do not pass by any of the road segments that have parked vehicles. This forces data acquisition to occur from the data center, or a caching node encountered along the request forwarding path. The prolonged delay from the data center due to the movement of requesters,

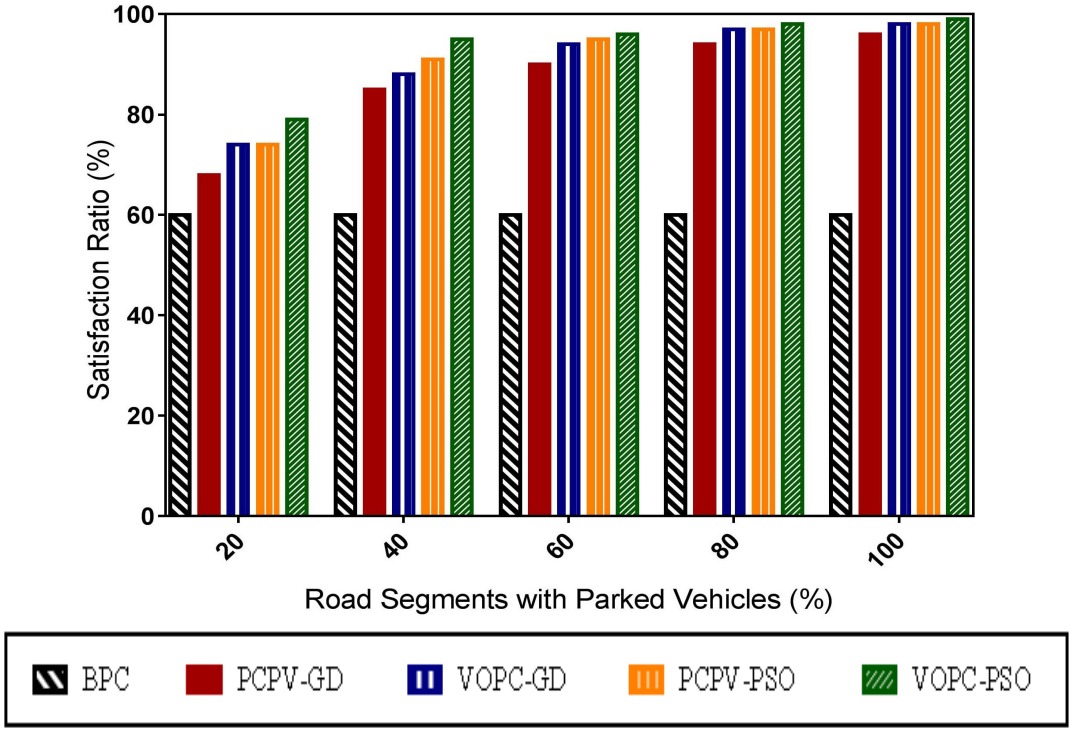


Figure 3.11: Satisfaction ratio over varying percentage of road segments with parked vehicles (κ).

further decreases the satisfaction ratio. Note that as κ increases, data availability increases, which means that even if the aforementioned risk occurs, the possibility of encountering a nearby caching node increases, which increases the satisfaction ratio.

Figure 3.11 shows that the predictive approach significantly outperforms BPC. In particular, it is shown that the satisfaction ratio increases by up to 60%, 63%, 63%, and 65% in PCPV-GD, VOPC-GD, PCPV-PSO, and VOPC-PSO, respectively, compared to BPC. This can be attributed to the fact that data acquisition in BPC does not abide to a specific time frame, and it tends to be prolonged due to its reliance on opportunistic encounter with caching nodes that happen to have the data. This increases the risk of acquiring the data after the deadline. In contrast, the predictive

approach enables data acquisition to occur in a deterministic way while abiding to a specific QoS.

As shown in Figure 3.11, VOPC-PSO and PCPV-PSO outperform their GD counterparts, where the former improves the satisfaction ratio by up to 7% compared to VOPC-GD, and the latter improves it by up to 9% compared to PCPV-GD. This can be attributed to the higher prediction accuracy rendered by PSO-LSTM compared to GD-LSTM, which reduces the aforementioned risk. In both cases, the gap between PCPV and VOPC is reduced as κ increases, thus enabling it to get significantly closer to the optimal solution. However, as κ decreases, this gap tends to increase, reaching up to 9% between PCPV-GD and VOPC-GD, and 7% between PCPV-PSO and VOPC-PSO. This implies that there is a slight room for improvement in the heuristic solution.

4- The Impact of the Deadline Factor

In this experiment, the deadline factor, denoted β, is varied from 10 to 50 seconds in order to study the effect of the deadline specified by users. In order to evaluate this effect when the cache capacity is low in particular, we set the percentage of cache capacity θ to 20%. Note that since BPC does not abide to any specified deadline, it remains the same throughout all the following results.

Figure 3.12 depicts the effect of varying β on the average delay. It shows that as β increases, the average delay increases in the predictive approach. This can be attributed to the fact that increasing β increases the possibility of the data center assigning replicas to road segments that requesters pass by within a longer time frame. This particularly occurs when road segments with which users have earlier time of encounter have already been occupied. This can lead to data acquisition at

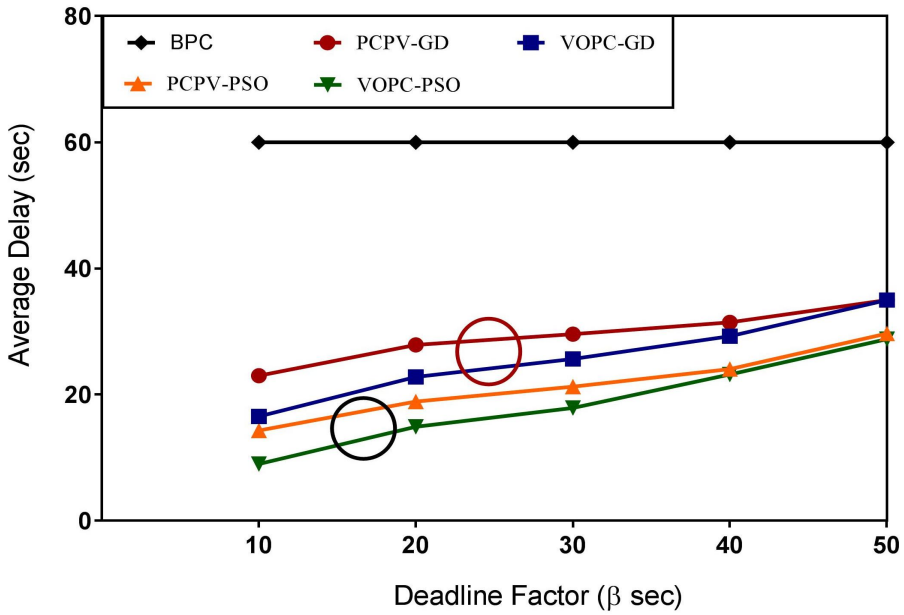


Figure 3.12: Average delay over varying deadline factor (β).

a later time. Due to the higher prediction accuracy rendered by VOPC-PSO and PCPV-PSO compared to their GD counterparts, VOPC-PSO reduces the delay by up to 46% compared to VOPC-GD, while PCPV-PSO reduces it by up to 38%. Note that VOPC-PSO renders the highest upper bound on the potential average delay improvement.

As shown in Figure 3.12, the gap between PCPV and the optimal solution significantly decreases as β increases, as it performs much closer to the optimal solution. In contrast, this gap increases as β decreases, reaching up to 28% between PCPV-GD and VOPC-GD, and up to 22% between PCPV-PSO and VOPC-PSO. This can be attributed to the fact that the aforementioned risk tends to increase in PCPV, since it handles the cache allocation problem on a group-by-group basis. Thus, a better solution for a later group of requesters might no longer be available. This leads to

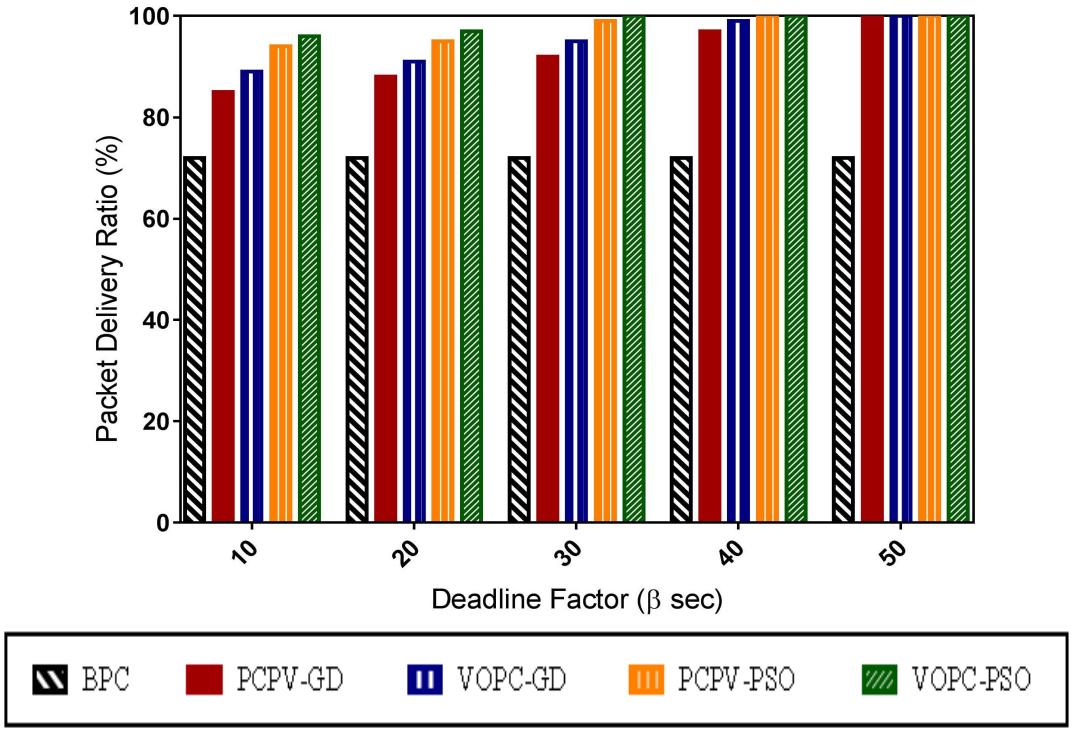


Figure 3.13: Packet delivery ratio over varying deadline factor (β).

yielding higher delay compared to VOPC. Note that the predictive approach significantly outperforms BPC, with an improvement of up to 62%, 73%, 76%, and 85% in PCPV-GD, VOPC-GD, PCPV-PSO, and VOPC-PSO, respectively. This is since the increase in delay rendered in the predictive approach is still guided by the specified deadline, as opposed to leaving it completely without any form of control in BPC.

Despite the effect of increasing β on increasing the average delay, there are several benefits gained from such an increase. One of these benefits is in terms of the packet delivery ratio, where as shown in Figure 3.13, the packet delivery ratio increases as β increases. This can be attributed to the fact that increasing β can increase the number of feasible road segments that can be used for caching for each request. This reduces

the risk of not being able to serve a request because all available spatiotemporal caching slots are either already occupied or they violate the user's specified deadline. In addition, it enables the same slot to serve more users, since it is considered feasible by multiple of them. In contrast, when β is too small, this forces the data center to allocate more replicas at different caching slots to serve all the users requesting the same data. This leads to occupying more of the available storage resources, and thus failing to find enough space to serve all requests. This leads to resorting to data acquisition from the far-away data center, which involves the previously discussed risks.

As shown in Figure 3.13, VOPC-PSO and PCPV-PSO outperform their GD counterparts, where VOPC-PSO improves the packet delivery ratio by up to 8% compared to VOPC-GD, while PCPV-PSO improves it by up to 11% compared to PCPV-GD. This is because VOPC-PSO and PCPV-PSO can reduce the risk of requesters passing by their corresponding caching road segments before the data has been cached or after it has expired. This is due to the higher prediction accuracy of PSO-LSTM compared to GD-LSTM. Note that even if this risk occurs, another chance for data acquisition can be given to the corresponding requesters the next time the data center re-examines the cache placement procedure. The increased available cache space due to increasing β can help seize such an opportunity to serve those who could not be accommodated because they reached the designated road segments too late or too early. Thus, increasing β can significantly improve the packet delivery ratio. It is worth mentioning that as β increases, the gap between PCPV and the optimal solution decreases, reaching a 0% gap at $\beta = 50$, while this gap slightly increases at lower values of β, reaching up to 5% between PCPV-GD and VOPC-GD, and 3% between

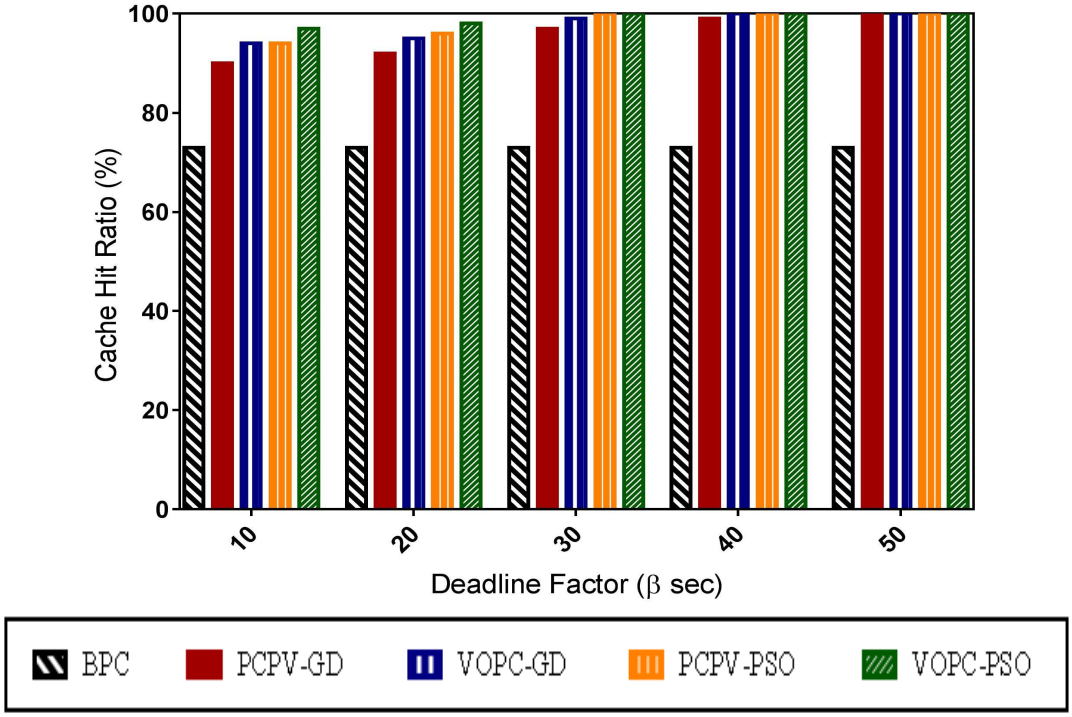


Figure 3.14: Cache hit ratio over varying deadline factor (β).

PCPV-PSO and VOPC-PSO. Accordingly, there is a very slight room for improving the heuristic solution. Note that due to the same previously discussed reasons, the predictive approach outperforms BPC by up to 39%.

In Figure 3.14, the effect of varying β on the cache hit ratio is demonstrated. As shown in the Figure, the cache hit ratio increases as β increases in the predictive approach. This can be attributed to the same aforementioned reasons that contributed to the increase in the packet delivery ratio.

As shown in Figure 3.14, VOPC-PSO renders the upper bound on the potential improvement of the cache hit ratio. Note that VOPC-PSO increases the cache hit ratio by up to 4% compared to VOPC-GD, while PCPV-PSO improves it by up to

5% compared to PCPV-GD. This is due to the higher prediction accuracy yielded by LSTM-PSO compared to LSTM-GD, which reduces the aforementioned risk. Meanwhile, the gap between PCPV and VOPC increases as β decreases, reaching up to 5% between PCPV-GD and VOPC-GD, and 4% between PCPV-PSO and VOPC-PSO. This occurs at $\beta = 10$ seconds, but as β increases, PCPV gets much closer to the optimal solution, until reaching a 0% gap starting from $\beta = 40$ seconds. This is since, even though PCPV handles the request allocation process on a group-by-group basis, the previously discussed risk can still be handled when the cache placement procedure is re-examined. This is since at higher β, there is a higher possibility of benefiting from this second chance of data acquisition due the increase in the utilization of the caching storage resources. Finally, as shown in Figure 3.14, the predictive approach can improve the cache hit ratio by up to 35% compared to BPC.

The effect of varying β on the satisfaction ratio is demonstrated in Figure 3.15. It is shown that as β increases, the satisfaction ratio increases in the predictive approach. This is because, as β increases, the window of opportunity for data acquisition increases, which can reduce the effect of erroneous predictions. As a result, the predictive approach can improve the satisfaction ratio by up to 120% compared to BPC. Note that the upper bound on the potential satisfaction ratio improvement is yielded by VOPC-PSO. As shown in Figure 3.15, VOPC-PSO and PCPV-PSO increase the satisfaction ratio compared to their GD counterparts, where VOPC-PSO renders an improvement of up to 12% compared to VOPC-GD, and PCPV-PSO yields an improvement of up to 14% compared to PCPV-GD. This is because the higher prediction accuracy provided by PSO-LSTM enables it to reduce the risk of the requesters passing by their allocated caching slots after the deadline. In addition, since VOPC

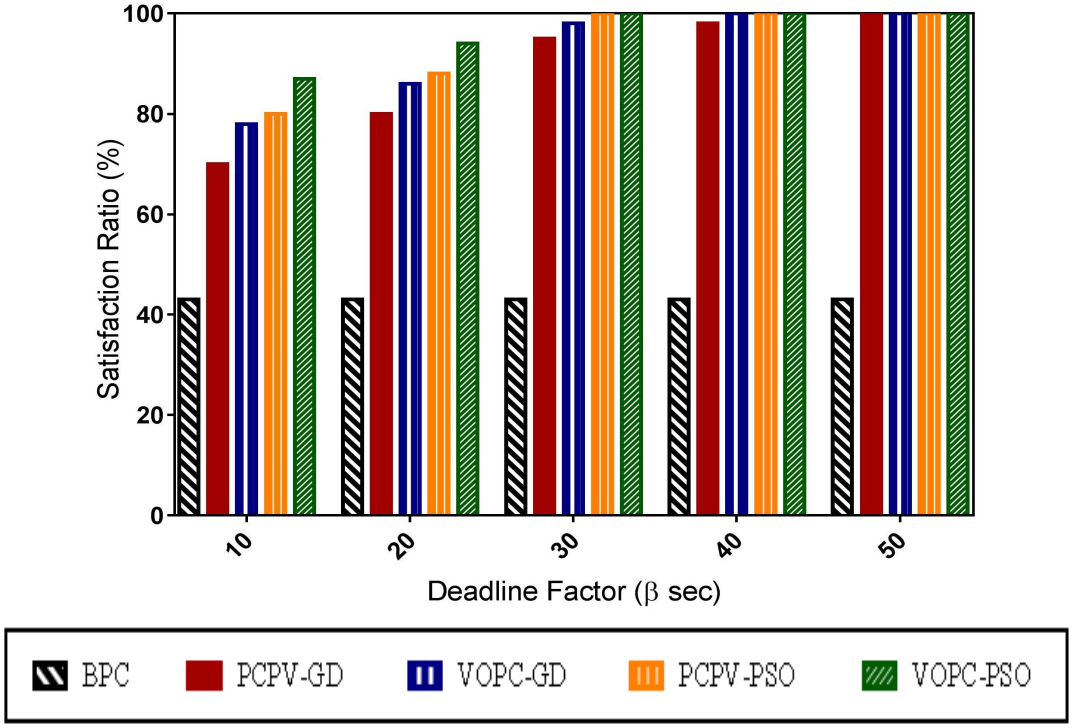


Figure 3.15: Satisfaction ratio over varying deadline factor (β).

seeks the optimal solution by maximizing cache hits that abide to the demanded time frame of the requesters, the gap between the heuristic solution, PCPV, and VOPC increases as β decreases, reaching up to 11% between PCPV-GD and VOPC-GD, and 9% between PCPV-PSO and VOPC-PSO. This implies a room for improving the heuristic solution. In contrast, this gap decreases as β increases, till it reaches a 0% gap between PCPV and the optimal solution at $\beta = 50$.

3.5 Summary

In this chapter, we proposed a predictive proactive caching framework that uses parked vehicles to pre-cache the data at the proper time and place for users to proactively procure as they pass by. This framework relies on the fact that some users

exhibit a predictable behavior as a result of following a daily routine. In order to exploit such a predictable behavior, the proposed framework includes a prediction module, as well as a cache placement module. In the former, the travel time of users at each road segment along their trajectory is estimated in order to be fed into the latter. This is in order to enable informed proactive cache placement decisions to be made by the data center.

In the prediction module, we proposed a PSO-based LSTM model that takes the effect of weather conditions into consideration. The prediction module also enables the estimation of a personalized travel time for each user. In the cache placement module, we have introduced VOPC as a benchmark that allows researchers to quantify the potential gains of predictive proactive caching schemes, evaluate their performance, and certify if there is a possibility for improvement. It presents an optimal solution to pre-cache the data at parked vehicles so as to be proactively procured by users within a specific time frame. In VOPC, the caching problem has been formulated as an Integer Linear Programming (ILP) optimization problem. Furthermore, we proposed a greedy heuristic approach that takes into consideration all the time-constraint issues required to select the most suitable road segments for caching. To maximize cache hits in both VOPC and PCPV, the probability of encounter of proactive users with road segments along their trajectory is also taken into consideration.

We evaluated the performance of the proposed prediction model in terms of prediction accuracy, and we compared it to two gradient descent-based LSTM models that are implemented with and without considering the weather conditions, namely GD-LSTM and GD-LSTM-NW. Simulation results have shown that PSO-LSTM outperforms both models in terms of $RMSE$ and R^2. We also implemented VOPC

and PCPV using both PSO-LSTM and GD-LSTM to evaluate the effect of the level of prediction accuracy on the cache placement process. In addition, we compared the predictive proactive caching approach (i.e., VOPC and PCPV) to the baseline broadcast-based approach. Simulations have shown that the former significantly outperforms the latter in terms of delay, packet delivery ratio, cache hit ratio, and satisfaction ratio. We have also demonstrated the benefit of VOPC in quantifying the potential gains of the heuristic-based predictive scheme.

Chapter 4

Tracking-based Content Discovery in VANETs

4.1 Introduction

One of the important decisions in caching is the one made during the cache discovery process [12]. Discovering where the content is cached is the first process that needs to be performed when a node requests any content. Most caching schemes in VANETs focus on the cache placement policy and ignore the cache discovery process. In fact, they mostly assume the use of a server-based cache discovery process, where cache discovery relies on the opportunistic encounter with a caching node en-route to the data center [12]. This significantly restricts the search space, and can lead to failure in locating caching nodes, which can eventually reduce cache hits.

Many existing cooperative cache discovery schemes in different network paradigms, including MANETs [12] and ICNs [13], depend on some form of information exchange for tracking cached contents [12]. Such information can be used to navigate requests towards nearby caching nodes rather than blindly directing them towards the far-away data center. This type of schemes is referred to as "tracking-based schemes" [12]. Such

schemes can expand the search space compared to server-based schemes, and can sustain lower overhead and delay compared to broadcast-based schemes, where requests are flooded [12]. However, in dynamic networks, preserving up-to-date tracking information might necessitate the exchange of excessive number of messages, which could trigger massive overhead [12]. Hence, information exchange is typically limited to neighboring nodes only in the majority of tracking-based schemes in MANETs [12]. This restriction can still limit the search space and can thus reduce cache hits [12]. Also, despite its irrefutable leverage in MANETs and ICNs, cooperative cache discovery has been rarely explored within the context of VANETs [12]. This can be attributed to the intensively dynamic nature of vehicles, which curtails the lifetime of cached content information and stimulates recurrent instabilities in caching decisions, including cache discovery decisions [12].

In order to tackle the aforementioned problems, we propose the Cooperative Content Discovery (CCD) scheme. In CCD, we utilize the static nature of parked vehicles to create a rather stable residence for cached content information. We do so to keep the information received from encountered vehicles alive at road segments for later use. In addition, we leverage such a static nature to provide a more stable tracking service of the movement of moving vehicles, including that of caching nodes. This further increases the lifetime of the cached content information. We rely on beacon messages that are typically exchanged periodically between neighboring nodes [12], as well as the mobile and static nature of moving and parked vehicles, respectively, to diffuse cached content information within the network. This diffusion occurs as parked and moving vehicles exchange certain information upon encounter, including their cached content information, via beacon messages. This information diffusion

helps expand the search space.

To the best of our knowledge, CCD is the first cooperative cache discovery scheme within VANETs that uses a tracking-based policy and extends the search space beyond the neighborhood scope. However, such an expansion is still restricted by the extent of the available trails that moving vehicles leave to the tracking service provided by roadside parked vehicles. In order to expand the search space beyond such restrictions, we propose the Prediction-Assisted Cooperative Content Discovery (PACD) scheme. In PACD, we enable vehicles to predict the location of mobile caching nodes based on partial knowledge of the trajectory of their ongoing trips, as well as the historical trajectories/trips of other moving vehicles that have similar movement patterns. All possible data providers are then dynamically ranked based on their proximity to the requester, as well as an entropy measure that assesses the uncertainty of their estimated positions. In addition, we reduce the amount of overhead associated with the exchange of cached content information by employing the use of bloom filters [14]. To the best of our knowledge, PACD is the first tracking-based cooperative cache discovery scheme in VANETs that relies on vehicles trajectory prediction to dynamically discover caching nodes closer to the requester.

Vehicles trajectory prediction has been considered a pivotal building block in many smart-mobility services, such as traffic management and hazard warning systems [106]. However, most existing techniques either detect individual movement patterns based on the historical trajectories of the subject vehicle [107]-[113], or they extract collective movement patterns from the historical trajectories of all available vehicles in the training set [110]-[112]. In the former, predictions rely on vehicles having some form of a repetitive pattern, which is not always the case [107, 114].

In the latter, predictions are made at too coarse a granularity and fail to account for the similarity between various trajectories [106]. In this thesis, we propose a novel clustering-based trajectory prediction scheme that builds a separate prediction model for each cluster of vehicles that share similar trajectories. In the proposed trajectory prediction scheme, we cluster similar trajectories using the Any Relation Clustering Algorithm (ARCA), which is a soft clustering algorithm that enables an object to belong to more than one cluster [115]. We then use the Mixture Transition Distribution-Probit (MTD-Probit) model to train each cluster [116].

In order to evaluate the performance of the proposed CCD and PACD schemes, we implement them along with the proposed cooperative cache placement scheme, which will be presented in Chapter 5. Thus, their performance evaluation is provided in Chapter 5. The remainder of this chapter is organized as follows. In Section 4.2, we overview some related work. In Section 4.3, we provide a detailed description of the proposed scheme CCD. In Section 4.4, we present the proposed scheme PACD. In Section 4.5, we summarize the discussion.

4.2 Related Work

An overview of the related work in cooperative cache discovery has already been discussed in Section 2.3 in Chapter 2. Thus, in this section, we only highlight some of the related work in vehicles trajectory prediction.

In [107] and [108], trajectory predictions are performed by capturing individual movement patterns based on the historical trajectories of the subject vehicle [107, 108]. However, such schemes work well only if the vehicles exhibit a frequent

routine that can be exploited in the prediction process [106]. In [110] and [112], predictions are made by exploiting all historical trajectories in the training set to extract collective movement patterns, by using a T-pattern tree, and a recurrent neural network, respectively. Such schemes perform predictions at too coarse a granularity and fail to capture the right scope of similarity among various trajectories [106].

In [106], [117], and [118], the authors consider clustering various trajectories based on their similarity, and then constructing a separate prediction model for each cluster. This helps improve the prediction accuracy compared to the collective-based approach [106]. Among the most commonly used similarity measures are the Minimum Edit Distance (MED) [106], and the Dynamic Time Warping (DTW) [117, 119]. However, MED can lack sequential context sensitivity in some situations [119, 121], and DTW is sensitive to noises that manifest in the training data [106, 119]. In addition, most of these schemes adopt a hard clustering algorithm, which does not allow any given trajectory to be assigned to more than one cluster [106]. A first-order Markov chain model [117], or a variable-length Markov chain (VLMC) model [106, 118] is used to train each cluster. The former disregards the historical states and focuses on current information only, thus reducing the prediction accuracy [120]. The latter can parsimoniously model high-order Markov chains, which take a sequence of historical states into consideration, and can thus ameliorate accuracy. However, VLMCs tend to overlook the frequency of a given sequence in the historical data [120]. Furthermore, they are capable of providing parsimonious efficiency only if there is enough number of branches that possess similar probability distributions to be aggregated in the tree-based approach [120].

In our proposed prediction algorithm incorporated into PACD, we strive to resolve

the aforementioned problems by presenting a novel clustering-based prediction algorithm that employs the Mixture Transition Distribution-Probit (MTD-Probit) model [116]. The MTD-Probit model is an improved variation of the MTD model [120]. It constructs parsimonious high-order Markov chains with significantly reduced parameters compared to the fully parameterized model, while considering the frequency of any given sequence in the training data [116]. It has been used in some problems, such as predicting stock market trends [122, 123], and has been shown to achieve high prediction accuracy [116]. In addition, in contrast to most existing schemes, we incorporate the use of the soft relational clustering algorithm ARCA [115] to enable trajectories to belong to more than one cluster. Furthermore, we use the XXDice-similarity coefficient [121] to determine the similarity between trajectories. It has been demonstrated in different text matching applications that the XXDice-similarity coefficient can yield higher precisions than other similarity measures, including MED [121].

4.3 Cooperative Content Discovery (CCD)

In this section, we provide a detailed description of the system model, as well as the tracking procedure, and the cooperative cache discovery process of CCD.

4.3.1 CCD System Model

Let U be the set of requesters. Each requester $u \in U$ can be interested in a certain data item $d \in D$, where D is the set of data items that can be requested. Each data item $d \in D$ is assigned a unique name, as the case in named data networking [13], and is coupled with a time-to-live TTL_d. This TTL reflects the estimated duration

before a given public figure generates a new post, rendering the previous one obsolete. Upon creating a data packet, the data center (i.e., the original data provider) appends its header with the corresponding TTL of the data. The expiry time of the cached content can be determined based on this TTL. A set of road segments R denotes all road segments in the network. A road segment $r_k \in R$ represents a directed edge e_{ij} between two different intersections I_i and I_j, where $e_{ij} \neq e_{ji}$. Let V be the set of moving vehicles in the network. Each moving vehicle $v \in V$ has a trajectory tr^v, where $tr^v = r_1, r_2, ..r_k$ represents the sequence of road segments traversed by vehicle v along its ongoing trip.

The data center recruits moving and parked vehicles for the caching service by offering some incentives, such as free parking spaces. Vehicles can have access to a plethora of traffic statistics via navigation services. This includes the traffic density and average speed of vehicles at every road segment. Changes in this traffic statistics are triggered by major traffic updates that occur at various traffic checkpoints throughout the day (example: morning, afternoon, rush hour, and evening). Each moving vehicle v_i knows the trajectory of its ongoing trip via the navigation service that it has. Typically, each vehicle periodically sends its current position, as well as its speed, and heading to its neighbors via beacon messages [5]. Note that the current road segment at which v_i is located, denoted r^i_{cur}, can be determined from its current position. In the proposed scheme, along with r^i_{cur}, each moving vehicle is also willing to share the next road segment to which it is heading, denoted r^i_{next}, with vehicles.

Parked vehicles located at each road segment are grouped into parking clusters, created using a similar scheme to the one in [11]. In order to increase the diversity of cached contents during the cache placement process (which will be explained later in

the next chapter), a cluster head (CH) is elected to manage the caching decisions at all parked vehicles within its cluster. Thus, the CH is responsible for maintaining the cached content information within its cluster. It is also responsible for interchanging such information with neighboring vehicles through the exchange of beacon messages. Hence, the nearest parked vehicle to the entrance of the road segment is the one chosen to act as the CH. This is to guarantee that the necessary information are interchanged between moving and parked vehicles once the former move into the road.

In CCD, each vehicle maintains a List of Cached Data (LCD), containing the names of its own cached data items. Using beacon messages, each vehicle sends its last encounter information to all of its neighbors. If the sender of the beacon message is a CH, the last encounter information consists of its position, as well as the LCD of its cluster. If the sender of the beacon message is a moving vehicle v_i, the last encounter information is composed of its LCD, position, speed, heading, and the next road segment to which it is heading r^i_{next}.

A time threshold, denoted $t_{i,next+1,e}$, is defined as the time of departure of vehicle v_i from road segment r_{next+1}, where the latter is the road segment to which v_i is heading after r^i_{next}. Note that e in $t_{i,next+1,e}$ stands for the end of the period of encounter with the road segment (i.e., the time of departure). The time threshold $t_{i,next+1,e}$ is estimated by the node that receives the beacon message, and is also added to the last encounter information. It is used to indicate the upper limit on the time during which v_i is known to be located within close proximity to the last node that knows where it headed, as explained later in details.

Once a vehicle receives a beacon message from v_i, it extracts the corresponding last encounter information, associates it with a timestamp t_i, representing the last

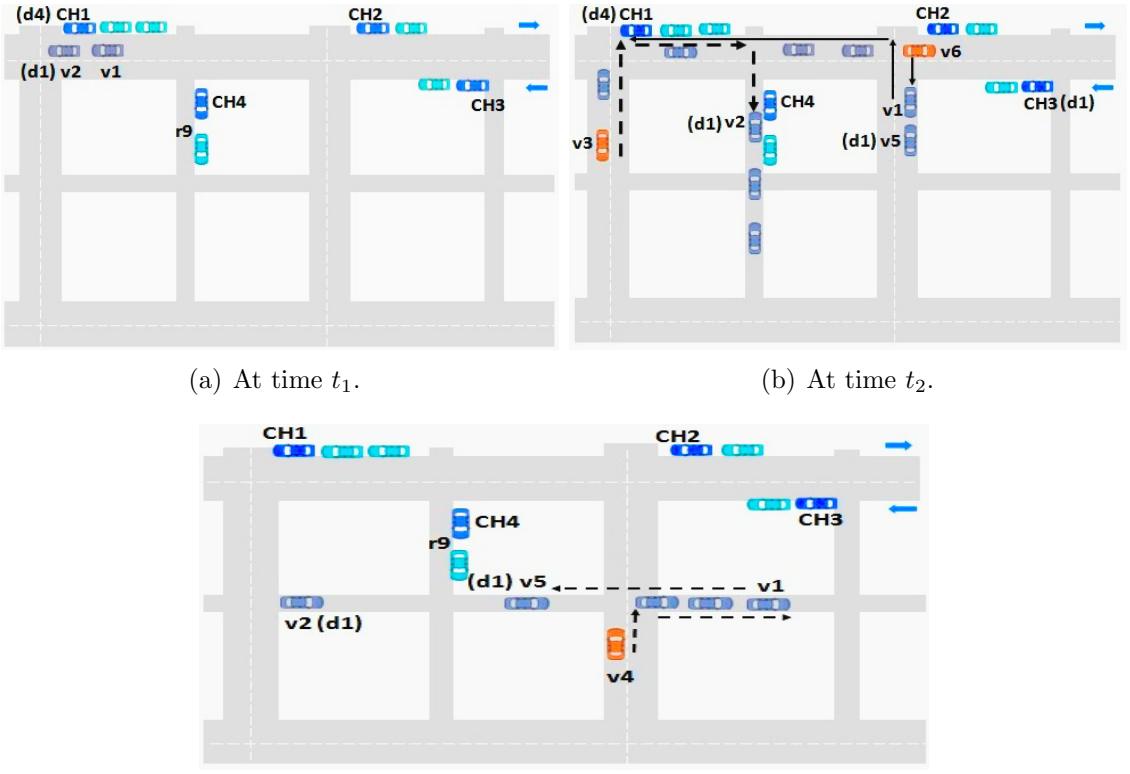
time of encounter with v_i, and sustains it in the Table of Possible Providers (TPP). If the beacon message is received from a moving vehicle, the receiving vehicle calculates the time of departure of v_i from r^i_{cur}, denoted $t_{i,cur,e}$. This time is calculated as the sum of $t_{i,cur,s}$ and the estimated travel time along r^i_{cur} starting from time $t_{i,j,s}$, as given by Eq. 4.1. For any road segment r_j, $t_{i,j,s}$ is the estimated time of arrival of v_i at r_j. For r^i_{cur}, $t_{i,j,s}$ is set to the last time of encounter t_i. The travel time of a vehicle v_i along a road segment r_j at time $t_{i,j,s}$ is denoted $\tau_j^{t_{i,j,s}}$, and is given by Eq. 4.2. The value of $\tau_j^{t_{i,j,s}}$ is calculated based on the length of the road segment, denoted L_j, and the estimated average velocity of vehicles on the road segment at time $t_{i,j,s}$, denoted $\xi_j^{t_{i,j,s}}$. The arrival time of v_i at road segment r_j is equivalent to the departure time of v_i from r_{j-1}, as given by Eq. 4.3. Similarly, the arrival and departure time of v_i to and from r^i_{next}, denoted $t_{i,next,s}$ and $t_{i,next,e}$, respectively, as well as the aforementioned time threshold $t_{i,next+1,e}$, are calculated.

$$t_{i,j,e} = t_{i,j,s} + \tau_j^{t_{i,j,s}} \tag{4.1}$$

$$\tau_j^{t_{i,j,s}} = \frac{L_j}{\xi_j^{t_{i,j,s}}} \tag{4.2}$$

$$t_{i,j,s} = t_{i,j-1,e} = t_{i,j-1,s} + \tau_{j-1}^{t_{i,j-1,s}} \tag{4.3}$$

The receiver of the beacon message adds the estimated departure time from r^i_{cur}, r^i_{next}, and r^i_{next+1} to the TPP for later use. Note that the TPP provides information about the currently and previously encountered vehicles that hold the data in their cache and that can thus act as data providers. Even if the LCD in the received beacon message is empty, the last encounter information is still maintained in the TPP of the receiving vehicle. This is done for location tracking purposes. Parked vehicles



(a) At time t_1.

(b) At time t_2.

(c) At time t_3.

Figure 4.1: An illustrative scenario of CCD.

subscribing to the caching service maintain a similar table. When a node receives a request packet, it ranks each caching node in the TPP based on its proximity to the requester, as well as the age of information. Based on the ranks, the request can be directed to a data holder that is closer to the requester than the current destined data provider. Initially, the latter is the data center. An entry in the TPP about a caching node is considered obsolete if the corresponding data reaches its expiry time.

In order to illustrate the main notion behind the scheme, consider the scenario depicted in Figure 4.1. As shown in Figure 4.1(a), at time t_1, the moving vehicle v_2, which has the data d_1 in its cache, encounters the moving vehicle v_1 and the cluster

head CH_1. The parking cluster of CH_1 has the data d_4 in its cache. The next road segment to which vehicle v_2 is heading is r_9. Based on the received beacon messages, each vehicle updates its TPP. Based on such information, v_1 and v_2 now both know that d_4 is available at CH_1. Also, v_1 and CH_1 are aware that d_1 is available at v_2, and that the latter is heading to r_9 next. At time t_2 (Figure 4.1(b)), vehicles v_1 and v_2 have already moved away. CH_1 receives a request for d_1 issued by the requesting vehicle v_3 and destined to CH_3 (dotted line). Assume that CH_3 is a caching node that was previously encountered by v_3. Upon receiving the request, CH_1 consults its TPP and determines that v_2 is closer to the requester than CH_3. Thus, it directs the interest packet to v_2 (dotted line). Similarly, when v_1 receives a request for d_4 from the requesting vehicle v_6, it consults its TPP and directs the packet to CH_1 (solid line). Meanwhile, v_1 encounters the caching vehicle v_5, which has d_1 in its cache. Accordingly, v_1 updates its TPP to indicate that d_1 is available at both v_2 and v_5. At time t_3 (Figure 4.1(c)), vehicle v_4 sends a request for d_1 to the distant data center. When v_1 receives the interest packet during the forwarding process, it checks its TPP and ranks the previously encountered data holders v_2 and v_5 to select the closest node to the requester. Accordingly, v_5 is selected. Note that if v_2 was selected, it would have been reachable via CH_4, since the latter knows where v_2 has headed after r_9.

4.3.2 CCD Tracking Procedure at Moving and Parked Vehicles

This tracking procedure can be performed by a moving vehicle or a CH in order to track the new position of another moving vehicle v_k that it maintains an entry about in its TPP. In order for a vehicle v_f to do that, it considers the following three cases: 1) The vehicle v_k has not reached r^k_{next} yet. In other words, it is still moving on r^k_{cur}.

This is the case if the current time t_{cur} is less than the estimated time of arrival of v_k to r^k_{next} (t_{cur}<$t_{k,next,s}$). Thus, the new position of v_k can be estimated based on its old position, which is recorded in the TPP of v_f, and the estimated distance that it has traversed on r^k_{cur} since the last time of encounter t_k. This distance is denoted $dist_{k,cur,t_k}$. The latter is calculated based on the vehicle's speed and heading (i.e., velocity vector $\overrightarrow{V_k}$), as given by Eq. 4.4.

$$dist_{k,cur,t_k} = \overrightarrow{V_k}(t_{cur} - t_k) \tag{4.4}$$

2) The vehicle v_k has reached r^k_{next}, and it is still there (i.e., $t_{k,next,s}$≤t_{cur}≤$t_{k,next,e}$). In this case, its new location can be estimated based on its last known position (i.e., the start point of r^k_{next}), and the total estimated distance that it has traversed on r^k_{next} since its time of arrival there $t_{k,next,s}$. This distance is denoted $dist_{k,t_{k,next,s}}$. This estimated distance is given by Eq. 4.5, where $\overrightarrow{V}_{t_{k,next,s}}$ is the velocity vector. This velocity vector is determined based on the heading of v_k on r^k_{next} and the estimated average speed of vehicles on r^k_{next} at time $t_{k,next,s}$.

$$dist_{k,t_{k,next,s}} = \overrightarrow{V}_{t_{k,next,s}}(t_{cur} - t_{k,next,s}) \tag{4.5}$$

3) The vehicle v_k has reached r^k_{next+1} (i.e., t_{cur}>$t_{k,next,e}$). In this case, the new position of v_k can be tracked through the CH at r^k_{next}, denoted CH^k_{next}. This is since it would have information about r^k_{next+1}. Hence, in this case, we set the new position of v_k to that of CH^k_{next}. If v_k has already left r^k_{next+1} (i.e., t_{cur}>$t_{k,next+1,e}$), the same logic is applied and the new position of v_k is set to that of CH^k_{next}. Note that CH^k_{next} has information about r^k_{next+1}, and the cluster head at r^k_{next+1} would also have information

about r^k_{next+2}, and so on.

4.3.3 CCD Cache Discovery at Moving and Parked Vehicles

This procedure is either triggered by a requesting vehicle or a vehicle that is forwarding an interest packet, denoted v_f. Initially, the current data provider to which the interest packet is directed is the data center. Note that the interest packet is associated with the last time of encounter of the requesting vehicle. This indicates the last time at which the requesting vehicle was located at the position included in the packet. Also, the last time of encounter of the current data provider v_c, and its $t_{c,next+1,e}$ (if it is a moving vehicle), are also included in the interest packet. As illustrated in Algorithm 3, the cache discovery procedure is executed as follows:
(a) Upon receiving an unexpired interest packet, the vehicle v_f tracks the most recent location of the requester v_{req}. The purpose of this tracking procedure is twofold. First, since we aim at finding a closer data provider to the requester, we strive to determine the most recently observed position of the latter. Second, we endeavor to alleviate the problem associated with the fact that by the time the data is issued back to the requester, its position might have significantly changed. Thus, the packet might be dropped if the requester cannot be tracked. In CCD, when v_{req} passes by a neighboring node, including a CH, it sends information about the next road segment to which it is heading. Due to the static nature of parked vehicles, it is possible to reach v_{req} by following its trails via the CH in the road segment where it has last been seen, denoted r_{last}. Note that even if the recent position of the requester cannot be closely estimated, it is possible to expedite the process of data access by finding a data holder that is closer to r_{last} than the current data provider. This is since v_{req} is reachable via r_{last}. The tracking procedure works as follows (lines 12-15): when a

Algorithm 3 : CCD Cache Discovery at Moving and Parked Vehicles

Input:
Forwarding Vehicle v
Interest Packet I
Reply Packet rep
Requested Data d
Requesting Vehicle v_{req} //source of I
Current Data Provider v_{cp} //destination of I
Neighborhood list NB

cache_discovery(I)
Begin
  if I is not expired **then**
    if v_{req} is recorded in v_f's TPP **then**
      if $t_{req}^{TPP} \geq t_{req}^{I}$ **then** //t is the last time of encounter
        $NewPos_{req}$ = *track_newPos(I, v_{req})* //Updated Pos.
        $t_{req} = t_{req}^{TPP}$ //Updated-t recorded in TPP
        Update the v_{req} position and t_{req} in I
    if there is d matching I in the cache **then**
      generate a reply rep
      forward rep
    else if any node in NB has d in its cache **then**
      update v_{cp} in I //The neighbor with the cached data
      forward I
    else
      if v_{cp} is a moving vehicle **then**
        if v_{cp} is recorded in v_f's TPP **then**
          if $t_{cp}^{TPP} \geq t_{cp}^{I}$ **then**
            $NewPos_{cp}$ = *track_newPos(I, v_{cp})*
            $t_{cp} = t_{cp}^{TPP}$ //Updated-t recorded in TPP
      if v_f is in the range of v_{cp}'s Position **then**
        if v_{cp} is not in NB **then**
          v_{cp}=data center
      if ID of d matches an entry in v_f's TPP **then**
        determine Ω //Set of possible providers of d in the TPP
        for all $v_i \in \hat{\Omega}$ **do** // $\hat{\Omega} = \Omega \cup$ the data center $\cup\ v_{cp}$
          if v_i is a moving vehicle **then**
            $NewPos_{v_i}$ = *track_newPos(I, v_i)*
            calculate $rank_{v_i}$ using Eq. 4.6
          else
            calculate $rank_{v_i}$ using Eq. 4.7
        Calculate $rank_{max} = \max_{v_i \in U} rank_{v_i}$
        $C = arg\max_{v_i \in \hat{\Omega}} rank_{v_i}$
    update v_{cp}, its position, t_{cp}, $t_{cp,next+1,e}$ in I //if any changed
    forward I
End

vehicle, v_f, receives an unexpired interest packet, it checks if the requesting vehicle is among the vehicles registered in its TPP (i.e., v_{req} has been previously encountered by v_f). If it is, and the recorded time of encounter in the TPP t_{req} is more recent than the time of encounter associated with the interest packet, the requester's new position is estimated according to the information in the TPP (i.e., its position, speed, heading, r^{req}_{next}, and $t_{req,next+1,e}$). Otherwise, its position remains the same as that indicated in the interest packet. In order to estimate the new position of v_{req}, v_f applies the aforementioned tracking procedure. Accordingly, the position of the requester and its associated last time of encounter are updated in the interest packet (lines 16 & 17). Note that when the data is found, the requester's position and its last time of encounter are copied in the reply packet. The requester tracking process is applied by all vehicles along the data delivery path as well.

(b) v_f checks if it has a match of the requested data in its own local cache (i.e., v_f is an intermediate caching node or the destination of the interest packet). If so, the vehicle issues a reply packet and sends it back to the requester (lines 18-20).

(c) If not, v_f checks if any of its 1-hop neighbors, denoted NB, has the data in its cache. If so, it sends the interest packet to it (lines 21-23).

(d) Otherwise, if the packet's destination (i.e., the current data provider) is a moving vehicle, v_f tracks its most recent position. To do so, it applies the same tracking procedure applied for the requester (lines 24-29).

(e) If the estimated position of the current data provider is within the communication range of v_f, but the former cannot be found, then the current data provider is set to the data center. Otherwise, the current data provider remains the same (lines 30-32).

(f) v_f checks its TPP. If an entry matching the name of the requested data is found

in the TPP, it determines the associated set of vehicles in the table that can act as potential providers of the requested data, denoted Ω. The vehicle then assigns a rank to each node $v_i \in \Omega$. This rank assesses the usefulness of the node as a potential data provider and the benefit of forwarding the interest packet towards it rather than the current data provider. The rank is based on two factors: 1) The age of information. That is, the older the information stored about the vehicle, the less the accuracy of its estimated position. In particular, the longer it has been since v_i left its r^i_{next+1}, the further it is from where it can be reached (i.e., CH_{next}), and thus the less reliable its proximity information. Accordingly, if the current time exceeds the estimated time threshold of v_i, denoted $t_{i,next+1,e}$, by more than a certain time step, the rank of v_i is set to zero. As previously mentioned, $t_{i,next+1,e}$, indicates the estimated departure time of v_i from r^i_{next+1}. 2) The second factor based on which the rank is calculated is the estimated distance (in hops) between the vehicle caching the data and the requester, denoted $\hat{d}_{iReq}$. The closer it is to the requester, the higher the rank. Note that the number of hops between v_i and the requester is calculated by dividing the distance d_{iReq} by the communication range. Thus, in order to calculate the ranks, if a vehicle $v_i \in \Omega$ is a moving vehicle, its most recent location must be estimated before calculating its rank. In this case, the vehicle v_f tracks the most recent location of v_i based on the last encounter information registered in the TPP, and using the same aforementioned tracking procedure. Taking the new estimated positions into consideration, the vehicle calculates the rank of each moving vehicle v_i, denoted $rank^m_i$, using Eq. 4.6, where t_{cur} is the current time, t_i is the last time of encounter with vehicle v_i as recorded in the TPP, t_{max} is the most recent time of encounter among that of v_{cp} and all moving vehicles in Ω, d_{min} is the minimum

distance between the requester and all possible providers, ∇ is a certain time step, and α_1 and α_2 are weighting factors set in the $(0, 1]$ range; $\alpha_1+\alpha_2$=1. If v_i is a parked vehicle, its rank, denoted $rank_i^p$, is calculated using Eq. 4.7. In addition, v_f ranks the current data provider v_{cp} using Eq. 4.6 or Eq. 4.7. Note that if v_{cp} is a moving vehicle, its $t_{cp,next+1,e}$, as well as its last time of encounter, are associated with the interest packet. The vehicle also ranks the data center using Eq. 4.7 since it might be closer to the requester, so it might be better to direct the packet to it (lines 33-40).

$$rank_i^m = \begin{cases} 0 & t_{cur} > t_{i,next+1,e} + \nabla \\ \alpha_1 \frac{t_i}{t_{max}} + \alpha_2 \frac{d_{min}}{\hat{d}_{iReq}} & \text{Otherwise} \end{cases} \tag{4.6}$$

$$rank_i^p = \frac{d_{min}}{\hat{d}_{iReq}} \tag{4.7}$$

(g) The maximum rank among that of all $v_i \in \Omega$, the current data provider, and the data center, is then determined. The node that has the maximum rank is selected as the current data provider and the packet's destination is updated. The last time of encounter, as well as the new estimated position of the destination and its $t_{c,next+1,e}$, are also updated in the packet (lines 41-43).

(h) v_f anchors the packet towards the estimated position of the current data provider using the following forwarding procedure (line 44): 1) If there is a CH in the neighborhood of v_f that is more adjacent to the destination than itself, v_f forwards the packet to it. This is to enable the packet to encounter as many CHs as possible to benefit from the information maintained in their TPPs in the discovery process. 2) Otherwise, greedy forwarding is used to direct the packet towards the destination. In greedy forwarding, the nearest neighboring node to the destination is the one to which

the packet is forwarded. Aside from CHs, forwarding occurs using moving vehicles only. However, if v_f fails to find any moving vehicle within its neighborhood, it is possible to forward the packet to parked vehicles.

If v_f is a requesting vehicle, it performs the aforementioned steps (b) and (c). In case of a cache miss, it sets the current data provider to the data center and applies steps (f)-(h).

4.4 Prediction-Assisted Cooperative Content Discovery (PACD)

In this section, we provide a detailed description of the system model of the proposed PACD scheme, the incorporated prediction model, as well as the cooperative cache discovery procedure.

4.4.1 PACD System Model

PACD shares the same system model as CCD, with some additional features. In PACD, along with r^i_{cur}, each moving vehicle is also willing to share the next road segment to which it is heading r^i_{next}, as well as the $\ell - 2$ consecutive road segments that preceded the current one. This provides information about a total of ℓ road segments, including r^i_{cur} and r^i_{next}, that represent a partial trajectory of the vehicle's ongoing trip.

The aforementioned partial trajectory is used for location prediction of mobile caching nodes that any vehicle maintains information about in its TPP. The data center is responsible for the training of the prediction model. For this purpose, we assume that the data center has access to a set $G = tr_1, tr_2, ...tr_N$ of N historical trajectories recorded from previous trips carried by different vehicles. Note that

the proliferation of the use of the Global Positioning System (GPS) and location-acquisition technologies in vehicles, has generated a huge amount of trajectories that can be exploited for this purpose [117]. The data center clusters these historical trajectories based on their movement similarity. It then trains each cluster using the MTD-Probit model to generate the corresponding matrix of ℓ-order transition probabilities. Each of these probabilities reflects the probability of moving to a given road segment given a previous set of ℓ road segments representing a known partial trajectory. The data center sends each of these clusters, along with the transition probability matrix corresponding to each cluster, to moving and parked vehicles subscribing to the service. This process can be done once every few months for example if new training trajectories become available to the data center.

Using its known trajectory, each moving vehicle that has received the aforementioned clusters from the data center, associates its ongoing trip to the clusters that reflect high similar movement patterns to its own trajectory. Along with the aforementioned information included in the exchanged beacon messages, each vehicle includes the ID of the clusters to which it belongs, as well as metadata about its cached contents (i.e., cached content information). During the information exchange process, we use a b-bit bloom filter to reduce the size, and thus the associated overhead of the exchanged cached content information. We use the term last encounter information to refer to all the information included in each beacon message. Any vehicle v_j can use the last encounter information it receives from another vehicle v_i, including the ℓ road segments of its ongoing trip and the clusters to which it belongs, to predict the future locations of v_i via the corresponding maximum transition probabilities associated with the clusters.

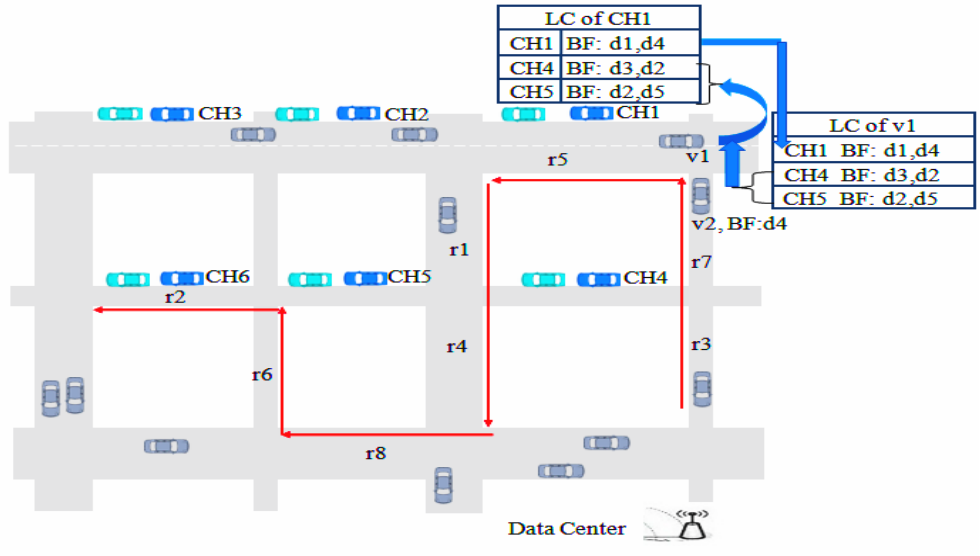
The cooperation range is further extended by adopting the same methodology used in our cooperative cache placement scheme. In this methodology, a data structure called the List of Clusters (LC) is maintained by the CH of each parking cluster. The LC includes the last encounter information of its own cluster, as well as that of other clusters. Note that for CHs, the last encounter information represents their positions and their cached content information. Thus, the LC maintained by a CH includes metadata about the cached contents of the CH's own cluster, as well as that of other parking clusters. The cached content information about other clusters is acquired during information exchange with moving vehicles that have passed by other parking clusters. Note that each entry in the LC is composed of the ID of a CH, and the corresponding b-bit bloom filter representing its cached data items. Moving vehicles preserve their own LCs as well. The LCs are exchanged between CHs and moving vehicles via beacon messages. Entries of LCs are removed once the expiry time of the corresponding data is reached.

Once a vehicle receives a beacon message from v_i, it extracts the corresponding last encounter information, associates it with a timestamp t_i representing the last time of encounter with v_i, and sustains it in its TPP. If the beacon message is received from a moving vehicle, the receiving vehicle gauges the time of departure of v_i from r^i_{cur}, denoted $t_{i,cur,e}$. Note that $t_{i,cur,e}$ is calculated in the same way that was previously explained in the CCD scheme. Similarly, the arrival and departure time of v_i to and from r^i_{next}, denoted $t_{i,next,s}$ and $t_{i,next,e}$, respectively are determined.

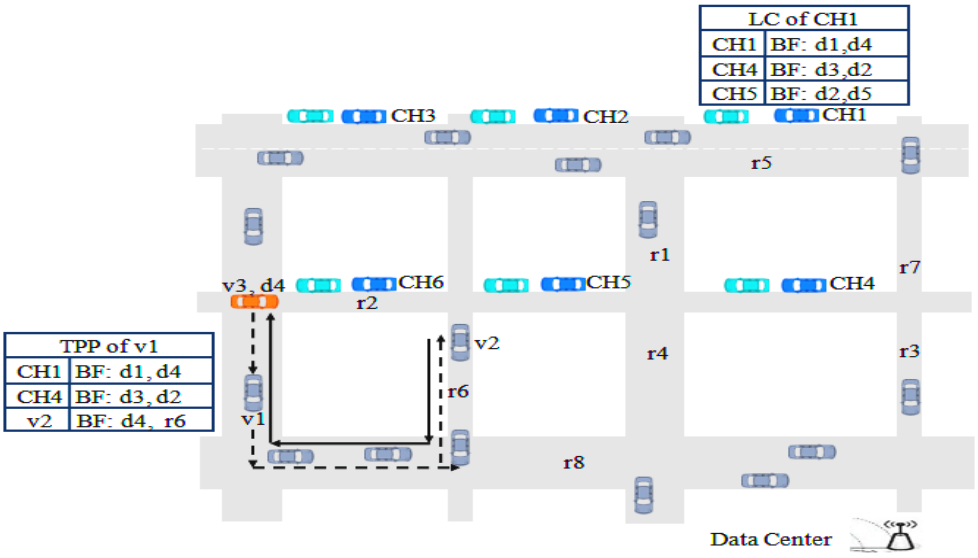
The receiver v_j of a beacon message from v_i adds the estimated departure time from r^i_{cur} and r^i_{next} to its TPP for later use. During the tracking procedure, v_j keeps track of the current location of all moving vehicles that it has record of in its TPP.

Note that if the current time t_{cur} is less than $t_{i,cur,e}$, then v_i is still at the road segment at which it was located at the time of encounter. If $t_{cur}<t_{i,next,e}$, then v_i is at r^i_{next}. Otherwise, the vehicle v_j uses the prediction model to predict the position of v_i at time t_{cur}. If v_i is a parked vehicle, no prediction is required, due to its static nature. When a request packet is received by v_j, it checks its TPP and ranks all possible data providers based on their distance from the requester (i.e., number of hops), as well as the entropy of the estimated position (i.e., level of prediction uncertainty). This procedure is dynamically performed by the requester, as well as each request-forwarding vehicle during the cache discovery process. Note that initially, the request packet is directed to the data center. This cache discovery process continues until the requested content is found.

Figure 4.2 demonstrates an illustrative scenario depicting the cache discovery process. In this scenario, assume that moving vehicle v_1 does not have any data in its cache, and that it has previously encountered CH_4 and CH_5. Accordingly, v_1 maintains the cached content information pertaining to CH_4 and CH_5 in the form of bloom filters (BF) in its LC. The LC of CH_1 has information about the cached contents in its own cluster only, which are d_1 and d_4. As shown in Figure 4.2(a), at time t_1, v_1 encounters both v_2 and CH_1, and they exchange their own cached content information, as well as that maintained in their LCs via beacon messages. This leads to the update of their corresponding LCs as depicted. Consider that vehicle v_2 is moving along the trajectory shown by the solid line, and that it has the data d_4 in its cache, whereas its LC is empty. Upon its encounter with v_1, v_2 sends its cached content information to v_1 in the form of a BF. In addition, v_2 sends the ℓ road segments representing its partial trajectory to v_1. Assuming that ℓ=3, the ℓ road segments of v_2 are r_{prev}=r_3,



(a) At time t_1



(b) At time t_2

Figure 4.2: An illustrative scenario of PACD.

r_{cur}=r_7, and r_{next}=r_5. Also, v_2 includes the set of trajectory clusters with which its own trajectory yields high membership. Based on the received information, v_1 updates its TPP, and predicts the remaining trajectory of v_2. As shown in Figure 4.2(b), at time t_2, v_1 receives a request packet for d_4 from the requester v_3. v_1 first checks its local cache, and when it does not find a match, it checks its own TPP and determines that d_4 can be provided by CH_1 and v_2, so it ranks them based on their proximity to v_3 and the entropy of their location. The current location of v_2 that has been predicted by v_1 is r_6, which is much closer to v_3 than CH_1. Thus, v_1 sends the request packet to v_2 (dotted line), which sends the data packet back to v_3 (solid line).

Based on the aforementioned discussion, and as depicted in Figure 4.3, the system architecture is composed of four modules; the prediction training module, the information exchange module, the tracking module (i.e., prediction module), and the cache discovery module. The first module takes place at the data center, while the remaining modules take place at moving and parked vehicles (i.e., CHs). Note that S, $U(w)$, $[\rho_{i,g}]$, and $[Q_c^{'}]$ shown in Figure 4.3 are the information that each vehicle needs in order to make the necessary prediction later on. They will be discussed later in details. In the next subsections, we discuss each of the aforementioned modules.

4.4.2 PACD Prediction Model Training at the Data Center

The data center clusters the available set of training trajectories G based on their route similarity. It then trains each cluster separately using the MTD-Probit model. Thus, the training procedure involves the following three stages:

Stage 1: Create the similarity matrix

In order to cluster the trajectories, the data center first creates the similarity

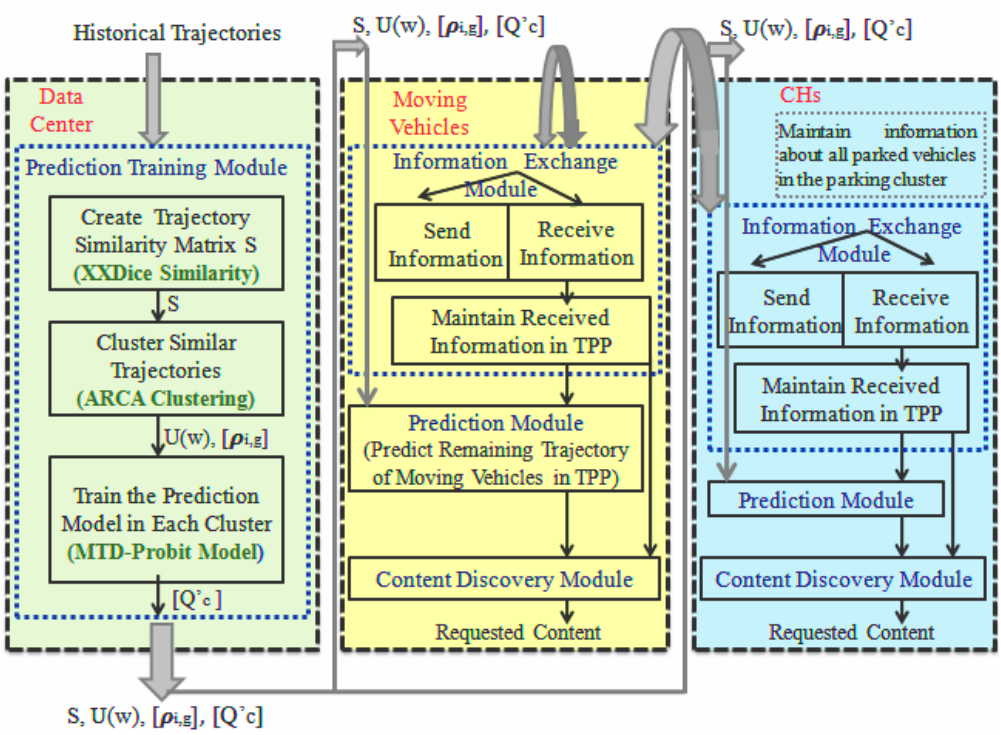


Figure 4.3: PACD system architecture.

matrix $S_{N\times N}$ by calculating the similarity $sim(tr_i, tr_j)$ between each pair of trajectories tr_i and tr_j in the training set G, where $i \neq j$, and $|G|{=}N$. Such a similarity is calculated based on the XXDice Similarity Coefficient (XXDSC) [124], which is the generalized version of DSC [121] [124]. This family of similarity coefficients has been widely used in text matching and DNA sequencing applications to determine the similarity between two sequences [121]. It has been demonstrated that XXDSC can outperform several other measures in terms of similarity precision, including the minimum edit distance and the longest common subsequence measures [121].

DSC, given by Eq. 4.8, is defined as the ratio between twice the number of shared ℓ-grams in both sequences to the sum of the total number of ℓ-grams in each sequence,

where an ℓ-gram is a token (i.e., subsequence) of length ℓ of the complete sequence.

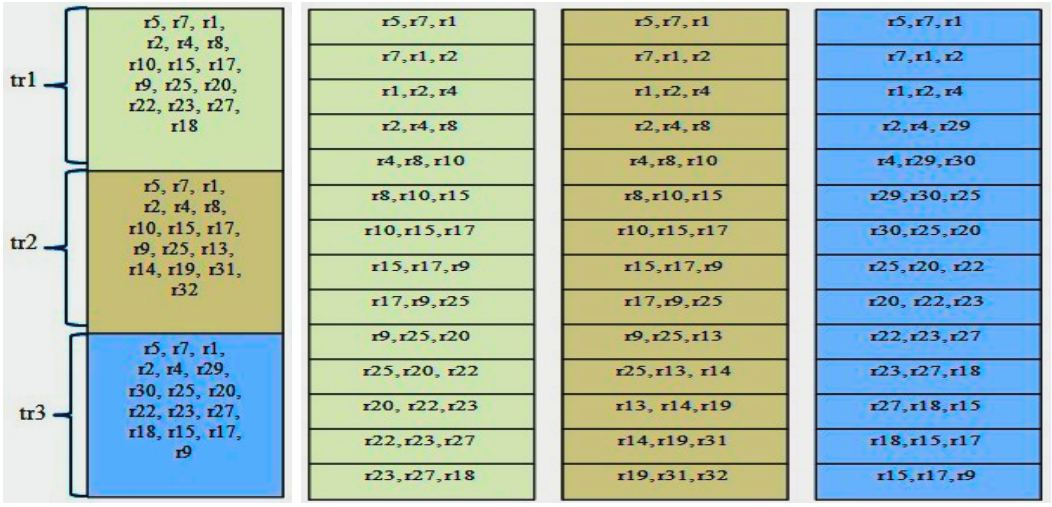
$$\mathrm{DC}(tr_i,\, tr_j) = \frac{2 \times |\ell\text{-grams}(tr_i) \cap \ell\text{-grams}(tr_j)|}{|\ell\text{-grams}(tr_i)| + |\ell\text{-grams}(tr_j)|} \tag{4.8}$$

To improve the precision of DSC, XXDSC was designed to take the order/position of the shared ℓ-grams into consideration [124]. In order to calculate XXDSC(tr_i, tr_j), we define two arrays A and B of size ϱ_{ij} for the two trajectories tr_i and tr_j, respectively, where A and B are the arrays of shared ℓ-grams between tr_i and tr_j, sorted in the order in which they appear in the corresponding sequence, and ϱ_{ij} is the number of these shared ℓ-grams (i.e., ϱ_{ij}=$|\ell\text{-grams}(tr_i) \cap \ell\text{-grams}(tr_j)|$).

For each pair of trajectories tr_i and tr_j in the training set G, where $i \neq j$, the value of XXDSC(tr_i, tr_j) that reflects the similarity $sim(tr_i, tr_j)$, is calculated as given by Eq. 4.9 [124], where $pos_A(A[x])$ and $pos_B(A[x])$ are the positions of the ℓ-gram $A[x]$ in array A and array B, respectively. The value of $sim(tr_i, tr_j)$ is set to 1 if $i = j$.

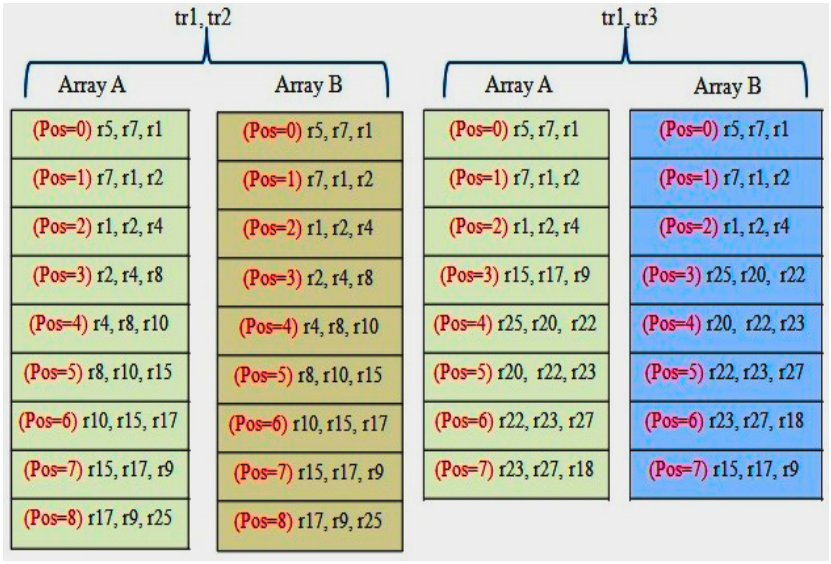
$$\mathrm{sim}(tr_i,\, tr_j) = \frac{\sum_{x=0}^{\varrho} \frac{2}{1+(\mathrm{pos}_A(A[x])\text{-}\mathrm{pos}_B(A[x])\ )^2}}{|\ell\text{-grams}(tr_i)| + |\ell\text{-grams}(tr_j)|} \tag{4.9}$$

An Example demonstrating the process of calculating the XXDice similarity coefficient is depicted in Figure 4.4. Figure 4.4(a) shows the sequence of road segments traversed along the routes represented by each of the three trajectories tr_1, tr_2, and tr_3. In Figure 4.4(b), consider ℓ=3. The ℓ-grams of tr_1 (green, left), tr_2 (brown, middle), and tr_3 (blue, right) are extracted. The number of ℓ-grams in each trajectory is 14. In Figure 4.4(c), for each of the two pairs of trajectories tr_1 and tr_2, and tr_1 and tr_3, the arrays A and B represent the shared ℓ-grams among each pair, organized in the order in which they appeared in the corresponding trajectory. This reflects the



(a) Trajectories tr_1, tr_2, and tr_3

(b) ℓ-grams of tr_1, tr_2, and tr_3



(c) Shared ℓ-grams between tr_1 & tr_2, and between tr_1 & tr_3

Figure 4.4: XXDice similarity coefficient example.

position of each of the shared ℓ-grams in each trajectory. Using Eq. 4.9, the XXDice similarity coefficient between tr_1 and tr_2 is 0.64, while that between tr_1 and tr_3 is 0.36.

Stage 2: Cluster similar trajectories

In this stage, the data center clusters the trajectories in the training set G based on their similarity. This is done while taking into consideration the possibility that some trajectories may belong to more than one cluster. Thus, we use a fuzzy clustering approach. In fuzzy clustering, N objects are clustered into C clusters by determining the degree of membership of each object to each cluster [115]. The Fuzzy C-Means (FCM) clustering technique is considered one of the most prominent and stable fuzzy clustering algorithms in the literature [115][125][126]. However, FCM is only applicable when the objects to be clustered are portrayed as points in a multi-dimensional space [115][125][126].

In order to be able to use fuzzy clustering for objects represented by relationships, typically expressed as pairwise dissimilarity values, a few fuzzy relational algorithms that are based on the FCM technique have been proposed [115][125][126]. One of these algorithms is the Any Relation Clustering Algorithm (ARCA) [115]. In contrast to other existing fuzzy relational algorithms, which impose some restrictions to ensure that the dissimilarity matrix between the objects is an Euclidean matrix [125][126], ARCA does not impose such restrictions [115]. In addition, ARCA has been shown to be more stable and scalable than most existing fuzzy relational clustering algorithms [115].

In fuzzy clustering, the clustering problem is resolved by first detecting C prototypes that are most representative of as many groups of objects, and then constructing a cluster around each of these prototypes [115][125][126]. Similarly, in ARCA, each object (i.e., each trajectory in our case) is defined by a vector of its relational strength with the remaining objects in the data set [115]. A prototype is a trajectory whose relationship with all the trajectories in the data set G is representative of the relationships between a cluster of similar trajectories [115].

ARCA employs an iterative algorithm that partitions the data set by minimizing the Euclidean distance between each object that has a high membership to a given cluster and the prototype of that cluster [115]. The optimal partitioning is deduced by minimizing the objective function in Eq. 4.10 subject to the constraints in Eq. 4.11 and Eq. 4.12, where $(tr_1, tr_2, ...tr_N)$ are the objects (i.e., trajectories) to be clustered, $(\rho_1, \rho_2, ...\rho_C)$ are the prototypes of each cluster, $u_{i,k}$ is the degree of membership of object tr_k to the cluster with prototype ρ_i, ϑ is the fuzzification coefficient which controls the degree of fuzziness of the clusters, and $\delta(tr_k, \rho_i)$ is the deviation between the dissimilarity between tr_k and all the other trajectories in G, and between ρ_i and all the other trajectories in G.

$$J_\vartheta(U, V) = \sum_{i=1}^{C} \sum_{k=1}^{N} u_{i,k}^{\vartheta} \delta^2(tr_k, \rho_i) \tag{4.10}$$

$$u_{i,k} \in [0, 1] \quad \forall i, k \tag{4.11}$$

$$\sum_{i=1}^{C} u_{i,k} = 1 \quad \forall k \tag{4.12}$$

The value of $\delta(tr_k, \rho_i)$ is given by Eq. 4.13, where $s'_{k,g}$ is the dissimilarity between the pair of trajectories tr_k and tr_g, and $\rho_{i,g}$ is the dissimilarity between the prototype

ρ_i and the trajectory tr_g. Note that, the dissimilarity matrix $S'_{N\times N}$ is obtained from the similarity matrix $S_{N\times N}$, where $s'_{k,g}$=$1 - sim(tr_k, tr_g)$.

$$\delta(tr_k, \rho_i) = \sqrt{\sum_{g=1}^{N}(s'_{k,g} - \rho_{i,g})^2} \tag{4.13}$$

We use the standard Lagrange multipliers minimization method to solve the aforementioned problem, where the Lagrangian multipliers are given by λ_k, $1 \leq k \leq N$, and the Lagrange function to be minimized is given by Eq. 4.14. Note that $J(U, V)$ is the objective function given by Eq. 4.10.

$$L(U, V, \lambda_1, \lambda_2 ... \lambda_k) = J(U, V) + \sum_{k=1}^{N} \lambda_k (1 - \sum_{i=1}^{C} u_{i,k}) \tag{4.14}$$

Using the standard Lagrange multipliers minimization method, the following iterative algorithm is preformed as shown in Algorithm 4:

I) Initialization:

1) Set the number of clusters C, where $2 \leq C \leq N$.

2) Set ϑ, where $1 \leq \vartheta < \infty$.

3) Select an initial partition $U(0)$=$[u^{(0)}_{i,k}]$.

II) Iteration: At step w, where $w = 0, 1, 2, ...$, do the following (lines 16 & 17):

1) For each cluster c_i, $1 \leq i \leq C$, and trajectory $traj_k$, $1 \leq k \leq N$, calculate $\rho^{(w)}_{i,k}$ using Eq. 4.15 (lines 18-20).

$$\rho^{(w)}_{i,k} = \frac{\sum_{g=1}^{N} u^{\vartheta(w)}_{i,g} s'_{k,g}}{\sum_{g=1}^{N} u^{\vartheta(w)}_{i,g}} \tag{4.15}$$

2) For each cluster c_i, $1 \leq i \leq C$, and trajectory $traj_k$, $1 \leq k \leq N$, calculate $\delta(tr_k, \rho_i)^{(w)}$ using Eq. 4.13 (line 21).

Algorithm 4 : PACD Prediction Training at the Data Center

Input:
Set of training trajectories G
Number of clusters C $2 \leq C \leq N$
The fuzzification coefficient ϑ $1 \leq \vartheta < \infty$
$U(0) = [u_{i,k}^{(0)}]$ initial degree of membership $\forall tr_k \in G$, $\forall i \in C$
Clustering termination threshold ϵ

PredModelTraining(G)
Begin
  for all $i \in G$ **do**
    for all $j \in G$ **do**
      if $i \neq j$ **then**
        Calculate $sim(tr_i, tr_j)$ Eq.4.9
      else
        Set $sim(tr_i, tr_j)$ to 1
  Set w=0
  while w ≥ 0 **do**
    for all $1 \leq i \leq C$ **do**
      for all $1 \leq k \leq N$ **do**
        Calculate $\rho_{i,k}^{(w)}$ Eq. 4.15
        Calculate $\delta(tr_k, \rho_i)^{(w)}$ Eq. 4.13
        Calculate $u_{i,k}^{(w+1)}$ to update to $U(w+1)$ Eq. 4.16
    Calculate $||U(w+1) - U(w)||$
    if $||U(w+1) - U(w)|| < \epsilon$ **then**
      Terminate clustering procedure by setting w= -1
    else
      Set $w = w + 1$
  for all $c \in C$ **do**
    Get the number of road segments in c, m
    for all $1 \leq i_0 \leq m$ **do**
      for all $1 \leq i_k \leq m$ **do**
        Calculate $P(i_0|i_k)$ via $\hat{P}(i_0|i_k)$ Eq. 4.26
    Estimate parameters $\eta_1, \eta_2, ..\eta_\ell$ that maximize LL in Eq. 4.27
    for all $1 \leq X_t = i_0 \leq m$ **do**
      for all $1 \leq X_{t-\ell} = i_\ell \leq m$ **do**
        ...
        for all $1 \leq X_{t-1} = i_1 \leq m$ **do**
          $\acute{Q}_c$=$[P(X_t$=$i_0|X_{t-\ell}$=i_ℓ,..X_{t-1}=$i_1)]$ Eq.4.25
  Send $S_{N \times N}$ to vehicles similarity matrix between N trajectories
  Send final $U(w)$ to vehicles Final membership degrees to clusters
  Send final $[\rho_{i,g}^{(w)}]$ to vehicles Final prototype values
  Send $\acute{Q}_c$ to vehicles $\forall c \in C$
End

3) Update the membership degrees $U(w)$ to $U(w+1)$ by calculating $u_{i,k}^{(w+1)}$ for each cluster c_i, $1 \leq i \leq C$, and trajectory $traj_k$, $1 \leq k \leq N$ using Eq. 4.16 (line 22).

$$u_{i,k}^{(w+1)} = \frac{1}{\sum_{j=1}^{C} \left(\frac{\delta(tr_k, \rho_i)^{(w)}}{\delta(tr_k, \rho_j)^{(w)}} \right)^{\frac{2}{\vartheta - 1}}} \tag{4.16}$$

4) Calculate $||U(w+1) - U(w)||$. If $||U(w+1) - U(w)|| < \epsilon$, terminate the clustering procedure. Otherwise, go to step 5 (lines 23-26).

5) Set $w = w + 1$ and repeat steps $1 - 4$ (line 27).

A trajectory is considered to be a member of any given cluster if its membership to that cluster is greater than or equal to a certain membership threshold U_{th}.

Stage 3: Train the MTD-Probit model in each cluster

In this stage, the data center trains the MTD-probit model in each cluster. The MTD-Probit model [116] is a variation of the standard MTD model [120]. Both models have been proposed to approximate the high-order Markov chain model using significantly fewer parameters than the fully parameterized model [120][116]. However, MTD-Probit has been shown to render higher precisions in estimating the transition probabilities of the high-order Markov chain model than the original MTD model [116]. Among the advantages of the MTD-Probit model is that it does not involve any constraints or super-imposed restrictions [116]. This facilitates capturing a broad range of associations among a set of variables that only nonparametric approaches can capture [116][122][123]. In order to explain the employed MTD-Probit model and the need for parsimonious high-order Markov chains, we first provide a brief description of the Markov chain model.

The Markov chain model is a probabilistic model that represents a sequence of

possible events [120][116]. It is simple and easy to implement [120], which makes it highly eligible to be used in a setting where there are multiple clusters that need to be separately trained. In discrete Markov chain models, an event is defined as a discrete-time random variable X_t which can take any value in the finite set $\{1, ..., m\}$. The main objective is to predict the value of X_t as a function of the values of the observations preceding that variable. In the first-order Markov chain model, the probability of X_t relies on the state yielded in the previous event only. Thus, as given in Eq. 4.17 [120], the current observation at time t is conditionally independent of those up to time $t-2$, and is only dependent on the immediate past that is represented by the previous observation at time $t-1$, where $q_{i_1 i_0}$ is the transition probability from state i_1 to state i_0.

$$\begin{aligned} P(X_t = i_0 | X_0 = i_t, ...X_{t-1} = i_1) &= P(X_t = i_0 | X_{t-1} = i_1) \\ &= q_{i_1 i_0} \end{aligned} \tag{4.17}$$

Considering all possible combinations of states, a transition matrix, denoted $Q_{m \times m}$, is created, where the sum of the transition probabilities in each row is equal to 1, and m is the number of states [120]. Thus, the total number of independent parameters to be estimated in a first-order Markov chain model is $m(m-1)$ [120]. One drawback of the first-order Markov chain model is that it is extremely limited, since it restricts its memory of past events to the immediately preceding event only, which could affect

the prediction accuracy [120].

$$Q_{m\times m} = \begin{array}{c} \\ X_{t-1} \\ 1 \\ 2 \\ 3 \\ ... \\ m \end{array} \begin{array}{c} \begin{array}{ccccc} & & X_t & & \end{array} \\ \left[\begin{array}{ccccc} 1 & 2 & 3 & ... & m \\ q_{11} & q_{12} & q_{13} & \cdots & q_{1m} \\ q_{21} & q_{22} & q_{23} & \cdots & q_{2m} \\ q_{31} & q_{32} & q_{33} & \cdots & q_{3m} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ q_{m1} & q_{m2} & q_{m3} & \cdots & q_{mm} \end{array}\right] \end{array}$$

In order to avoid the aforementioned restriction, high-order Markov chain models are designed to model situations where the current state does not depend on the first lag only but rather on the last ℓ observations [120]. Thus, as the case in our model, predicting the next road segment in an on-going trip would depend on the last ℓ road segments that have been traversed. This is referred to as an ℓ-order Markov chain model, where the transition probability from the sequence $i_\ell, ..i_1$ to i_0 is given by Eq. 4.18 [120].

$$\begin{aligned} & P(X_t = i_0 | X_0 = i_t, ...X_{t-1} = i_1) \\ & = P(X_t = i_0 | X_{t-\ell} = i_\ell, ...X_{t-1} = i_1) \\ & = q_{i_\ell..i_0} \end{aligned} \tag{4.18}$$

The high-order Markov chain model can provide higher prediction accuracy than the first-order Markov chain model [120]. However, the number of independent parameters that need to be estimated in the former is equal to $m^\ell(m-1)$ [120][116]. Hence, as the order ℓ of the chain and the number m of potential states increase, the number of independent parameters grows exponentially [120][116]. This can trigger

biased estimates in cases when certain sequences of states do not frequently appear in the data [120][116][127]. In other words, the number of independent parameters can get too large to be efficiently, or even identifiably estimated, which could affect the prediction accuracy [120][116][127]. The MTD model has been proposed in order to approximate high-order Markov chains using much less parameters compared to the fully parameterized model [120].

In the MTD model, each entry of the transition matrix constitutes the probability of observing an event at time t, given the previous events occurring from time $t-\ell$ to $t-1$. The effect of each lag on the current state is considered separately, and the transition probability is given by Eq. 4.19 [120], where λ_g is the weight parameter incorporated with lag g, and $q_{i_g i_0}$ is the transition probability from state i_g to state i_0. Note that the transition matrix Q used in Eq. 4.19 for the MTD model is not the same as the transition probability matrix generated in the first-order Markov chain model [120]. The only similarity is that they both have the same size $m \times m$ [120].

$$\begin{aligned} & P(X_t = i_0 | X_{t-\ell} = i_\ell, ...X_{t-1} = i_1) \\ & = \sum_{g=1}^{l} \lambda_g P(X_t = i_0 | X_{t-g} = i_g) \\ & = \sum_{g=1}^{l} \lambda_g q_{i_g i_0} \end{aligned} \tag{4.19}$$

In order to estimate the parameters $Q_{m\times m}$ and $\{\lambda\}$ of the MTD model, the log-likelihood of the model, given by Eq. 4.20, should be maximized [120][127][128]. Note that in Eq. 4.20, $n_{i_\ell ... i_0}$ is the number of times that the sequence $i_\ell ... i_0$ appears in the data, and the summation is carried out for each of the $\ell + 1$ variables $i_\ell, ...i_0$ varying from 1 to m. In order to make sure that the results yielded by the model are in

the form of probabilities, the vector of lag parameters, $\lambda = (\lambda_\ell, ...\lambda_1)'$, is subject to the constraints given by Eq. 4.21 and Eq. 4.22. In addition, as given by Eq. 4.23 and Eq. 4.24, respectively, Q should be a stochastic matrix where each transition probability q_{ij} is not less than zero and the transition probabilities in each row sum to one [120][127]. Thus, the maximization of the log-likelihood is done subject to the constraints given by Eq. 4.21, Eq. 4.22, 4.23, and 4.24 [120][127][128].

$$\max\ LL = \sum_{i_\ell ..., i_0 = 1}^{m} n_{i_\ell ... i_0} \log(\sum_{g=1}^{\ell} \lambda_g q_{i_g i_0}) \tag{4.20}$$

Subject to

$$\sum_{g=1}^{\ell} \lambda_g = 1 \tag{4.21}$$

$$\lambda_g \geq 0 \quad \forall g = 1, ...\ell \tag{4.22}$$

$$q_{i_k i_0} \geq 0 \quad \forall i_k \in 1, ...m, \forall i_0 \in 1, ...m \tag{4.23}$$

$$\sum_{i_0=1}^{m} q_{i_k i_0} = 1 \quad \forall i_k \in 1, ...m \tag{4.24}$$

As previously mentioned, the number of independent parameters that need to be estimated to create the transition probability matrix Q is $m(m-1)$ [120]. In addition to such parameters, an ℓ^{th}-order MTD model requires the estimation of $\ell - 1$ independent parameters (since they all sum to 1). Thus, in contrast to the high-order model that involves the estimation of $m^\ell(m-1)$ independent parameters, the number of independent parameters in the MTD model is reduced to $m(m-1)+(\ell-1)$, increasing only linearly with ℓ [120][127][128]. For example, if m=4, the number of parameters for a second-order chain (i.e., ℓ=2) in the MTD model is 13, as opposed

to the 48 parameters in the typical third-order Markov chain model. Accordingly, as m and ℓ increase, the MTD model can be much more parsimonious than the fully parameterized high-order Markov chain model [120][127][128]. However, the MTD parameters $Q_{m\times m}$ and $\{\lambda\}$ are difficult to estimate due to the nonlinearity of the objective function and the large number of constraints that the model has to adhere to [120][128][116]. Thus, no analytical solution is available to the log-likelihood maximization problem subject to the constraints that $Q_{m\times m}$ and $\{\lambda\}$ have to satisfy [128]. It could be profoundly challenging to reach a global maximum, specially if the initial values are too distant from the optimal values.

Recently, the MTD-Probit model has been proposed to parsimoniously approximate the high-order Markov chain model while avoiding the estimation problem associated with the MTD model [116]. To do so, the MTD-Probit model eliminates any form of constraints or restrictions, thus facilitating the estimation procedure [116]. It has been shown to significantly outperform the MTD model in terms of providing a more accurate representation of the transition probability $P(X_t = i_0|X_{t-\ell} = i_\ell, ...X_{t-1} = i_1)$ [116][122][123]. In the MTD-probit model, the transition probability is modeled as given by Eq. 4.25, where the parameters $\eta_j \in \Re$ are the weights of the nonlinear combination, which indicate that the larger the coefficient, the larger the significance of the corresponding variable $P(X_t = j)$, and Φ represents the (cumulative) standard normal distribution function [116]. Note that it is possible to replace Φ by a different distribution function of any continuous random variable that has the state space $\Re$ [116]. No constraints are required since $P^{\Phi}(X_t = i_0|X_{t-\ell} = i_\ell, ...X_{t-1} = i_1)$

in Eq. 4.25 is bounded within the range $[0, 1]$ regardless of the values of η_j [116].

$$\begin{aligned} &P(X_t = i_0 | X_{t-\ell} = i_\ell, ...X_{t-1} = i_1) = \\ &P^{\Phi}(X_t = i_0 | X_{t-\ell} = i_\ell, ...X_{t-1} = i_1) := \\ &\frac{\Phi(\eta_0 + \eta_1 P(i_0|i_1) + ...\eta_\ell P(i_0|i_\ell))}{\sum_{i_k=1}^{m} \Phi(\eta_0 + \eta_1 P(i_k|i_1) + ...\eta_\ell P(i_k|i_\ell))} = \\ &\frac{\Phi(\eta_0 + \sum_{g=1}^{\ell} \eta_g q_{i_g i_0})}{\sum_{i_k=1}^{m} \Phi(\eta_0 + \sum_{g=1}^{\ell} \eta_g q_{i_g i_k})} \end{aligned} \tag{4.25}$$

The argument of $\Phi(.)$ in the numerator of Eq. 4.25 adopts the same principle used in the MTD model, with an additional constant term η_0 [116]. This parameter can be eliminated but it has been shown that it can usually improve the fit (i.e., bring $P^{\Phi}(X_t = i_0 | X_{t-\ell} = i_\ell, ...X_{t-1} = i_1)$ closer to $P(X_t = i_0 | X_{t-\ell} = i_\ell, ...X_{t-1} = i_1)$ [116]. The denominator of Eq. 4.25 is used for normalization purposes [116].

The estimation process in the MTD-probit model is a two-step procedure that works as follows [116]: 1) The quantities $P(i_0|i_k)$=$q_{i_k i_0}$ are estimated nonparametrically via the use of the consistent estimators $\hat{P}(i_0|i_k)$=$q_{i_k i_0}$ given by Eq. 4.26 [116].

$$\hat{P}(i_0|i_k) = q_{i_k i_0} = \frac{n_{i_k i_0}}{\sum_{i_x=1}^{m} n_{i_k i_x}} \tag{4.26}$$

Note that this is the ratio of the number of transitions from state i_k to state i_0, denoted $n_{i_k i_0}$, to the total number of transitions from state i_k to every other state. 2) The parameters η_j are estimated via the maximum likelihood method by maximizing the log-likelihood LL given by Eq. 4.27. Due to the absence of any constraints, the estimation procedure becomes much easier, and standard numerical optimization routines can be employed [116]. Thus, we utilize the Constrained Maximum Likelihood

module in the GAUSS software for the estimation procedure [129].

$$LL = \sum_{i_\ell \ldots, i_0 = 1}^{m} n_{i_\ell \ldots i_0} \log P^{\Phi}(X_t = i_0 | X_{t-\ell} = i_\ell, \ldots X_{t-1} = i_1) \tag{4.27}$$

As depicted in Algorithm 4, for each cluster $c \in C$ obtained in the previous clustering stage (line 28), the following training procedure is applied: 1) get the number of road segments m in all the training trajectories available in c. (line 29), 2) calculate the parameters $P(i_0|i_k)$ via the estimators $\hat{P}(i_0|i_k)$ in Eq. 4.26, $\forall$ $1 \leq i_0 \leq m$ and $1 \leq i_k \leq m$ (lines 30-32), 3) estimate the parameters η_j that maximize LL in Eq. 4.27 using the Constrained Maximum Likelihood module in the GAUSS software (line 33), 4) create the high-order transition matrix $Q_c^{'}$ by calculating the transition probabilities using Eq. 4.25 (lines 34-38).

Upon subscribing to the service, the vehicles download the similarity matrix $S_{N \times N}$, the final membership values of the trajectories to the created clusters $U(w)$, the final prototype values $[\rho_{i,g}^{(w)}]$, and the $Q_c^{'}$ matrix, $\forall$ $c \in C$ from the data center (lines 39-42). As explained later in details, this information is used when a vehicle needs to predict the current location of a mobile caching vehicle.

4.4.3 PACD Information Exchange at Vehicles

As previously mentioned, parked and moving vehicles on the road exchange some information via beacon messages upon encounter. Typically, beacon messages include certain information observed at the time of encounter, including the vehicle's position, speed, and heading [5]. Intuitively, the current position of a vehicle v_k provides information about the current road segment r_{cur}^k that the vehicle is located at. Along with the information typically enclosed in beacon messages, each vehicle adds its own

cached content information, as well as the cached content information maintained in its LC. In addition, for each moving vehicle v_k, beacon messages also include the set of IDs of the clusters that its trajectory has high membership with. This set, denoted $\hat{C}_k$, where $\hat{C}_k \subset C$, is the set of clusters whose trajectories exhibit high similarity with the vehicle's trajectory $traj_k$.

In order for v_k to determine $\hat{C}_j$, it calculates the membership $u_{i,k}$ of its own trajectory $traj_k$ to each cluster $c_i \in C$ using Eq. 4.16. The clusters with which $traj_k$ yields a membership greater than or equal to a certain membership threshold U_{th}, are included in $\hat{C}_k$. Furthermore, v_k includes in the beacon message its expected time of trip termination $t_{k,last,e}$, which is the expected time of departure from the last road segment in its ongoing trip. Note that this time is calculated using Eq. 4.1. This time helps informing other vehicles when a vehicle whose information they maintain in the TPP has reached its destination, and thus access to its cached contents might no longer be feasible.

Thus, the information included in the beacon message of each moving vehicle v_k consists of the following: 1) current position (and thus r^k_{cur}), 2) speed, 3) heading, 4) next road segment along its trajectory r^k_{next}, 5) sequence of the $\ell - 2$ previous road segments that were traversed before r^k_{cur} in the vehicle's ongoing trip, 6) $\hat{C}_k$, 7) $t_{k,last,e}$, 8) its own cached content information (i.e., names of contents that it holds in its cache), and 9) cached content information maintained by v_k in its LC, which includes the cached content information pertaining to the clusters of parked vehicles that it has encountered so far. Note that each CH of a parking cluster also includes the typical information in its own beacon messages (i.e., position, speed, and heading), along with its maintained LC, which includes the cached content information reflecting the

names of the data cached in its own cluster, as well as in other clusters. The latter are the ones the CH acquire information about through the exchanged beacon messages with neighboring moving vehicles.

The problem with exchanging the aforementioned cached content information is the relatively large number of bytes that the name of each content can take [13][130]. As the number of contents increases, cached content information can lead to a significant increase in the size of beacon messages, which can in turn lead to an increased overhead. In order to reduce the size of beacon messages, and consequently the amount of overhead triggered, we send the cached content information pertaining to each vehicle in the form of a b-bit bloom filter. Similarly, each entry maintained in the LC of each vehicle is expressed as a b-bit bloom filter to reflect the cached content information in the corresponding parking cluster.

A bloom filter is a probabilistic data structure that facilitates set membership queries, and is highly efficient in terms of space [14][131]. It is designed to quickly and space-efficiently provide information as to whether or not an element is present in a set. This efficiency comes at the expense of providing such information in the form of a probability [14][131]. In particular, a bloom filter either states that an element is definitely not in the set, leaving no room for a probability of false negative, or that it is probably in the set, leading to a probability of false positive [14][131]. Such a probability is not an issue as long as it is kept sufficiently low, which is highly feasible [14][131], as explained later.

A bloom filter is composed of an array of b bits that represents a set $D = \{d_1, d_2, ...d_z\}$ of z elements [131]. Note that in our case, the set of elements represents the set of possible content names (i.e., data items). Initially, all bits of the

array are set to zero [131]. The main idea is to use y independent hash functions h_1, h_2,...h_y to map each item $d \in D$ to one of the positions of the array, with a uniform random distribution over the range $\{0, ..b-1\}$ [131]. We use MurmurHash3 since it is one of the most prominent hash functions that has a low collision rate and high performance [131].

In order to include the cached content information in beacon messages when they are sent from vehicles, the name of each cached content is hashed using all y hash functions, and the corresponding positions in the b-bit bloom filter are set to 1. When a vehicle v_j receives the beacon message, it maintains the bloom filter reflecting the cached content information of the corresponding node in its TPP. If v_j receives an interest packet requesting a certain content later on, it checks whether the requested content x is in the set of cached elements represented by the bloom filter belonging to any of the data holders in its TPP. In order to do that, it checks whether all bits at the positions indicated by $h_n(x)$, $\forall\ 1 \leq n \leq y$, are set to 1. If so, it is assumed that x is present in the set of cached contents. Otherwise, if at least one of the bits of $h_n(x)$ is set to zero, then x is definitely not in the set.

An illustrative scenario of bloom filters is depicted in Figure 4.5, where all bits of the b-bit bloom filter are initially set to zero. In this scenario, we assume that $b = 16$, and the number of hash functions is $y = 3$. In order to send a bloom filter indicating the presence of d_4 and d_5 in its cache, the beacon message sender v_s determines the value of each of the three hash functions of d_4 and d_5, and sets the bits at the corresponding positions to 1. When the beacon message receiver v_r receives a request packet for d_7 later on, it calculates the three hash functions of d_7, and checks if all the bits at the corresponding positions are set to 1. If so, then d_7 is probably a member.

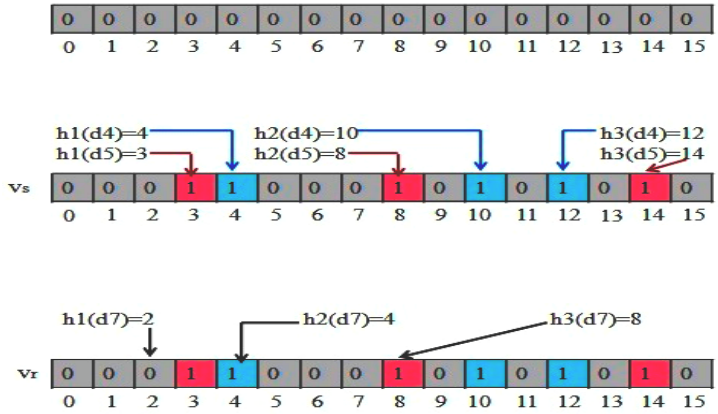


Figure 4.5: An illustrative scenario of bloom filters.

Otherwise, as the case here, d_7 is definitely not a member.

There is a probability of false positive in bloom filters, which is triggered by indicating that an element x is present in the set of cached contents when it is actually not. Such a probability is denoted f and is given by Eq. 4.28 [14][131]. Note that the number of hash functions y can have a profound effect on the value of f. Accordingly, the optimal number of y that minimizes f, denoted y_{opt}, has been estimated by taking the derivative and has been shown to be equivalent to Eq. 4.29 [14][131].

$$f = (1 - e^{-yz/b})^y \tag{4.28}$$

$$y_{opt} = \frac{b}{z} ln2 \tag{4.29}$$

In order to further reduce the overhead, we assume that vehicles send the original beacon message that does not involve any additional information as long as the time for sending the one with the extra information has not been triggered. This time is triggered periodically but over a longer interval than that of the original beacon message. For example, if the latter is sent every t seconds, the larger beacon message

is sent every xt seconds, where $x > 1$. We can also restrict the number of parking clusters that the LC maintained by each node can include information about.

4.4.4 PACD Trajectory Prediction at Moving & Parked Vehicles

This procedure is triggered when one of the following two cases occurs: The first case is when a vehicle v_f (a moving vehicle or a parked CH) receives a beacon message from a moving vehicle v_i for the first time (i.e., v_f's TPP does not already have an entry related to v_i). The second case is when an entry about v_i already exists but the information enclosed in the new beacon message pertaining to the ℓ road segments of its trajectory differs from what v_i has previously predicted. For example, assume that at time t, v_f receives a beacon message from v_i for the first time, where it shares the following $\ell = 3$ road segments of its ongoing trip: r_1, r_3, r_6, corresponding to $r_{prev}, r_{cur}, r_{next}$. This matches the first aforementioned case, so the prediction procedure is triggered, and v_f predicts the remaining trajectory of v_i to be: r_4, r_5, r_7, r_9, r_2. Then, at time $t + \Delta$, v_f and v_i encounter each other once again, and v_f receives a new beacon message from v_i. By then, v_i has already traversed r_1, r_3, and r_6. Thus, in this beacon message, v_i shares some updated information about the ℓ road segments of its ongoing trip, indicating them as follows: r_6, r_4, r_8. Since this partial trajectory does not match the previously predicted trajectory by v_f, the prediction procedure is triggered and v_f repeats the prediction process given the new received information.

In order to predict the remaining trajectory of any moving vehicle v_i, v_f explores $\hat{C}_i$, which is the set of clusters of similar trajectories with which v_i has high membership. Note that $\hat{C}_i$ is indicated in the received beacon message from v_i. As previously

mentioned, each cluster $c \in C$ is associated with an MTD-Probit high-order matrix Q'_c. Thus, for each cluster $c_k \in \hat{C}_i$, the corresponding matrix Q'_{c_k} is used to predict the following road segment that v_i is expected to traverse after r^i_{next}. Another piece of information that is used in the prediction process is the sequence of ℓ road segments $\{j_1,..j_\ell\}$ constituting part of v_i's ongoing trip. Since we have no prior knowledge as to how many steps h should be predicted (i.e., the number of the following road segments that v_i is estimated to visit after r^i_{next}), we start with h=1 first and then iteratively increment h if the expected departure time of v_i from r^i_{next+h} is less than $t_{i,last,e}$. Note that $t_{i,last,e}$ is included in the received beacon message to indicate the expected time at which the trip of v_i is expected to terminate. We define the set of ℓ road segments used for high-order prediction as L_h=$j_{\ell-(h-1)},..j_{1-(h-1)}$. Note that, when $h = 1$, the sequence L_1 is equal to $j_\ell,..j_1$, which is the sequence that was provided by v_i upon encounter. We also define $W^{c_k}_{j_{\ell-(h-1)},..j_{1-(h-1)}}$=$\{w^{c_k}_{1,L_h}, ..w^{c_k}_{m,L_h}\}$ as the row in the matrix Q'_{c_k} that represents all the transition probabilities from the sequence L_h to each of the m road segments that are included in the trips comprising the cluster. Note that for each possible state, representing all possible sequences of ℓ road segments in Q'_{c_k}, v_f keeps track of the state yielding the maximum transition probability in the corresponding vector $W^{c_k}_{j_{\ell-(h-1)},..j_{1-(h-1)}}$. At each step h, r^i_{next+h} is predicted within each cluster $c_k \in \hat{C}_i$, and the road segment yielding the least prediction uncertainty (i.e., the one with minimum entropy) is selected.

It is worth mentioning that v_f associates each road segment constituting the relevant part of v_i's trajectory with the expected time of departure from that road segment, as well as the entropy of prediction associated with it. This includes r^i_{cur}, r^i_{next}, and the set of predicted road segments following r^i_{next}. Since r^i_{cur} and r^i_{next} are

Algorithm 5 : PACD Trajectory Prediction at Moving & Parked Vehicles

1: **Input:**
2: Set of ℓ road segments of v_i, $L_1^i = j_\ell,..j_1$
3: Set of clusters v_i's trajectory has high membership with, $\hat{C}_i$
4: Time of v_i's trip termination, $t_{i,last,e}$
5:
6: *Trajectory_Prediction($L(1)$, $\hat{C}_i$, $t_{i,last,e}$)*
7: **Begin**
8: Set $minEntropy = \infty$
9: Set $h = 1$
10: **while** $h \geq 1$ **do**
11: **for all** $c_k \in \hat{C}_i$ **do**
12: Predict the following road segment r_{next+h}^{i,c_k} // Eq. 4.30
13: Calculate individual entropy $E_{i,h}^{c_k}$ // Eq. 4.31
14: **if** $E_{i,h}^{c_k} \leq minEntropy$ **then**
15: Set $c_{\acute{k}}$=c_k
16: Set the final predicted r_{next+h}^{i} to $r_{next+h}^{i,c_{\acute{k}}}$
17: Set the individual entropy $E_{i,h}$ to $E_{i,h}^{c_{\acute{k}}}$
18: Calculate the sequence entropy $E_i^{seq,h}$ // Eq. 4.32
19: Calculate the departure time $t_{i,next+h,e}$ // Eq. 4.1
20: Add r_{next+h}^{i}, $E_i^{seq,h}$, and $t_{i,next+h,e}$ to the set $\hat{R}_i$
21: **if** $t_{i,next+h,e} < t_{i,last,e}$ **then**
22: Set $h = h + 1$
23: **else**
24: Set $h = 0$ // Terminate the procedure
25: **End**

known and not predicted, the entropy associated with both of them is set to 0. The time of departure from r_{cur}^i and r_{next}^i, denoted $t_{i,cur,e}$ and $t_{i,next,e}$, respectively, are calculated using Eq. 4.1. The road segments, along with their corresponding entropy, and time of departure are maintained by v_f in a set denoted $\hat{R}_i$. If $t_{i,next,e} < t_{i,last,e}$ (i.e., r_{next}^i is not the last road segment along v_i's trajectory), v_f predicts the remaining trajectory of v_i by performing the following steps depicted in Algorithm 5:

1) Initially, set h to 1 (line 9). Do the following as long as $h = 1$ or $t_{i,next+h,e} < t_{i,last,e}$ (line 10)

2) For each $c_k \in \hat{C}_i$, do the following (line 11):

I) Predict the road segment r_{next+h}^{i,c_k} that v_i is most likely to visit in h steps following r_{next}^{i}. This road segment is estimated to be the one that yields the maximum transition probability in $W_{j_{\ell-(h-1)},..j_{1-(h-1)}}^{c_k}$, as given by Eq. 4.30 (line 12).

$$\begin{aligned} r_{next+h}^{i,c_k} &= \arg \max_{1\leq j_0 \leq m} P(X_{t+(h-1)} = j_{0-(h-1)} | X_{t-(\ell-(h-1))} \\ &\qquad = j_{\ell-(h-1)}, ..X_{t-(1-(h-1))} = j_{1-(h-1)}) \\ &= \arg \max_{1\leq j_{0-(h-1)} \leq m} q^{c_k}_{j_{\ell-(h-1)}\cdots j_{1-(h-1)} j_{0-(h-1)}} \\ &= \arg \max_{1\leq x \leq m} w_x^{c_k, L_h} \end{aligned} \tag{4.30}$$

II) For each cluster, v_f calculates the individual entropy of the predicted road segment r_{next+h}^{i,c_k} yielded in the previous step (line 13). This entropy is denoted $E_{i,h}^{c_k}$ and is given by Eq. 4.31 [132]. Note that the uncertainty increases when the states are equally probable (i.e., resembling a uniform distribution) [132]. This occurs when the transition probabilities from the current state to any of the other possible states are too close to each other [132]. For example, if a vehicle is at state $j_\ell..j_1$ and can transition to either state $j_0 = 2$ with transition probability 0.5 or state $j_0 = 3$ with transition probability 0.5, the entropy would be extremely high. This is as opposed to a case where the probabilities are 0.8 and 0.2, respectively (i.e., the two states are not equally likely).

$$E_{i,h}^{c_k} = \sum_{j_{0-(h-1)=1}}^{m} q^{c_k}_{j_{\ell-(h-1)}..j_{0-(h-1)}} \log_2 q^{c_k}_{j_{\ell-(h-1)}..j_{0-(h-1)}} \tag{4.31}$$

3) Find the cluster $c_{\acute{k}} \in \hat{C}_i$ that renders the minimum entropy $E_{i,h}^{c_k}$, and set the final predicted road segment at step h, denoted r_{next+h}^{i}, to $r_{next+h}^{i,c_{\acute{k}}}$. Set the individual

entropy of r^i_{next+h}, denoted $E_{i,h}$, to $E^{c_k}_{i,h}$ (lines 14-17).

4) v_f uses the set of predicted road segments starting from r^i_{next+1} to r^i_{next+h}, denoted Υ^i, where $|\Upsilon^i| = a$, to calculate the prediction entropy of the entire sequence until r^i_{next+h}. We make a distinction between the sequence entropy $E^{seq,h}_i$ and the individual entropy $E_{i,h}$. The latter reflects the degree of uncertainty associated with a single estimated road segment r^i_{next+h} in step h. In order to measure the sequence entropy $E^{seq,h}_i$, the sum of the individual entropy $E_{i,h}$ of the current and preceding values of h is calculated as given by Eq. 4.32 (line 18).

$$E^{seq,h}_i = \sum_{h=1}^{a} E_{i,h} \tag{4.32}$$

5) Calculate the departure time from r^i_{next+h}, denoted $t_{i,next+h,e}$, using Eq. 4.1. Add r^i_{next+h}, along with its corresponding sequence entropy $E^{seq,h}_i$, and $t_{i,next+h,e}$ to the set $\hat{R}_i$. If $t_{i,next+h,e} < t_{i,last,e}$, set h to $h+1$ and go to step 2. Otherwise, terminate the procedure (lines 19-24).

The vehicle v_f keeps track of the current road segment that v_i is traversing at the current time t_{cur}, denoted r^i_{curNew}. Initially r^i_{curNew} is set to r^i_{cur}, $t_{i,curNew,e}$ is set to $t_{i,cur,e}$, and $E^{seq,curNew}_i$ is set to 0. Note that as time goes on, this position differs from r^i_{cur} that was recorded at the time of encounter. Once t_{cur} exceeds $t_{i,curNew,e}$, the next road segment in the set $\hat{R}_i$ is set as r^i_{curNew}, and its corresponding time of departure and entropy are set as $t_{i,curNew,e}$ and $E^{seq,curNew}_i$, respectively. This goes on continuously so that the information regarding r^i_{curNew} can be maintained by v_f in its TPP. As explained later, such information can then be accessed by v_f during the content discovery process. Note that the new current position of v_i can be determined based on the start position of r^i_{curNew} and the total distance it has traversed on r^i_{curNew}

starting from the time of arrival at r^i_{curNew} (given by Eq. 4.2) till t_{cur}. This total distance is denoted $dist_{i,t_{i,curNew},s}$, and is given by Eq. 4.5.

4.4.5 PACD Content Discovery at Moving and Parked Vehicles

This algorithm is either performed by a requesting vehicle v_{req} or a vehicle that is forwarding an interest packet v_f. The purpose is to dynamically find a closer data holder to the requester than the current data provider. Initially, the current data provider v_{cp} to which the vehicle navigates the request packet is set as the data center (i.e., the original data provider). Four pieces of information are appended to the interest packet. The first is the last time of encounter of the requesting vehicle $\hat{t}_i$, which specifies the last time at which the requesting vehicle was spotted in the indicated position in the packet. The second is the entropy $E^{seq,h}_{req}$, which reflects the degree of uncertainty in the estimated position of the requesting vehicle that is included in the packet. Note that when the requesting vehicle issues the request packet, such an entropy is set to 0. This is since the position indicated is known to be 100% correct. The last time of encounter of the current data provider $\hat{t}_{cp}$, and the entropy of its estimated position $E^{seq,h}_{cp}$ are also included in the interest packet. If the current data provider v_{cp} is a static node, such as the data center or any parked vehicle, the corresponding entropy is set to 0. As portrayed in Algorithm 6, the cache discovery procedure is performed as follows:

(a) Once a vehicle v_f receives an unexpired interest packet, it determines the most accurate estimation of the current position of the requester v_{req} (lines 8-18). To do so, v_f checks its own TPP to determine if v_{req} is among the vehicles it maintains information about (i.e., v_{req} has been previously encountered). If so, and if the entropy

Algorithm 6 : PACD Cache Discovery at Moving and Parked Vehicles

Input:
Forwarding Vehicle v_f, Interest Packet I
Reply Packet rep, Requested Data d
Requesting Vehicle v_{req}, Current Data Provider v_{cp} //source of I, destination of I
Neighborhood list NB
content_discovery(I)
Begin
  if I is not expired **then**
    if v_{req} is recorded in v_f's TPP **then**
      if $E_{req}^{seq(r_{curNew}^{TPP})} < E_{req}^{seq(r_{curNew}^{I})}$ **then** //E is the entropy
        $\text{NewPos}_{req} = r_{curNew}^{TPP}$
      else if $E_{req}^{seq(r_{curNew}^{TPP})} = E_{req}^{seq(r_{curNew}^{I})}$ **then**
        if $t_{req}^{TPP} \geq t_{req}^{I}$ **then** //t is the last time of encounter
          $\text{NewPos}_{req} = r_{curNew}^{TPP}$
        else if $t_{req}^{TPP} < t_{req}^{I}$ **then**
          $\text{NewPos}_{req} = r_{curNew}^{I}$
      else
        $\text{NewPos}_{req} = r_{curNew}^{I}$
      Update v_{req}'s position, t_{req}, and $E_{req}^{seq(r_{curNew}^{I})}$ in I
    if there is D matching I in the cache **then**
      generate a reply rep
      forward rep
    else if any node in NB has d in its cache **then**
      update v_{cp} in I //The neighbor with the cached data
      forward I
    else
      if v_{cp} is a moving vehicle **then**
        if v_{cp} is recorded in v_f's TPP **then**
          if $E_{cp}^{seq(r_{curNew}^{TPP})} < E_{cp}^{seq(r_{curNew}^{I})}$ **then**
            $\text{NewPos}_{cp} = r_{curNew}^{TPP}$
          else if $E_{cp}^{seq(r_{curNew}^{TPP})} = E_{cp}^{seq(r_{curNew}^{I})}$ **then**
            if $t_{cp}^{TPP} \geq t_{cp}^{I}$ **then**
              $\text{NewPos}_{cp} = r_{curNew}^{TPP}$
            else if $t_{cp}^{TPP} < t_{cp}^{I}$ **then**
              $\text{NewPos}_{cp} = r_{curNew}^{I}$
          else
            $\text{NewPos}_{cp} = r_{curNew}^{I}$
          Update v_{cp}'s position, t_{cp}, and $E_{cp}^{seq(r_{curNew}^{I})}$ in I
    if v_f is in the range of v_{cp}'s Position **then**
      if v_{cp} is not in NB **then**
        v_{cp}=data center
    if d matches an entry in v_f's TPP **then**
      determine Ω //Set of potential providers of d in the TPP
      calculate $rank_k$ using Eq. 4.33 // $\forall\ v_k \in \hat{\Omega}$
      calculate $rank_{max} = \max_{k \in \hat{\Omega}} rank_k$
      set $v_{cp} = arg\max_{k \in \hat{\Omega}} rank_k$
    update v_{cp}, its position, t_{cp}, and $E_{cp}^{seq(r_{curNew}^{I})}$ in I
    forward I
End

associated with the most recent location r^{req}_{curNew} indicated in the TPP is less than that included in the interest packet, the current position of the requester is acquired from the TPP. If the entropy indicated in the TPP is greater than that in the interest packet, v_{req}'s current position is set to that indicated in the interest packet. If the entropy associated with the position indicated in the TPP is equal to that in the interest packet, the one associated with the most recent time of encounter is selected.

(b) Based on the outcome of step (a), the requester's current position, its corresponding prediction entropy, and the last time of encounter are updated in the interest packet (line 19). Note that later on, when the requested content is found, such information is copied in the data packet to enable the requester tracking process to be continued by all encountered vehicles along the data forwarding path as well.

(c) The vehicle v_f checks its own local cache to determine whether there is a match of the requested content. If so, v_f sends a reply packet back to the requester (lines 20-22).

(d) If not (i.e., local cache miss), v_f checks the set of its neighbors NB, to determine if any of them has the data in its cache. If so, v_f forwards the interest packet to it (lines 23-25). Otherwise, the following steps are preformed.

(e) If the current data provider v_{cp} (i.e., the destination of the interest packet) is a moving vehicle, v_f determines the most accurate estimation of v_{cp}'s current road segment r^{cp}_{curNew}, as well as its corresponding entropy. To do this, it applies the same logic performed for the requester in steps (a) and (b) (lines 26-38). Otherwise, if v_{cp} is a static node (i.e., a parked vehicle or the data center), its position and corresponding entropy remain the same. Note that the entropy of any static node is set to 0, since its position is fixed, and thus no uncertainty is involved.

(f) If the estimated position of v_{cp} is within the communication range of v_f but the former cannot be found, then the current data provider is changed to the data center (lines 39-41). Otherwise, v_{cp} remains the same.

(g) The vehicle v_f checks its TPP to determine the set of vehicles that have the requested content $d \in D$ in their cache (lines 42 & 43). To do so, v_f checks the bloom filter associated with each vehicle in its TPP to determine if all the bits at the positions resulting from the hash functions $h_x(d)\ \forall\ 1 \leq x \leq y$, are set to 1. If so, then the requested content is said to be present among the cached contents of the corresponding vehicle. Accordingly, v_f adds the latter to the set of possible providers of the requested content, denoted Ω.

(h) Once all possible data providers have been determined, v_f ranks each vehicle $v_k \in \Omega$ to evaluate the profit of navigating the request packet to it instead of the current data provider v_{cp} (line 44). Such a rank, denoted $rank_k$, is calculated based on two factors: 1) The distance (in hops) between the data holder v_k and the requester, $\hat{d}_{kReq}$. The closer they are to each other, the better. Thus, the lower the distance, the higher the rank. Note that the number of hops between v_k and the requester is calculated by dividing the distance d_{kReq} by the communication range. Intuitively, the distance is calculated using the requester's updated position obtained in step (a). In order to determine the distance d_{kReq}, v_f uses r^k_{curNew} (i.e., the road segment at which v_k is currently traversing) and the corresponding position, maintained in v_f's TPP. As previously explained in the aforementioned trajectory prediction procedure, such a position reflects the most recent estimation of v_k's current position. If $v_k \in \Omega$ is a parked vehicle, the position maintained in the TPP is a well known position rather than an estimated position, since its location is fixed. 2) The second factor used to

rank each potential data provider $v_k \in \Omega$ is the prediction entropy (i.e., uncertainty) of its estimated position $E_k^{seq,h}$. The lower the entropy, the higher the rank. The entropy associated with r_{curNew}^k of each vehicle $v_k \in \Omega$ has already been obtained during the trajectory prediction procedure. Note that if v_k is a parked vehicle, the corresponding entropy is set to 0. Based on the two aforementioned factors, v_f ranks each $v_k \in \Omega$ by multiplying the normalized values of $\hat{d}_{kReq}$ and $E_k^{seq,h}$, as given by Eq. 4.33. For simplicity, we refer to $E_k^{seq,h}$ as E_k. The vehicle v_f also ranks the data center and the current data provide v_{cp} using Eq. 4.33 in order to consider all possible data providers. The rank of v_{cp} is calculated based on its position and associated entropy obtained in step (e).

$$rank_k = \frac{(\max_{v_j \in \Omega} E_j) - (E_k)}{(\max_{v_j \in \Omega} E_j) - (\min_{v_j \in \Omega} E_j)} \times \frac{(\max_{v_j \in \Omega} \hat{d}_{jReq}) - (\hat{d}_{kReq})}{(\max_{v_j \in \Omega} \hat{d}_{jReq}) - (\min_{v_j \in \Omega} \hat{d}_{jReq})} \tag{4.33}$$

(i) The vehicle v_f determines the node yielding the maximum rank among $v_k \in \Omega$, the current data provider v_{cp}, and the data center. This node is set as the current data provider, and the destination of the interest packet is updated accordingly. Thus, the corresponding new position of the current data provider, its entropy, and last time of encounter are updated in the interest packet (lines 45-47).

(j) The vehicle v_f directs the interest packet towards the estimated position of the current data provider (line 48). This is done using the same forwarding procedure previously illustrated in CCD.

If v_f is the requesting vehicle, it starts the content discovery process by associating the interest packet with its current position, setting its associated entropy to 0, and setting the last time of encounter to the current time. It then performs the

aforementioned steps (c) and (d). In case of a cache miss, it executes steps (g)-(j) after setting the data center as the current data provider.

Once the requested data is found, a data packet is sent from the data holder back to the requester. Each vehicle along the data forwarding path makes a cache placement decision to determine whether or not to cache the content. We discuss the proposed cooperative cache placement scheme in the next chapter, along with the performance evaluation of both CCD and PACD when implemented with the cache placement procedure.

4.5 Summary

In this chapter, we proposed two tracking-based cooperative cache discovery schemes within VANETs, namely CCD and PACD, that can expand the search space beyond the neighborhood scope. Such an expansion aims at increasing the possibility of locating replicas that are close to the requester during the request-forwarding process, and thus increasing cache hits and improving the quality of service. Both CCD and PACD rely on beacon messages, as well as the mobility and stability of moving and parked vehicles, respectively, to diffuse cached content information into the network.

CCD exploits the static nature of parked vehicles to keep the cached content information alive at road segments for subsequent use, as well as to provide a rather stable tracking service. This helps expand the search space without having to send extra messages and incurring additional overhead. The tracking service in CCD exploits the trail of breadcrumbs that moving vehicles leave behind as they pass by the stationary parked vehicles. This trail is used to dynamically find closer replicas to the requester. CCD dynamically ranks potential data providers based on their

proximity to the requester, as well as the age of the maintained information.

PACD strives to expand the search space even further by dynamically predicting the location of mobile caching nodes. For this purpose, ARCA is employed to cluster historical trips of various vehicles based on their route similarity using the XXDice similarity coefficient. Each cluster is then trained using the MTD-Probit model to predict the remaining trajectory of vehicles. Using these predictions, PACD tracks all possible data providers and ranks them based on their proximity to the requester, as well as their prediction entropy. The latter reflects the level of uncertainty in the predicted location. In addition, PACD enables the exchange of cached content information that belong to parking clusters residing beyond the communication range of vehicles. This is done while reducing the amount of overhead incurred via the use of bloom filters. The performance evaluation of both CCD and PACD is presented in the next chapter, since there is a need to implement them along with the cache placement scheme, and the entire solution (i.e., cache discovery and cache placement) needs to eventually be compared to existing caching schemes in VANETs.

Chapter 5

Probabilistic Cooperative Caching in VANETs

5.1 Introduction

Cooperative cache placement has been shown to be a beneficial technique for ameliorating the performance of data access in multiple network paradigms, such as MANETs [12] and ICNs [13]. In cooperative caching, the nodes tend to trade information pertaining to the data they carry in their cache. Cooperative caching has been shown to help move the data within a closer proximity to the requester, augment data diversity, and achieve efficient utilization of the nodes' cache resources [12, 13]. Consequently, this leads to increased cache hits [12, 13]. This is paramount when the amount of contents is so profuse that they cannot always be fully accommodated, as the case in social media. However, despite its demonstrated leverage in many network paradigms, cooperative caching within VANETs has been mostly overlooked. This is due to the highly dynamic nature of vehicles, which can lead to unstable caching decisions. For example, caching decisions made by neighbors based on their information exchange can get rapidly revoked as vehicles move out of range.

In this thesis, we propose the Probabilistic Cooperative Caching at Moving and

Parked Vehicles (PCCMPV) scheme. In order to take advantage of the benefits of cooperative caching within VANETs, we counteract the aforementioned problems by exploiting the static nature of parked vehicles. We do so to provide a more stable residence for both the cached replicas and the received cached content information about other vehicles. This enables the extension of the cooperation range between nodes, and thus making more informed caching decisions. Such an extension is facilitated by sending different nodes' cached content information from parked to moving vehicles, and vice versa, via beacon messages.

As opposed to most existing caching schemes in VANETs that either cache the data at static nodes only, such as RSUs, or at moving vehicles only, PCCMPV caches the data at both parked and moving vehicles. In addition, PCCMPV is the first cooperative caching scheme in VANETs that extends the cooperation range beyond the neighborhood scope. In PCCMPV, we populate valuable road segments with diverse cached data to increase cache hits. This is to allow data to be acquired from nearby caching nodes rather than the far-away data center. We do so by dynamically assigning a probability of caching to vehicles along the data delivery path. Such a probability evaluates the importance of vehicles as caching nodes. For parked vehicles, this probability is calculated based on the traffic density of the corresponding road segment, as well as its closeness centrality, and remoteness from the closest data holder. Most existing cooperative caching techniques tend to adopt on-path caching techniques only [12, 13]. In PCCMPV, we exploit the trajectory of moving vehicles to also apply an implicit form of off-path caching.

We evaluate the performance of PCCMPV using the NS-3 simulator [64]. We

compare it to the Caching-Assisted Data Delivery (CADD) scheme [35] and the Distributed Probabilistic Caching (DPC) scheme [9]. This is since CADD is a caching scheme in VANETs that, despite not involving any explicit form of cooperation between the nodes, implicitly inherits some features of cooperative caching, while DPC is a non-cooperative caching scheme that has been shown to outperform many other caching schemes in VANETs, including the baseline (cache all) reactive caching scheme [9]. However, since CADD performs caching only at static nodes deployed at intersections, while DPC performs caching at moving vehicles only, we implement a combination of CADD and DPC. This is in order to ensure a fair comparison with PCCMPV that implements caching at both moving and parked vehicles. We refer to the combination of CADD and DPC as CADPC. In addition, in order to have a representative of explicit cooperative caching, as well as to ignore the effect of the cache discovery component during comparison, we implement both CADPC and PCCMPV while adopting the same explicit cooperative cache discovery methodology used in the GroupCaching (GC) scheme that is commonly used in dynamic networks [59]. Note that GC has been discussed in Section 2.3.4 in Chapter 2. We refer to the two schemes as CADPC-GC and PCCMPV-GC. Simulation results show that PCCMPV-GC outperforms CADPC and CADPC-GC in terms of access delay, packet delivery ratio, and cache-hit ratio.

In order to evaluate the performance of our proposed tracking-based cache discovery schemes CCD and PACD, which were presented in Chapter 4, we implement them both via the NS-3 simulator. We compare CCD and PACD to each other, as well as to the tracking-based cache discovery approach adopted in GC. This is since GC is the baseline neighborhood-restricted cache discovery approach that is typically used

in dynamic networks [12]. In addition, since CCD does not involve the exchange of LCs that PACD adopts, we also implement CCD with LCs (CCDLC), and compare it to GC, CCD, and PACD. Note that all cache discovery schemes are implemented while using the same cache placement scheme (i.e., PCCMPV). This is in order to evaluate their performance in an independent way, without the influential effect of the cache placement procedure.

We evaluate the proposed prediction technique in PACD by implementing the latter using the Regional Markov Model (RMM) scheme [106]. This is since RMM is a cluster-based prediction scheme that has been shown to outperform a number of trajectory prediction techniques in the literature. We refer to it as RPACD. Simulation results show that PACD outperforms RPACD in terms of prediction accuracy. They also show that: 1) CCD, CCDLC, and PACD outperform GC, 2) CCDLC outperforms CCD, and 3) PACD achieves the best results among all schemes. This is in terms of access delay, packet delivery ratio, and cache-hit ratio. Furthermore, it has been shown that the use of bloom filters in PACD enables it to significantly reduce beacon overhead compared to CCDLC.

Finally, we compare our proposed schemes to the aforementioned CADPC, and CADPC-GC as well. This is in order to evaluate our entire solution, which integrates both the proposed cache discovery and cache placement schemes, compared to existing caching schemes in the literature.

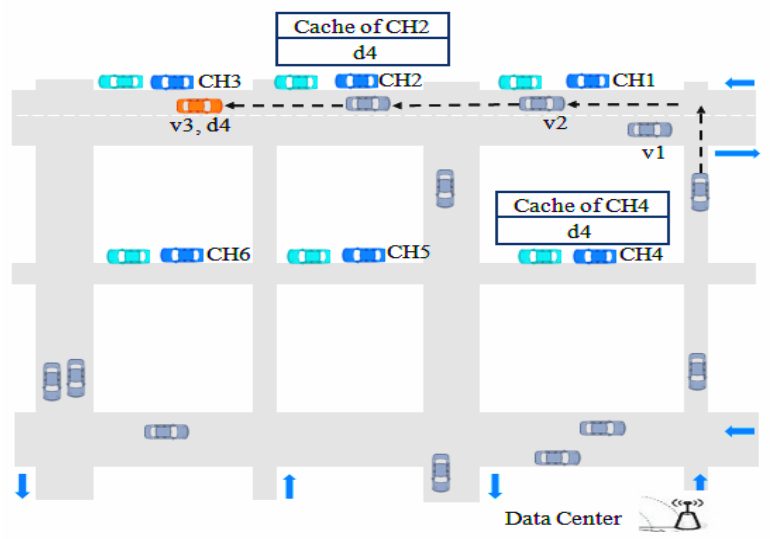
An overview of the related work in reactive and cooperative caching in VANETs, as well as other network paradigms, has already been provided in Sections 2.2, and 2.4 in Chapter 2. The remainder of this chapter is organized as follows. In Section 5.2, we provide a detailed description of the proposed scheme (PCCMPV). In Section 5.3,

we discuss the performance evaluation, as well as the simulation results of PCCMPV, CCD, CCDLC, and PACD. In Section 5.4, we summarize the discussion.

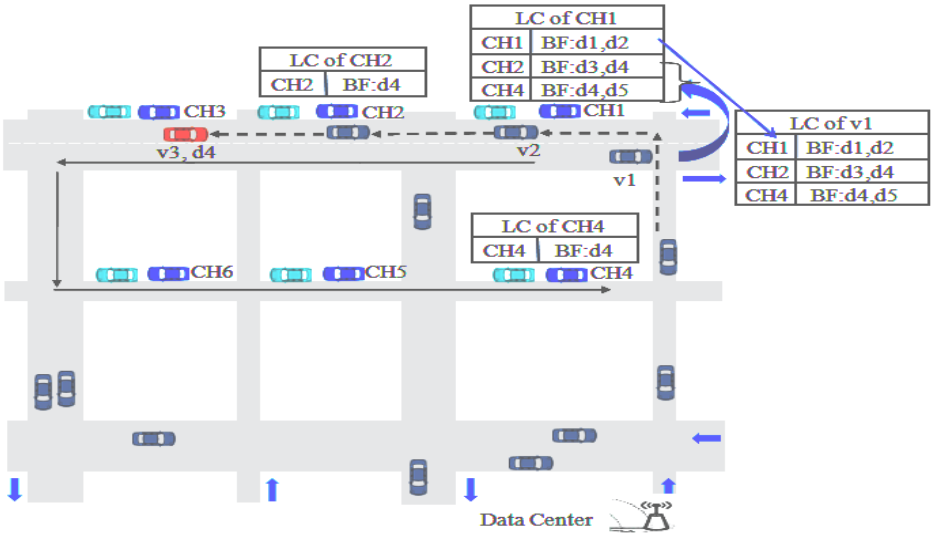
5.2 Probabilistic Cooperative Caching At Moving and Parked Vehicles (PCCMPV)

The system model adopted in PCCMPV is the same as that of the cache discovery scheme PACD that was discussed in Chapter 4. In PCCMPV, we strive to make informed caching decisions by relying on information exchange between parked and moving vehicles via beacon messages. This exchange can help reduce the redundancy of the cached contents, which can help sustain efficient usage of the storage resources. For this purpose, we use the same information exchange procedure that was adopted by PACD. The leverage of this information exchange is depicted in Figure 5.1.

In Figure 5.1(a) that illustrates a non-cooperative caching scenario, as the data packet d_4 propagates back to the requester v_3 along the data delivery path (dotted line), it encounters CH_1, which needs to determine whether or not to cache the data. Unaware that CH_2 and CH_4 already have d_4 cached in their own clusters, CH_1 caches the data. This reduces data diversity and wastes cache space due to the proximity between CH_1 and CH_2, as well as between CH_1 and CH_4. In Figure 5.1(b), consider a similar scenario using PCCMPV. Assume that vehicle v_1 has previously passed by CH_2 and CH_4, and so it already has information about the cached data at both clusters. Such information is maintained in its own LC. Afterwards, when v_1 enters the road where CH_1 resides, both CH_1 and v_1 exchange their LCs via beacon messages. Accordingly, CH_1 now knows the cached data in CH_2 and CH_4, including d_4. CH_1 then receives d_4, intended to the requester v_3. Thus, it makes an informed



(a) Non-cooperative caching.



(b) Cooperative caching using PCCMPV.

Figure 5.1: An illustrative scenario of non-cooperative versus cooperative caching.

decision and does not cache the data. Meanwhile, CH_1 and v_2 have exchanged their LCs. Consequently, v_2 also has information about the cached data in CH_2 and CH_4. Hence, when v_2 receives d_4, it checks its own trajectory (solid line), and determines that it will pass by both CH_3 and CH_4 (data holders). Also, it assesses the importance

of the other roads along its trajectory based on a number of metrics, including their proximity to CH_2 and CH_4. Thus, it does not cache d_4. In the next subsections, we provide a detailed description of the proposed cache placement procedure at parked and moving vehicles.

5.2.1 PCCMPV at Parked Vehicles

This procedure is triggered when a CH receives a data packet to be forwarded. Note that the packet forwarding procedure is the same as the interest forwarding procedure presented in the previous chapter. Two flags, referred to as caching and forwarding flags are included in the data packets to indicate whether the received data should be cached only, forwarded only, or both. Initially, only the forwarding flag is set. When a parked vehicle receives a data packet, it checks the caching and forwarding flags in the data packet. If both flags are set, then the parked vehicle caches a copy of the received data and forwards the original packet by applying the data forwarding procedure.

When a data packet is received by a CH, it determines whether or not to cache the data at its parking cluster. To do so, it calculates its own probability of caching. Such a probability represents the importance of the road segment at which the parking cluster resides. This importance is based on three metrics; the traffic density of the road segment, its closeness centrality, as well as its remoteness from the nearest data holder. The traffic density is used to reflect that the more congested the road segment is, the greater the possibility for the cached data to be hit. The closeness centrality is used to determine whether the cached content would be closely located, and thus rapidly accessed by requesters at other highly populated road segments.

This is considering that the latter do not have the data cached in their corresponding parking clusters. We focus on such road segments due to the high chance of requests occurring at or passing by them. The remoteness from the nearest data holder, including the data center point of contact, is used to ensure data diversity.

In order to calculate the probability of caching, the CH calculates three scores representing each of these metrics. Each score is a value in the range $[0, 1]$. Such a score represents the normalized value of the corresponding metric, calculated relative to that of the road segments in R, where R is the set of road segments in the road network. The traffic density score of a road segment r_j at time t is denoted $\hat{\varphi}_j^t$, its closeness centrality score is denoted $\hat{\chi}_j^t$, and its remoteness from the closest data holder score is denoted $\hat{\Gamma}_{j,d}^t$. Note that in each data packet, we include a field in its header, referred to as the caching status field, which specifies the last parking cluster along the packet's delivery path that cached the data. Initially, this field is empty.

Once a cluster head CH_j receives a data packet d, it checks the caching status field in the packet and updates its LC accordingly. If d is not already cached in the cluster, the following procedure, illustrated in Algorithm 7, is then performed:
(a) CH_j checks if a major traffic update has occurred or a change in the cache status of the road segments in R has become known to it since the last time the three scores of the road segment r_j, $\hat{\varphi}_j^t$, $\hat{\chi}_j^t$, and $\hat{\Gamma}_{j,d}^t$ were calculated (line 8). If not, the previously calculated values of the scores can be used (lines 9-11). Otherwise, CH_j recalculates them. CH_j uses Eq. 5.1 to calculate the traffic density score $\hat{\varphi}_j^t$, where φ_j^t denotes the raw (non-probabilistic) traffic density of r_j (lines 12 & 13).

$$\hat{\varphi}_j^t = \frac{\varphi_j^t - \min_{k \in R} \varphi_k^t}{\max_{k \in R} \varphi_k^t - \min_{k \in R} \varphi_k^t} \tag{5.1}$$

Algorithm 7 : PCCMPV at Parked Vehicles

1: **Input:**
2: Set of All Road Segments R
3: Reply Packet d
4: $\hat{\varphi}_i^{t_{prev}}$, $\hat{\chi}_i^{t_{prev}}$, $\hat{\Gamma}_{i,d}^{t_{prev}}$ //previous scores
5:
6: *CachePlacement(d, CH_i)*
7: **Begin**
8: **if** no update in the traffic or cache status since t_{prev} **then**
9: $\hat{\varphi}_i^t$=$\hat{\varphi}_i^{t_{prev}}$
10: $\hat{\chi}_i^t$=$\hat{\chi}_i^{t_{prev}}$
11: $\hat{\Gamma}_{i,d}^t$=$\hat{\Gamma}_i^{t_{prev}}$
12: **else**
13: calculate $\hat{\varphi}_i^t$ //Eq. 5.1
14: check LC and detect M // M=set of road segments storing d
15: **for all** $r \in R$ **do**
16: **if** $r \notin M$ and $\hat{\varphi}_i^t >$ avg **then**
17: add r to ξ
18: construct Graph $G(V, E)$ //V=$\xi \cup r_i$
19: **for all** $k \in V | k \neq j$ **do**
20: $dist_{jk}$+=$dist_{jk}$
21: χ_i^t=$1/dist_{ik}$ //Eq. 5.2
22: calculate $\hat{\chi}_i^t$ //Eq. 5.3
23: calculate $\hat{\Gamma}_{i,d}^{t_{prev}}$ //Eq. 5.4
24: calculate $P_{r_i,d}^t$, $\hat{O}_{d,t}$, and $\tilde{P}_{r_i,d}^t$ //Eq. 5.5, Eq. 5.6, and Eq. 5.7
25: **if** $\tilde{P}_{r_i,d}^t \geq th_c$ **then**
26: cache d at parked vehicle with max cache capacity in the cluster
27: **End**

It then calculates the closeness centrality score $\hat{\chi}_j^t$ of r_j. To do so, the value of the closeness centrality of r_j, before being normalized, denoted χ_j^t, is first calculated as follows: CH_j checks its LC to determine, to the best of its knowledge, the set of road segments, denoted M, that already have the data cached (line 14). The remaining roads in R are then filtered based on their traffic density. Thus, CH_j creates a set, denoted ξ, of road segments that do not have the data cached, and whose normalized traffic density is above average (lines 15-17). It then creates a graph $G(V, E)$ of $|V|$ vertices that represent the road segments in $\xi \cup r_j$, and $|E|$ edges (line 18). This graph depicts the accessibility of r_j from every other road segment (i.e., parking cluster)

in ξ. Each edge e_{xy} is associated with a weight that reflects the shortest distance between r_x and r_y, represented in the form of hop counts. As given by Eq. 5.2, CH_j calculates the raw value of the closeness centrality of r_j, denoted χ_j^t, by calculating the inverse of the sum of the shortest-path distances between r_j and the other nodes in $G(V, E)$, where $dist_{jk}$ is the shortest-path distance between r_j and r_k (lines 19-21).

$$\chi_j^t = \frac{1}{\sum_{k \in V, j \neq k} dist_{jk}} \tag{5.2}$$

CH_j then calculates the closeness centrality score $\hat{\chi}_j^t$ of r_j by applying Eq. 5.3 (line 22). It also calculates the remoteness from the nearest data holder score $\hat{\Gamma}_{j,d}^t$, as given by Eq. 5.4, where $\Gamma_{j,d}^t$ is the raw value of the score (line 23). This raw value is the minimum distance between r_j and all the possible data holders in the set $M^{'}$, $\min_{w \in M'} dist_{jw}$, where $M^{'} = M \cup$ data center.

$$\hat{\chi}_j^t = \frac{\chi_j^t - \min_{k \in R} \chi_k^t}{\max_{k \in R} \chi_k^t - \min_{k \in R} \chi_k^t} \tag{5.3}$$

$$\hat{\Gamma}_{j,d}^t = \frac{\Gamma_{j,d}^t - \min_{k \in R} \Gamma_{k,d}^t}{\max_{k \in R} \Gamma_{k,d}^t - \min_{k \in R} \Gamma_{k,d}^t} \tag{5.4}$$

(b) Once the three scores have been determined, CH_j calculates the probability of caching content d at time t due to the importance of the road segment r_j at which it resides (line 24). Such a probability is denoted $P_{r_j,d}^t$, and is calculated using Eq. 5.5, where ω_1, ω_2, and ω_3 are weighting factors in the range $(0, 1]$, such that $\Sigma_{x=1}^3 \omega_x = 1$.

$$P_{r_j,d}^t = \omega_1 \theta_i^{t,norm} + \omega_2 \psi_{i,d}^{t,norm} + \omega_3 \chi_{i,d}^{t,norm} \tag{5.5}$$

(c) CH_j then calculates the final probability of caching, denoted $\tilde{P}^t_{r_j,d}$ by multiplying $P^t_{r_j,d}$ by the normalized value of the popularity of the content d at time t, denoted $\hat{O}_{d,t}$. Note that $\hat{O}_{d,t}$ is calculated as given by Eq. 5.6, where $O_{d,t}$ is the raw value of d's popularity (line 24). $O_{d,t}$ is the number of users following the public figure that generates d. Thus, $\tilde{P}^t_{r_j,d}$ is given by Eq. 5.7 (line 24).

$$\hat{O}_{d,t} = \frac{O_{d,t} - \min_{f \in D} O_{f,t}}{\max_{f \in D} O_{f,t} - \min_{f \in D} O_{f,t}} \tag{5.6}$$

$$\tilde{P}^t_{r_j,d} = \hat{O}_{d,t} \times P^t_{r_j,d} \tag{5.7}$$

(d) CH_j refrains from caching the data at its parking cluster if $\tilde{P}^t_{r_j,d}$ is less than a certain caching threshold th_c. Otherwise, the parked vehicle that encloses the highest available caching space in the parking cluster is selected by CH_j to cache the data (lines 25 & 26). When CH_j decides where the data should be cached, it creates a copy of the data packet after setting the caching flag, and sends it to the selected parked vehicle for caching. CH_j also modifies the caching status field of the original data packet to indicate its own cluster. When there is no enough cache space in the parking cluster, replacement occurs. A Least Frequently Used (LFU) replacement policy that takes the popularity of the content into consideration is used when needed.

5.2.2 PCCMPV at Moving Vehicles

This procedure is triggered when a moving vehicle v_m receives a data packet d to be forwarded. In order to determine whether or not to cache the data, v_m calculates its own probability of caching. Such a probability is determined based on the importance of each road segment along the remaining part of its trajectory during which d remains

valid, provided that v_m is estimated to traverse the road segment before the expiry time of d. For example, if the remaining trajectory of v_m is $\{r_2, r_6, r_8\}$, the expiry time of d is t_{exp}, and v_m is estimated to arrive at r_8 after t_{exp}, then the valid remaining trajectory is $\{r_2, r_6\}$ only. Such a valid remaining trajectory is denoted $\hat{tr}^m_{rem}$.

The importance of a road segment r^m_j along $\hat{tr}^m_{rem}$ is determined based on three metrics: 1) the stand-alone importance of r^m_j, 2) the period of time that v_m is expected to spend traversing r^m_j, and 3) the chance of the data packet reaching r^m_j during the data delivery process. The first metric is considered to be the same as the aforementioned probability of caching given by Eq. 5.5. Note that this is the probability of caching d at r_j at time $t_{m,j,s}$ due to the importance of r_j, where $t_{m,j,s}$ is the time of arrival of v_m at r_j. The second metric is utilized to indicate that the longer the period of time that v_m spends moving on r^m_j, the better. This is since the cached data would then reside longer at r^m_j and can thus accommodate more users. Such a period is represented by the travel time of v_m on r^m_j, starting from its time of arrival $t_{m,j,s}$. As previously mentioned in Chapter 4, this period is denoted $\tau^{t_{m,j,s}}_j$, and is given by Eq. 4.2. The third metric is used to reflect the notion that the less the chance that the data packet will pass by r^m_j during the data delivery process, the higher the probability of caching should be. This is because the decision to cache d at v_m would then grant the road segment r^m_j another chance to obtain and hold d. Such a possibility is determined based on the remoteness of r^m_j from the packet destination. This is because the more distant the road segment is, the lower the possibility of it receiving the data packet. This is since the data packet is typically forwarded towards the destination.

The aforementioned metrics are calculated in the form of three scores in the range

$[0, 1]$. Such scores represent the normalized values of each metric, calculated for the road segment r_j^m, relative to the road segments in R. The stand-alone importance score of r_j^m is denoted $\hat{\Theta}_{j,d}^{t_{m,j,s}}$, its travel time score is denoted $\hat{\eta}_j^{t_{m,j,s}}$, and its distance score is denoted $\hat{\Psi}_{j,d}^{t_{m,j,s}}$. The probability of caching d at v_m due to the individual importance of each $r_j^m \in \hat{tr}_{rem}^m$, denoted $P_{v_m,j,d}^{t_{m,j,s}}$, is determined based on these three scores. Note that such an importance is equal to zero if the road segment already has the data cached in the corresponding parking cluster. This is provided that the existing replica does not expire while v_m is on the road segment. This is since if that occurred, a cached replica in the vehicle would be useful for r_j^m, as it would make up for the expired one.

When a data packet d is received by a moving vehicle v_m, the latter updates its LC based on the information provided in the caching status field associated with the data packet. If v_m does not already have d in its local cache, it performs the following procedure, as demonstrated in Algorithm 8:

(a) v_m determines its valid remaining trajectory $\hat{tr}_{rem}^m$ (lines 7-15). In order to do that, it first detects its original remaining trajectory $\hat{tr}_o^m$ before omitting any parts that need to be removed to ensure the validity constraint. In addition, its total travel time along $\hat{tr}_o^m$, denoted $\tau_{m,o}^{total}$, is determined. Note that $\tau_{m,o}^{total}$ is the sum of the travel time of v_m along each road segment in $\hat{tr}_o^m$. The vehicle v_m then calculates the amount of time that d would spend in its cache before it expires, denoted $t_{m,valid}^d$. It does so by determining the minimum value among the time-to-live of d, TTL_d and $\tau_{m,o}^{total}$. For example, if the validity time of d is 20 minutes and the total travel time of v_m along T_m^{ro} is 30 minutes, then d would be valid at v_m for 20 minutes only. The road segments in $\hat{tr}_o^m$ whose travel time does not exceed $t_{m,valid}^d$ are sequentially included

Algorithm 8 : PCCMPV at Moving Vehicles

1: **Input:**
2: TTL of the Data, TTL_d
3: Reply Packet d
4:
5: *CachePlacement(d, v_m)*
6: **Begin**
7: determine $\hat{tr}_o^m$ //original remaining trajectory of v_m
8: calculate $\tau_{m,o}^{total}$ //total travel time along trajectory $\hat{tr}_o^m$
9: $t_{m,valid}^d$=$min(TTL, \tau_{m,o}^{total})$ //time during which d is valid
10: **for all** $r_k^m \in \hat{tr}_o^m$ **do**
11: $\tau_{m,rem}^{total}$+=$L_k/V_k^{t_{m,k,s}}$ // travel time of trajectory $\hat{tr}_{rem}^m$
12: **if** $\tau_{m,rem}^{total} \leq t_{m,valid}$ **then**
13: add r_k to $\hat{tr}_{rem}^m$
14: **else**
15: break
16: **for all** $r_j^m \in \hat{tr}_{rem}^m$ **do**
17: get $t_{m,j,s}$ and $t_{m,j,e}$ // using the travel time at each road
18: check LC
19: **if** d is cached at r_j^m **then**
20: **if** $t_{exp,d} \geq t_{m,j,e}$ **then**
21: $P_{v_m,j,d}^{t_{m,j,s}} = 0$
22: **else if** $t_{exp,d} < t_{m,j,e}$ **then**
23: $t_{m,j,s}$=$t_{exp,d}$
24: **if** $P_{v_m,j,d}^{t_{m,j,s}}$ has not been assigned **then**
25: calculate $\hat{\Theta}_{j,d}^{t_{m,j,s}}$, $\hat{\eta}_j^{t_{m,j,s}}$, $\hat{\Psi}_{j,d}^{t_{m,j,s}}$ // Eq. 5.8, Eq. 5.9, Eq. 5.10
26: calculate $P_{v_m,j,d}^{t_{m,j,s}}$ // Eq. 5.11
27: $P_{v_m,j,d}^{t_{m,j,s}}$+=$P_{v_m,j,d}^{t_{m,j,s}}$ // calculate sum
28: Calculate $Prob_{v_m,d}^t$, $\hat{O}_{d,t}$, $\widetilde{Prob}_{v_m,d}^t$ // Eq. 5.12, Eq. 5.6, Eq. 5.13
29: **if** $\widetilde{Prob}_{v_m,d}^t \geq th_c$ **then**
30: cache d at v_m
31: **End**

in $\hat{tr}_{rem}^m$.

(b) For each road segment $r_j^m \in \hat{tr}_{rem}^m$ (line 16), v_m applies the following steps: (1) Calculate the time of arrival and departure to and from r_j^m, $t_{m,j,s}$ and $t_{m,j,e}$, respectively (line 17). (2) Check the LC to determine whether r_j^m already has the data d cached in its parking cluster, and whether the expiry time of d, denoted $t_{exp,d}$, exceeds $t_{m,j,e}$. If this is the case, set $P_{v_m,j,d}^{t_{m,j,s}}$ to zero (lines 18-21). (3) If r_j^m already has the

data cached but $t_{exp,d}$ is less than $t_{m,j,e}$, let $t_{m,j,s}$ be equal to $t_{exp,d}$ (lines 22 & 23). (4) If case (3) occurs or case (2) does not apply, go to step (5) and (6) (lines 24-26). (5) Calculate the three aforementioned scores $\hat{\Theta}_{j,d}^{t_{m,j,s}}$, $\hat{\eta}_{j}^{t_{m,j,s}}$, and $\hat{\Psi}_{j,d}^{t_{m,j,s}}$, by applying Eq. 5.8, Eq. 5.9, and Eq. 5.10, respectively. Note that $P_{r_j,d}^{t_{m,j,s}}$ is determined based on Eq. 5.5.

$$\hat{\Theta}_{j,d}^{t_{m,j,s}} = P_{r_j,d}^{t_{m,j,s}} \tag{5.8}$$

$$\hat{\eta}_{j}^{t_{m,j,s}} = \frac{\eta_{j}^{t_{m,j,s}} - \min_{g \in R} \eta_{g}^{t_{m,g,s}}}{\max_{g \in R} \eta_{g}^{t_{m,g,s}} - \min_{g \in R} \eta_{g}^{t_{m,g,s}}} \tag{5.9}$$

$$\hat{\Psi}_{j,d}^{t_{m,j,s}} = \frac{\Psi_{j,d}^{t_{m,j,s}} - \min_{g \in R} \Psi_{g,d}^{t_{m,g,s}}}{\max_{g \in R} \Psi_{g,d}^{t_{m,g,s}} - \min_{g \in R} \Psi_{g,d}^{t_{m,g,s}}} \tag{5.10}$$

(6) Calculate $P_{v_m,j,d}^{t_{m,j,s}}$ using Eq. 5.11. Determine the probability of caching d at v_m at time t, denoted $Prob_{v_m,d}^{t}$, by calculating the average of the individual caching probabilities at the road segments belonging to $\hat{tr}_{rem}^{m}$, as given by Eq. 5.12, where $c = |\hat{tr}_{rem}^{m}|$ (lines 27 & 28).

$$P_{v_m,j,d}^{t_{m,j,s}} = \begin{cases} 0 & d \text{ is cached}, \\ & t_{exp,d} \geq t_{m,j,e} \\ \omega_4 \hat{\Theta}_{j,d}^{t_{m,j,s}} + \omega_5 \hat{\eta}_{j}^{t_{m,j,s}} + \omega_6 \hat{\Psi}_{j,d}^{t_{m,j,s}} & \text{Otherwise} \end{cases} \tag{5.11}$$

$$Prob_{v_m,d}^{t} = \frac{\sum_{z \in \hat{tr}_{rem}^{m}} P_{v_m,z,d}^{t_{m,z,s}}}{c} \tag{5.12}$$

(c) v_m then calculates the final probability of caching, denoted $\widetilde{Prob}_{v_m,d}^{t}$ by multiplying $Prob_{v_m,d}^{t}$ by $\hat{O}_{d,t}$, that is calculated as given by Eq. 5.6 (line 28). Thus, $\widetilde{Prob}_{v_m,d}^{t}$ is

given by Eq. 5.13 (line 28).

$$\widetilde{Prob}^{t}_{v_m,d} = \hat{O}_{d,t} \times Prob^{t}_{v_m,d} \tag{5.13}$$

(d) If $\widetilde{Prob}^{t}_{v_m,d}$ is less than the caching threshold, th_c, the data will not be cached. Otherwise, v_m caches the data (lines 29 & 30). A LRU replacement policy is used if needed.

5.3 Performance Evaluation

In this section, we evaluate the performance of PCCMPV compared to CADD [35] and DPC [9]. Both CADD and DPC have been discussed in Section 2.2.2. We have selected CADD for comparison because it inherits some implicit features of cooperative caching. This is advantageous since cooperative caching has rarely been explored within the context of VANETs. Meanwhile, DPC is a representative of non-cooperative caching, and it has been shown to outperform a number of caching schemes in VANETs, including the baseline reactive caching scheme that follows a cache all policy [33]. Note that CADD enables caching to occur at static roadside caching units only, while DPC adopts caching at moving vehicles only. Accordingly, since our proposed PCCMPV scheme exploits both moving and parked vehicles for caching, we implement a hybrid approach of both CADD and DPC, where static nodes use the caching policy adopted in CADD, and moving vehicles adopts the one used in DPC. This is to ensure a fair comparison. We refer to this combination of both CADD and DPC as CADPC.

The comparison of PCCMPV to CADPC is used to show how explicit cooperative caching performs compared to implicit, as well as to non-cooperative caching in

VANETs. Note that the cache discovery approach applied in CADPC is the one used in both CADD and DPC, which is the server-based approach (Section 2.3.1).

In order to evaluate the performance of the proposed cache discovery schemes, CCD and PACD, that were presented in the previous chapter, we implement both of them while using PCCMPV for cache placement. We refer to each of them as PCCMPV-CCD and PCCMPV-PACD. For the purpose of exploring the effect of expanding the search space beyond the neighborhood scope, we also implement the neighborhood-restricted tracking-based cache discovery approach adopted by the GroupCaching (GC) scheme while using PCCMPV for cache placement. We refer to it as PCCMPV-GC. GC has been commonly used in MANETs, and it is considered to be adequately applicable to VANETs [12]. This is since it yields much less overhead for highly dynamic networks than other cooperative cache discovery schemes [12].

It is worth mentioning that the exchange of LCs (i.e., the cached content information of encountered parking clusters) via beacon messages is employed by PCCMPV, and this information exchange is utilized by the cache discovery module in PACD but not in CCD. In fact, CCD only enables the vehicles to exchange their own cached content information. Thus, we also implement CCD with LCs, and we refer to it as PCCMPV-CCDLC. This is to show the effect of the further expansion of the cooperation scope that the exchange of LCs achieves. In addition, comparing PACD to CCDLC facilitates demonstrating the effect of the predictive approach proposed in PACD.

We also assess the performance of the proposed trajectory prediction scheme incorporated in PACD. In order to do that, we implement the same cache discovery employed in PACD using the Regional Markov Model (RMM) [106] for comparison.

This is since RMM is a clustering-based trajectory prediction technique that has been shown to outperform other state-of-the-art trajectory prediction schemes in VANETs [106]. Note that RMM has been discussed in Section 4.2. We refer to the use of PACD with RMM for prediction as RPACD, and since we also apply it with PCCMPV for cache placement, the entire scheme is referred to as PCCMPV-RPACD. In summary, for the purpose of evaluating the proposed cache discovery schemes, we compare PCCMPV-GC, PCCMPV-CCD, PCCMPV-CCDLC, PCCMPV-PACD, and PCCMPV-RPACD. For simplicity, during discussion, we will refer to those schemes using their cache discovery names only; GC, CCD, CCDLC, PACD, and RPACD.

In order to show the impact of using bloom filters during the information exchange process, only PACD is executed using them. Note that we ignore the amount of overhead rendered in PCCMPV in all schemes, and we focus on the overhead yielded due to the cache discovery approach used. Since CCDLC adopts the same information exchange module as PACD, comparing CCDLC to PACD can demonstrate whether or not using bloom filters can help reduce the overhead, and to what extent.

For the purpose of assessing the performance of the proposed cache placement scheme only, without taking the cache discovery aspect into consideration, we also implement CADPC with the cache discovery scheme used in GC. We refer to it as CADPC-GC. This enables PCCMPV-GC to be compared to CADPC-GC. Note that the explicit exchange of cached content information in CADPC-GC is also used to cache the data only when none of the neighbors of a node has it. In summary, for the purpose of evaluating the cache placement scheme, we compare PCCMPV-GC to CADPC and CADPC-GC.

Assessments and comparisons are done in terms of the following performance

metrics: 1) the average delay, which is the average time taken starting from the request generation time until the requested data is acquired, 2) the packet delivery ratio, which is the ratio of the number of data packets successively procured by requesters to the total number of data packets issued, 3) the cache hit ratio, which is the ratio of the number of data packets retrieved from caching nodes to the total number of data packets acquired, 4) the beacon overhead, which is the ratio of the total extra information exchanged among vehicles for the purpose of cooperative caching to the total amount of information exchanged, including the original amount typically included in beacon messages in VANETs, and 5) the prediction accuracy, which is the ratio of the number of accurate predictions to the total number of predictions.

5.3.1 Simulation Setup

The NS-3 network simulator [64] is used to implement and evaluate all the aforementioned schemes. For the road topography, we use the same settings and parameters presented in Chapter 3. Table 5.1 summarizes the simulation parameters. A 6×6 grid topography with 120 edges is created. Realistic vehicular mobility traces are generated using the traffic simulator SUMO [86]. Simulations are carried out for a total simulation period of 2000 seconds each. We use the IEEE 802.11p WAVE standard with a communication range of 200 meters. Two types of beacon messages are sent out; the original beacon message with no additional information, and the one with extra information due to cooperative caching. Either one of the two types of beacon messages is used, depending on the corresponding beacon time interval. The former is sent every 1 second, and the latter is sent every 5 seconds as a replacement to the former. The original size of the beacon message is 500 bytes, since this lies within its

Table 5.1: Simulation parameters of PCCMPV, CCD, and PACD

Simulation Parameters	**Value**
Dimensions of the Road Topography	6×6
Simulation Time	2000 sec
Communication Technology	IEEE 802.11p WAVE
Communication Range	200 m
Original Beacon Interval	1 sec
Extra Information Beacon Interval	5 sec
Original Beacon Size	500 Bytes
Number of Requesters	200
Number of Public Figures	50
TTL of Contents	5−7 minutes
Content Name Size	30 Bytes
Number of Parked Vehicles	240
Bloom Filter Size (b)	24 Bytes
Number of Hash Functions (y)	3
Caching Threshold (th_c)	0.25
Number of Trips	1000
Number of Clusters (C)	25
Clustering Fuzzification Coefficient (ϑ)	2
Clustering Termination Threshold (ϵ)	0.001
Membership Threshold (U_{th})	0.2

typical size range [133][134].

The number of requesters is set to 200, and the interest generation is distributed among 50 public figures on social media based on a Zipf-like distribution with a skewness factor=0.5. Each public figure creates a new post every $5 - 7$ minutes rendering the previous one obsolete. The size of each content name is set to 30 bytes. Note that Named Data Networking (NDN) is capable of supporting a maximum name length of up to 30 bytes [130]. The time of each request is set to a random value within the requester's trip duration. Unless otherwise specified, the number of moving vehicles is set to $600/km^2$, and 240 parked vehicles are uniformly distributed

among all road segments. Each parked vehicle resides at its designated parking space throughout the entire simulation period. We express the cache capacity of moving vehicles, RCSs in CADPC, and the collection of parked vehicles at each road segment in terms of the percentage β of the total contents that can be requested. Unless otherwise specified, β is set to 60%, the percentage of road segments with parked vehicles is 100%, and the order of the Markov model ℓ is set to 3. The values of ∇, α_1, and α_2 in CCD and CCDLC are set to 2 minutes, 0.2 and 0.8, respectively.

The bloom filter size b is set to 24 bytes and the number of hash functions y is set to 3. The caching threshold th_c is set to 0.25. We created 1000 trajectory sequences, 80% were used for training, while the remaining 20% were used for testing. In the clustering algorithm ARCA, we set the number of clusters C to 25, the fuzzification coefficient ϑ to 2, the termination threshold ϵ to 0.001, and the membership threshold U_{th} to 0.2. In PCCMPV, the weighting factors ω_1-ω_3 are set to 0.3, 0.4, and 0.3, respectively, while ω_4-ω_6 are set to 0.4, 0.3, and 0.3, respectively.

5.3.2 Results and Discussion

In our experiments, we evaluate the performance of PCCMPV, CCD, CCDLC, and PACD under varying vehicular densities, cache capacity percentage β, percentage of road segments with parked vehicles κ, and Markov model order ℓ. The results obtained are presented below. Simulation results are presented at a confidence level=90%.

1- The Impact of Vehicular Density

In this experiment, we vary the number of moving vehicles from 200 to 1000 to assess the performance of PCCMPV, CCD, and PACD under varying vehicular

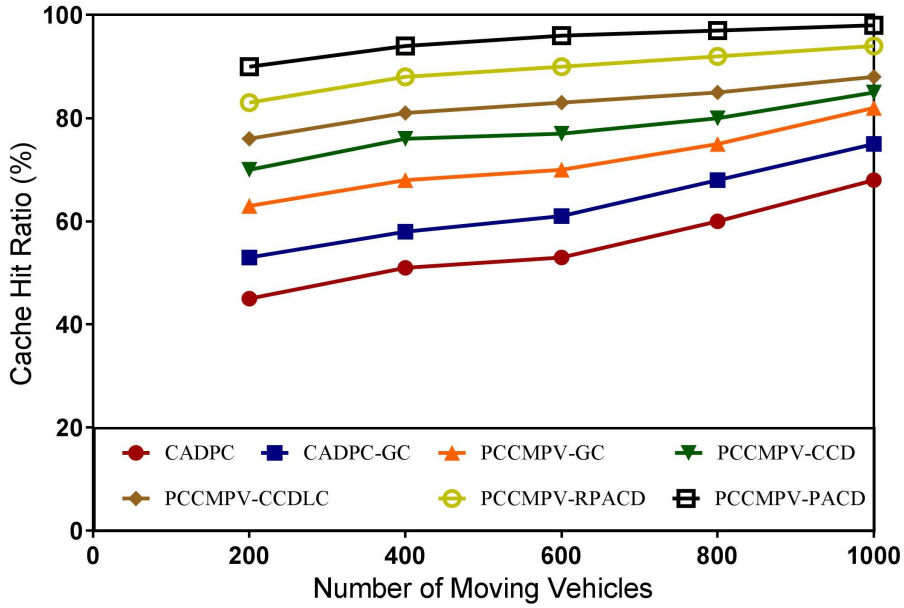


Figure 5.2: Cache hit ratio over varying vehicular densities.

densities. We set the cache capacity percentage β to 40% to also evaluate their performance given a small cache capacity.

Figure 5.2 depicts the effect of this variation on the performance in terms of cache hit ratio. As shown in the Figure, PCCMPV-GC yields a significant increase of up to 40% and 20% in cache hit ratio compared to CADPC and CADPC-GC, respectively. This can be attributed to the more informed caching decisions made in PCCMPV due to the use of explicit cooperative caching, as well as the extension of the cooperation range beyond the neighborhood scope. Such an extension is facilitated by the exchange of the vehicles' own cached content information, as well as that of other parking clusters maintained in their LCs. This enables caching more diverse data at valuable nodes, which reduces the wasted cache capacity, and leads to more cache hits. As the number of moving vehicles increases, more cached content information can be exchanged. This increases data availability, as it enables more efficient utilization of

the vehicles' cache capacity, which improves the cache hit ratio. In contrast, CADPC incurs higher delay than CADPC-GC and PCCMPV due to the lack of any form of explicit collaboration between the nodes in both CADD and DPC. Also, CADPC uses a server-based cache discovery scheme, which relies on the opportunistic encounter with caching nodes along the data delivery path to the data center. This limits the search space compared to CADPC-GC and PCCMPV-GC, and reduces the chances of finding caching nodes, which in turn reduces cache hits.

CADPC-GC performs better than CADPC due to the explicit exchange of cached content information among the neighboring nodes in GC, which increases data diversity compared to CADPC. Also, the neighborhood-restricted tracking-based cache discovery approach adopted in GC increases the search space compared to the server-based scheme used in CADPC, which further increases cache hits.

As shown in Figure 5.2, CCD outperforms PCCMPV-GC by up to 17%. This is because the tracking-based cache discovery procedure employed in CCD expands the search space beyond the neighborhood scope. Such an expansion is attributed to the stable tracking service provided by parked vehicles, which tracks the location of caching nodes. This leads to extending the lifetime of the cached content information, which helps sustain the expanded search space for a longer time, and in turn makes it easier to locate the caching nodes. Note that as the number of moving vehicles increases, road segments get more congested and thus vehicles tend to slow down. This increases the validity time of the last encounter information registered in the vehicles' TPP in CCD, which leads to more reliable tracking of the caching nodes. In contrast, in PCCMPV-GC, once two neighboring nodes move out of range, their cached content information gets immediately nullified.

CCDLC further improves the cache hit ratio, with an increase of up to 27% and 10%, compared to PCCMPV-GC and CCD, respectively. This is due to the exchange of the LCs maintained by vehicles, along with their own cached content information, which enables the vehicles in CCDLC to also track the cached data at static parked vehicles that they have not necessarily encountered.

PACD achieves the highest cache hit ratio among all the other cache discovery schemes, with an increase of up to 47%, 30%, and 20% compared to PCCMPV-GC, CCD, and CCDLC, respectively. This is attributed to the fact that PACD increases the chances of data acquisition from caching nodes by eliminating any restrictions on the search space during the data discovery process. This is since it follows a prediction-assisted tracking-based scheme, which enables reaching caching nodes wherever they are located. In contrast, CCD and CCDLC limit the search space to the extent of the trails left by moving vehicles at the CHs they encounter.

The same cache discovery scheme used in PACD is employed in RPACD, but with a different prediction scheme. As depicted in Figure 5.2, PACD outperforms RPACD by up to 11%. This is because RPACD renders a lower prediction accuracy than PACD, which increases the risk of having to navigate the requests to the data center when the locations of caching nodes are not accurately predicted. As the number of moving vehicles increases, more cached content information is exchanged among vehicles, which provides them with more options for possible data providers to select from in their TPP. This enables vehicles to select caching nodes that have lower entropy, and thus a higher certainty in their predicted location.

It is worth mentioning that PCCMPV-CCD outperforms CADPC and CADPC-GC in terms of cache hit ratio, by up to 56% and 32%, respectively, while PCCMPV-CCDLC outperforms them by up to 69% and 43%, respectively. PCCMPV-PACD significantly increases cache hit ratio compared to both CADPC and CADPC-GC, by up to 100% and 70%, respectively. This is due to all the aforementioned reasons related to both the cache placement scheme PCCMPV, and each of the cache discovery schemes CCD, CCDLC, and PACD.

We conduct the same experiment to assess the performance of PCCMPV, CCD, and PACD in terms of average delay. As shown in Figure 5.3, PCCMPV-GC outperforms both CADPC and CADPC-GC by up to 31%, and 23%. This is because of the significant increase in the cache hit ratio, due to the aforementioned reasons. Such an increase facilitates acquiring the data from nearby caching nodes rather than the far-away data center. As the number of moving vehicles increases, the amount of exchanged cached content information increases, which improves caching decisions, and thus improves the delay.

As shown in Figure 5.3, CCD significantly reduces the delay compared to PCCMPV-GC, with a reduction of up to 30%. This can be attributed to the fact that CCD adopts a tracking procedure that facilitates finding nearby caching nodes that can be located beyond the neighborhood scope. This increases the potential of data acquisition from nearby caching nodes, which further improves the delay. This is as opposed to the restricted search space in PCCMPV-GC, where a vehicle can fail to find a nearby caching node if none of its neighbors has the requested data. The average delay is further improved by CCDLC, with a reduction of up to 39% and 18% compared to PCCMPV-GC and CCD, respectively. This is due to the LCs exchanged among

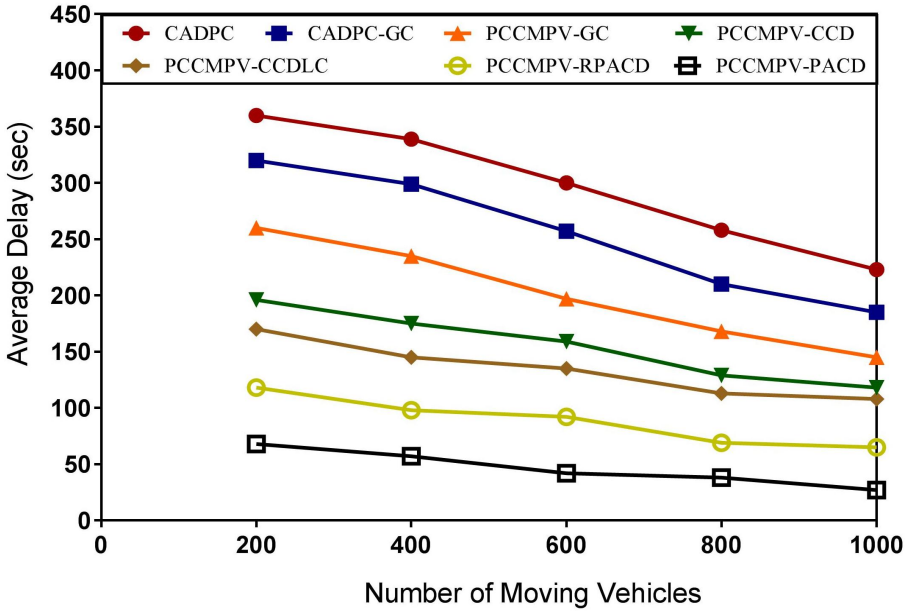


Figure 5.3: Average access delay over varying vehicular densities.

vehicles in CCDLC, which provides the vehicles with information about more potential caching nodes that can be much closer to the requester, and can in turn reduce the delay even further. Such information increases as the number of moving vehicles increases. Thus, more delay reduction is yielded under higher vehicular densities.

PACD yields the least amount of delay compared to the remaining schemes, with an improvement of up to 81%, 77%, and 70% compared to PCCMPV-GC, CCD, and CCDLC, respectively. This can be attributed to the significant improvement in cache hit ratio, and the unrestricted search space facilitated by the incorporated prediction technique. Also, the ranking process plays an important role as it emphasizes on replicas within closer proximity to the requester. This is in contrast to the tracking procedure in CCD and CCDLC that relies on navigating the requests to the CH (i.e., parked vehicle) that has last seen it, rather than predicting its actual position. Accordingly, data acquisition in PACD has higher chances of being acquired from

caching nodes closer to the requester than all the other schemes.

PACD renders a lower delay than RPACD, with a reduction of up to 42%. This is due to the higher prediction accuracy rendered by PACD (as will be demonstrated later on). Such an improved prediction accuracy reduces the risk of navigating requests to the wrong location of a caching node, thus forcing it to be redirected to another, or to the far-away data center. Note that due to traffic congestion, vehicles tend to slow down as the vehicular density increases. This extends the lifetime of the exchanged cached content information in PCCMPV-GC. It also increases the chance of tracking mobile caching nodes before they navigate too far away from the last trails they leave behind in CCD and CCDLC, or before too many predicted locations are accumulated in PACD and RPACD. Thus, the delay decreases as the number of vehicles increases.

Evidently, PCCMPV-PACD significantly improves the delay by up to 88% and 83% compared to CADPC and CADPC-GC. This can be attributed to all the aforementioned reasons related to PCCMPV and PACD.

We evaluate the packet delivery ratio by performing the same experiment. As depicted in Figure 5.4, as the number of vehicles increases, PCCMPV-GC increases the packet delivery ratio by up to 33%, and 15% compared to CADPC and CADPC-GC, respectively. This can be attributed to the highly improved delay in PCCMPV-GC, which in turn reduces the chance of dropping data packets due to the inability to find the requester. This typically occurs when the requester moves too far away from its request initiation position. The earlier the data arrives, the less the chance for this to occur. Lower delay also reduces the risk of failure to reach the requesting vehicle due to the fact that its trip has already ended.

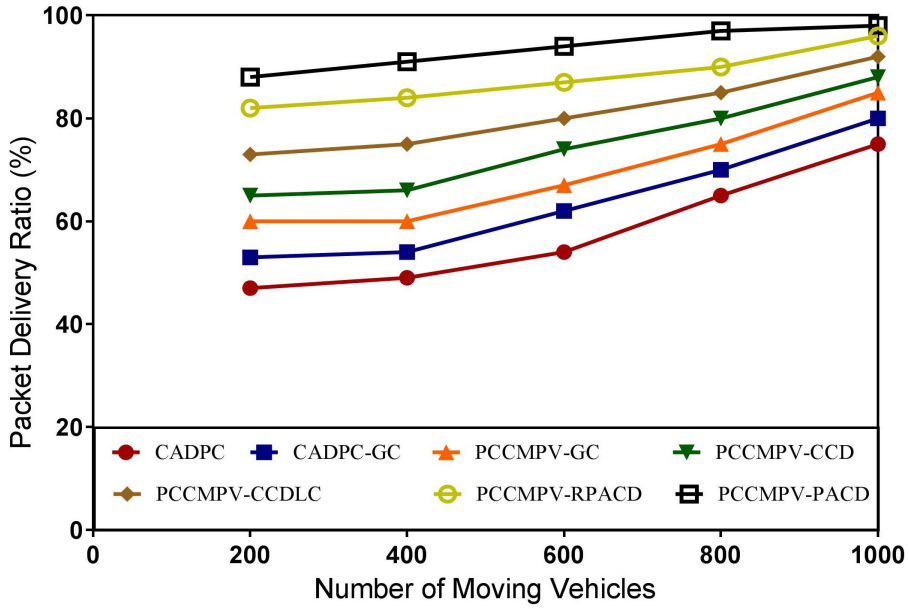


Figure 5.4: Packet delivery ratio over varying vehicular densities.

As depicted in Figure 5.4, as the vehicular density increases, CCD increases the packet delivery ratio by up to 14% compared to PCCMPV-GC. This is due to the significant improvement in delay yielded by CCD compared to PCCMPV-GC. In addition, the tracking procedure applied in CCD as a part of the cache discovery process provides a means to track the requester's position. This is achieved through a rather stable tracking service leveraged by the static nature of parked vehicles. As a result, the risk of dropping the data packets is reduced.

CCDLC further improves the packet delivery ratio by up to 22% and 12% compared to CCD. This is due to the reduced delay in CCDLC, which increases the chance of data packets reaching the designated requesters before their trips are terminated. PACD renders the highest packet delivery ratio among all schemes, yielding an improvement of up to 47%, 35%, and 25% compared to PCCMPV-GC, CCD, and CCDLC. This can be attributed to the much lower delay it achieves, as well as the

prediction-based tracking procedure that it adopts to locate the requester, which diminishes the risk of dropping the data packet. As a result of the higher prediction accuracy that PACD yields compared to RPACD, it improves the packet delivery ratio by up to 10% over the latter.

Note that due to all the aforementioned reasons pertaining to the performance of PCCMPV and PACD, PCCMPV-PACD significantly improves the packet delivery ratio by up to 87% and 66% compared to CADPC and CADPC-GC.

We consider the same comparison to evaluate the amount of beacon overhead triggered due to information exchange. As depicted in Figure 5.5, CADPC does not involve any explicit exchange of cached content information, so it does not render any extra beacon overhead. In contrast, CADPC-GC, PCCMPV-GC, and CCD all render the same amount of overhead, since in all of them neighboring vehicles exchange their own cached content information. Note that such an overhead slightly increases as the number of moving vehicles increases due to the increase in the total amount of exchanged cached content information, which can slightly affect the beacon overhead ratio.

Due to the fact that each vehicle also exchanges with its neighbors the cached content information pertaining to the parking clusters maintained in its LC, CCDLC increases the overhead by up to 159% compared to PCCMPV-GC. The exchange of LCs also applies to PACD. However, the use of bloom filters in PACD ensures a much lower overhead compared to CCDLC, with a reduction of up to 57%. Since we adopt the use of bloom filters that have a fixed size, the amount of information exchanged or maintained in the caches does not have an effect on the beacon overhead. Thus, the beacon overhead yielded by PACD remains the same as the vehicular density inc-

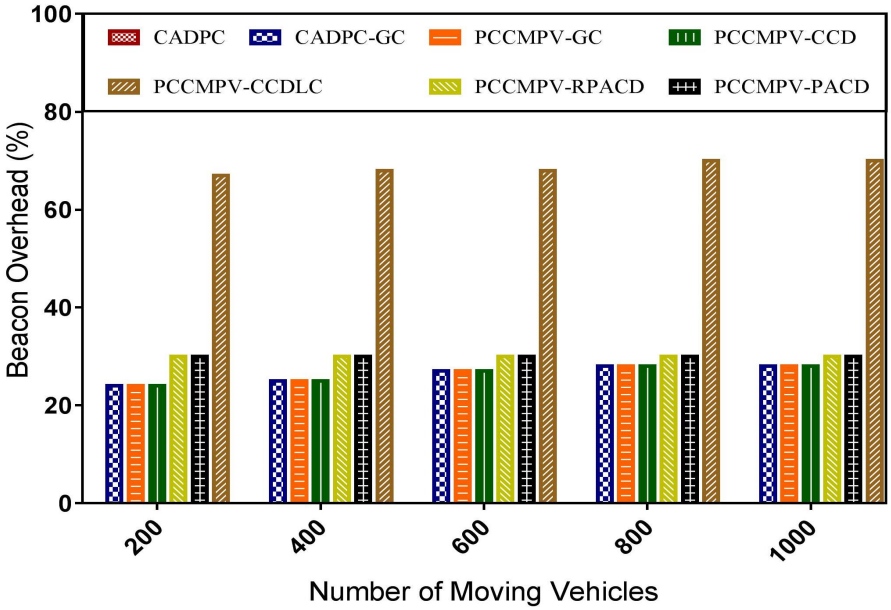


Figure 5.5: Beacon overhead over varying vehicular densities.

reases.

Despite the reduction in overhead that PACD achieves compared to CCDLC, its overhead is still higher than that of PCCMPV-GC and CCD by up to 15%. This is because the number of content names that can be included in GC and CCD is low, since in this experiment, only 40% of the cache capacity can be used. Note that since PACD and RPACD are the same in everything except the prediction technique, RPACD renders the same amount of beacon overhead as PACD.

We perform the same experiment in order to evaluate the performance in terms of prediction accuracy. Note that PACD and RPACD are the only schemes that involve prediction. Hence, they are the only ones considered in this experiment. As depicted in Figure 5.6, as the number of moving vehicles increases, the prediction accuracy in-

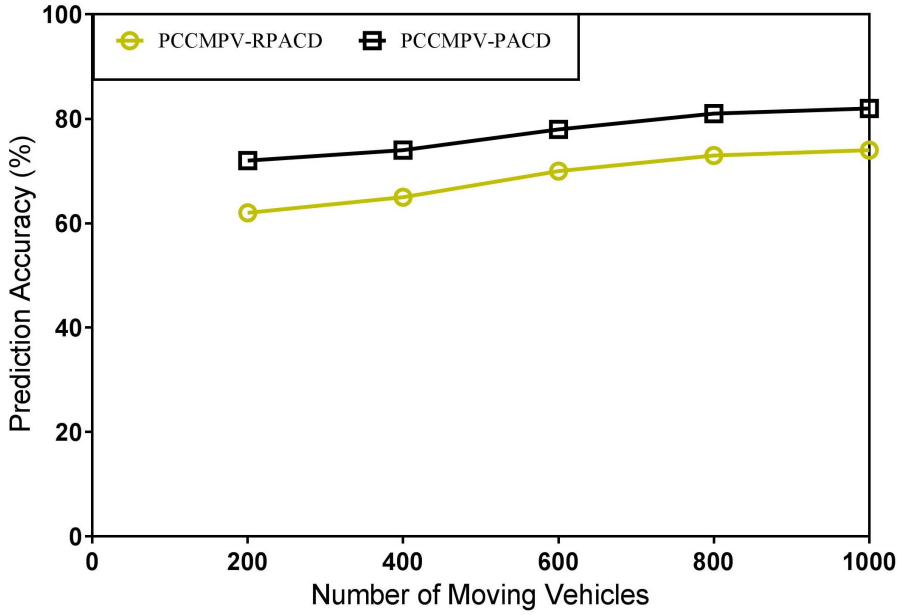


Figure 5.6: Prediction accuracy over varying vehicular densities.

creases in both schemes. This can be attributed to the fact that the speed of vehicles becomes lower as the traffic becomes more congested. This reduces the number of road segments that vehicles traverse since their last time of encounter, which facilitates more accurate predictions due to the less reliance on ℓ predicted road segments, and more reliance on known or partially known ℓ road segments for prediction. PACD improves the prediction accuracy compared to RPACD, with an increase of up to 15%. This can be attributed to three main reasons. First, the use of the XXDice similarity measure in PACD, which can achieve higher precisions compared to the Minimum Edit Distance (MED) that is used in RPACD. This is due to the lack of sequential context sensitivity in MED in many situations [119][121]. Second, the realization of more accurate representation of the trajectory clusters in PACD. This

is due to the use of a soft clustering algorithm, as opposed to the hard clustering one used in RPACD. Third, the use of the MTD-probit model in the training procedure adopted in PACD, which takes the frequency of a given prediction sequence in the historical data into consideration, as opposed to the VLMC model used in the training procedure in RPACD [116][122][123][127].

2- The Impact of Cache Capacity

In this experiment, we vary the cache capacity percentage β from 20% to 100% to assess the performance of PACD under low, medium, and high cache capacity.

Figure 5.7 demonstrates the effect of this variation on the cache hit ratio. As shown in Figure 5.7, the cache hit ratio diminishes as β decreases in all schemes. This is since as β decreases, vehicles can host less amount of data in their caches. This makes it imperative for cache placement schemes to make efficient utilization of the available storage resources so as to reduce the amount of wasted cache space. As a result, PCCMPV-GC manages to yield a much higher cache hit ratio than CADPC and CADPC-GC, with an increase of up to 48% and 20%, respectively. This is also due to the much extended cooperation range offered by the stable residence of cached content information, including the vehicles' LCs, at parked vehicles. This is as opposed to the neighborhood-restricted range in CADPC-GC and the lack of any explicit cooperation in CADPC.

In addition to the aforementioned reasons, PCCMPV implements an implicit form of off-path caching, along with the typical on-path caching methodology. This extends the potential candidates for caching a replica, as it takes into consideration the road segments that moving vehicles are expected to traverse along their trajectory. Thus,

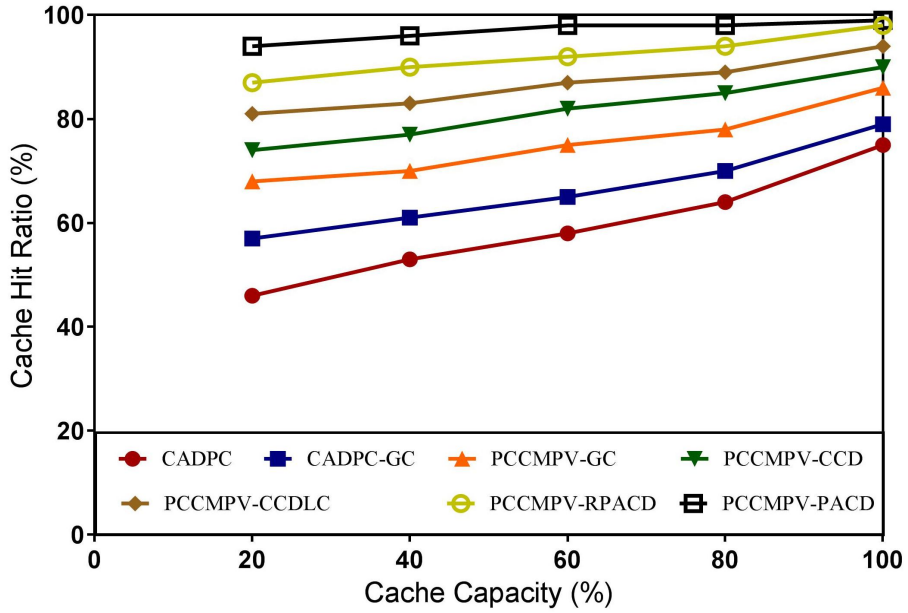


Figure 5.7: Cache hit ratio over varying cache capacity (β).

the cache capacity utilization increases even further. Furthermore, PCCMPV considers the traffic density at road segments. This, along with considering the content popularity, increase the possibility of caching the data where matching requests are triggered or encountered.

As shown in Figure 5.7, as β decreases, the cache hit ratio is reduced in all cache discovery schemes. This is because as β decreases, the need for an expanded search space increases, since the nodes cannot cache many contents. Thus, the chance of finding the requested data within a limited search space decreases. Accordingly, CCD improves the cache hit ratio by up to 14% compared to PCCMPV-GC.

Due to the exchange of the vehicles' LCs in CCDLC, which enables vehicles to track more cached contents in the network, it outperforms both PCCMPV-GC and CCD by up to 23%, and 10%, respectively. With its unrestricted search space and prediction technique, PACD further increases the cache hit ratio by up to 42%, 30%,

and 18%, respectively compared to PCCMPV-GC, CCD, and CCDLC.

PACD also outperforms RPACD by up to 8%. This is due to the lower prediction accuracy rendered in RPACD. Note that the lower the cache capacity, the less the potential data providers of a particular content that are maintained in a vehicle's TPP. This, along with the lower prediction accuracy yielded by RPACD, cause the vehicles to navigate requests to caching nodes that do not necessarily render a low entropy, but rather the lowest among the potential candidates. Accordingly, failure to find the caching nodes at their estimated position might occur, thus forcing more requests to eventually be directed to the data center.

Due to the aforementioned leverages gained by PCCMPV and PACD, Figure 5.7 also shows that PCCMPV-PACD outperforms CADPC and CADPC-GC by up to 104% and 65%, respectively.

We conduct the same experiment to assess the average access delay of the proposed schemes. As depicted in Figure 5.8, the delay decreases as β increases in all schemes. This is attributed to the higher cache hits, which increase the chance of acquiring the data from caching nodes rather than the remote data center. PCCMPV-GC yields the lowest delay among CADPC and CADPC-GC, with an improvement of up to 41% and 29%, respectively. This is because, along with the higher cache hit ratio achieved by PCCMPV as β increases, it also makes caching decisions while taking the closeness centrality of road segments into consideration. This helps increase data acquisition from caching nodes that reside within closer proximity to the requester, which further reduces the delay. Note that as β increases, the potential of selecting caching nodes located at road segments with high closeness centrality increases, since their cache capacity can accommodate more contents.

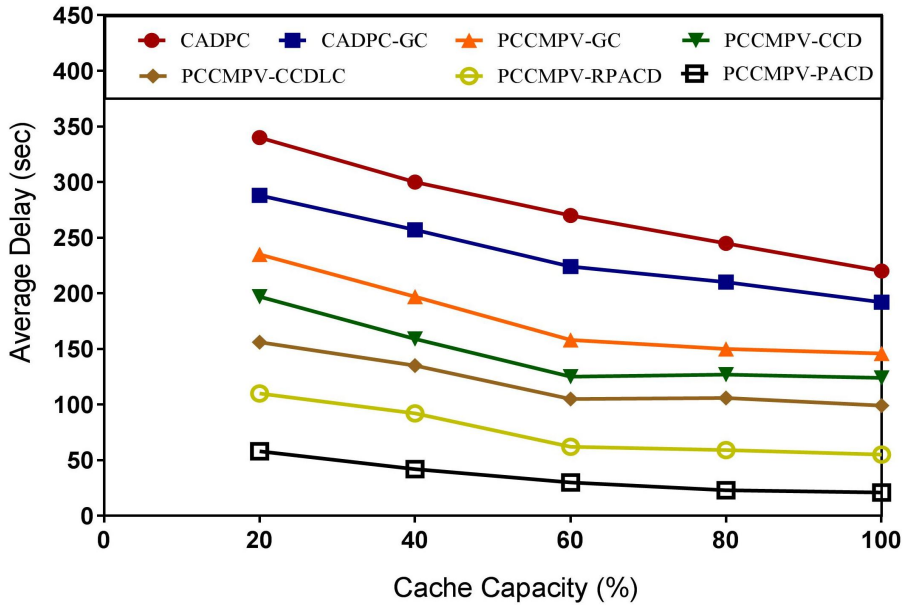


Figure 5.8: Delay over varying cache capacity (β).

Figure 5.8 also shows that CCD achieves a lower delay than PCCMPV-GC, with an increase reaching up to 21%. This is because in contrast to GC, vehicles in CCD keep track of previously encountered caching nodes even when they move out of range, and dynamically rank them based on their proximity to the requester. Thus, the incorporated tracking procedure and ranking process in CCD dynamically detect closer replicas to the requester. Note that this is done while taking the age of information into consideration. Thus, the ranking process might favor a further replica than a closer one if the cached content information about the former is more recent. Such a risk increases if there are only few potential data providers to select from. Thus, as β increases, this risk decreases, since the chance of having more potential data providers in the vehicles' TPP increases.

Due to the exchange of the LC maintained by vehicles, the aforementioned risk

decreases even further in CCDLC. Consequently, as shown in Figure 5.8, CCDLC renders a lower delay than PCCMPV-GC and CCD, with an improvement that reaches up to 34% and 21%, respectively. However, once the caching nodes maintained in the TPP navigate far away from the next road segment to that where the CH that has last seen them resides, the tracking procedure in CCD and CCDLC fails to locate nearby caching nodes. In contrast, with its underlying prediction technique, PACD can perform its tracking procedure without any restrictions, and can still be able to dynamically select a closer replica to the requester. Accordingly, PACD outperforms PCCMPV-GC, CCD, and CCDLC by up to 86%, 82%, and 74%, respectively. Note that the increase in the number of caching nodes in the vehicles' TPP because of the increase in β, reduces the risk of favoring a more remote replica to the requester due to its lower entropy than a closer one with higher entropy. Also, PACD improves the delay by up to 54% compared to RPACD. This is since RPACD renders a lower prediction accuracy than PACD. Hence, the aforementioned risk has a higher occurrence rate in the former than the latter, particularly at low values of β, thus triggering a higher delay. It is worth mentioning that PCCMPV-PACD improves the delay by up to 90% and 83%, over CADPC and CADPC-GC, respectively.

In order to assess the packet delivery ratio under varying β, we perform the same experiment. As depicted in Figure 5.9, as β increases, the packet delivery ratio increases. Note that PCCMPV-GC achieves the best performance among CADPC and CADPC-GC, yielding an increase of up to 38% and 19%, respectively. This is due to the lower delay achieved by PCCMPV-GC as β increases, which in turn reduces the risk of dropping the data packets. As previously mentioned, this could be because the data packets manage to reach the requesters before they move too far away from

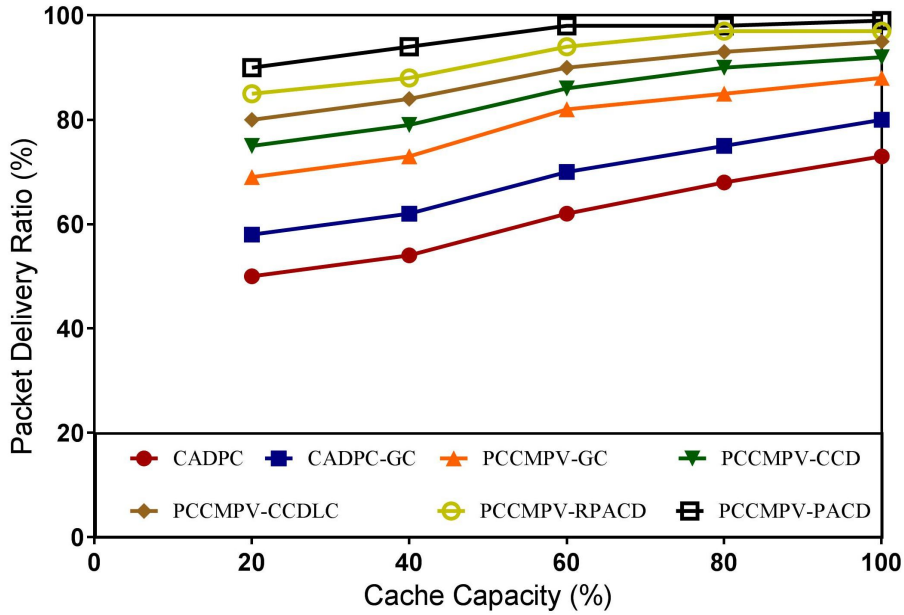


Figure 5.9: Packet delivery ratio over varying cache capacity (β)

their position at the time of the request initiation, or data packets reach the requesting vehicles before they conclude their trips and leave the system. Since CCD and CCDLC achieve a lower delay than PCCMPV-GC, and provide a means to track the requester's location, this risk is further reduced. Thus, CCD improves the packet delivery ratio by up to 12% compared to PCCMPV-GC. Also, as CCDLC renders an even lower delay than CCD, it yields a higher packet delivery ratio, with an increase of up to 17%, and 7% over PCCMPV-GC and CCD, respectively.

Figure 5.9 also shows that as β increases, the packet delivery ratio of PACD increases, where it yields the best performance among all the other schemes. This is due to the significant reduction in delay achieved by PACD as β increases, as well as its prediction-based tracking procedure of the requester's location. Thus, PACD outperforms PCCMPV-GC, CCD, and CCDLC by up to 30%, 20%, and 13%, respectively.

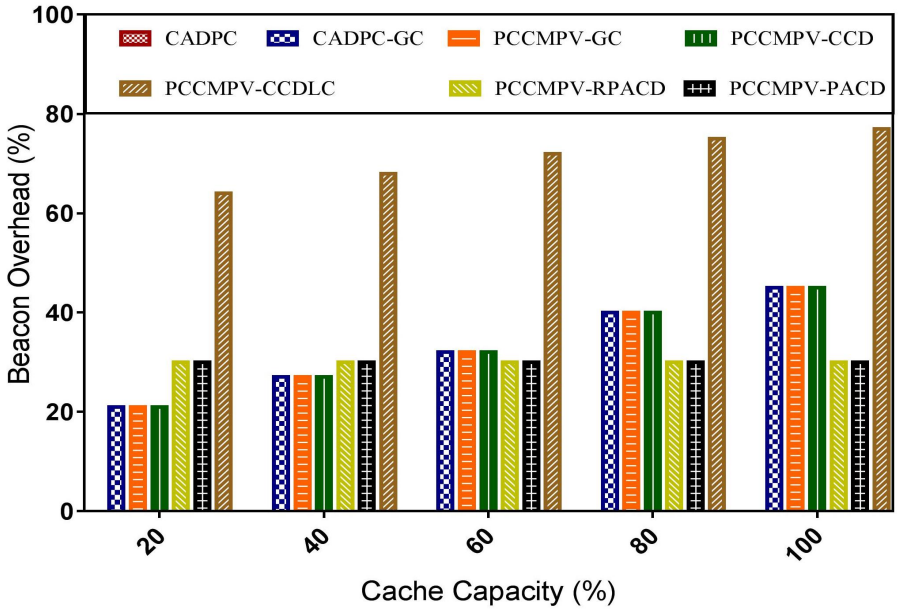


Figure 5.10: Beacon overhead over varying cache capacity (β)

Since PACD can more accurately predict the location of the requester than RPACD, higher packet delivery ratio is rendered in the former, with an increase of up to 6%. Note that as β increases, the gap between PACD and RPACD decreases. This is due to the significantly low delay achieved by both schemes, which increases the chances that the requester is still within close vicinity to its original position, thus reducing the need for predicting its location. As depicted in Figure 5.9, and due to all the aforementioned reasons, PCCMPV-PACD outperforms CADPC and CADPC-GC by up to 75% and 55%, respectively

We conduct the same experiment to evaluate the amount of beacon overhead. Due to the lack of any explicit exchange of cached content information in CADPC, it does not involve any beacon overhead. As depicted in Figure 5.10, the amount of overhead in CADPC-GC, PCCMPV-GC, and CCD increases as β increases. This can be attributed to the increase in the number of content names that can be added to

beacon messages to reflect the vehicles' cached content information. Since in the three schemes, vehicles exchange their own cached content information only, they render almost the same amount of overhead. CCDLC yields the highest amount of overhead among all schemes. This is due to the fact that neighboring vehicles exchange their own cached content information, as well as that of other parking clusters sustained in their LCs. Since such LCs are also exchanged in PACD but using bloom filters, the amount of overhead yielded in PACD is significantly reduced compared to CCDLC, with a reduction of up to 62%.

Note that varying β has no effect on PACD. This is since no matter how much information is cached, the size of the exchanged bloom filters remains the same. Thus, at lower values of β, PACD yields higher beacon overhead than GC and CCD, with an increase of up to 32%. However, as β increases, PACD starts to render a lower overhead than GC and CCD, with a reduction of up to 40%. This is due to the leverage gained by the use of bloom filters in parsimoniously representing the cached content information, as opposed to their increase in the other schemes. The same applies to RPACD, since the only difference between PACD and RPACD is the adopted prediction technique.

3- The Impact of the Percentage of Road Segments with Parked Vehicles

In this experiment, we vary the percentage of road segments that have parked vehicles, κ, from 20% to 100% in order to study the effect of the scalability of parked vehicle-occupied road segments on the performance of the proposed schemes. Note that this variation does not have any effect on CADPC, and CADPC-GC, since neither of these scheme involves the use of parked vehicles.

We evaluate the impact of the variation of κ on the cache hit ratio of the proposed

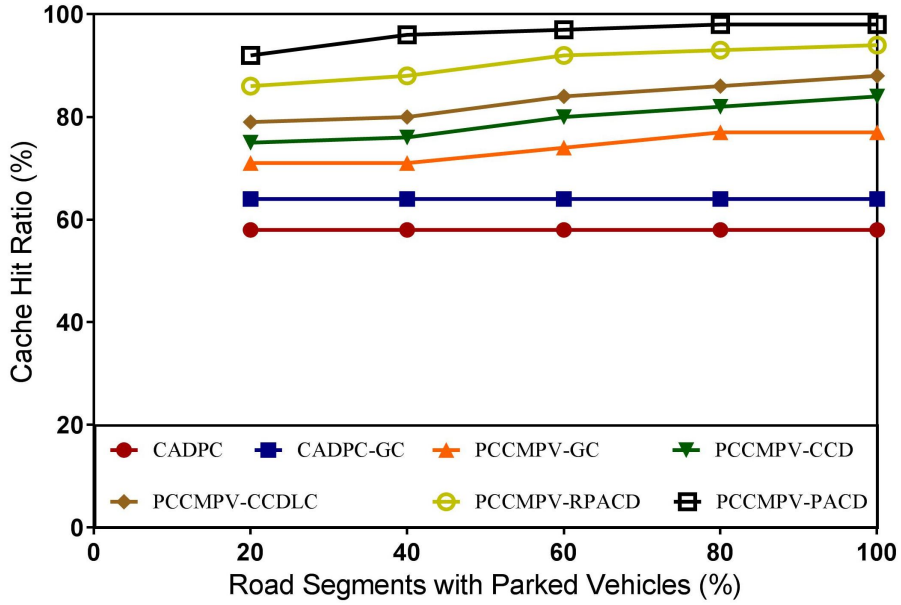


Figure 5.11: Cache hit ratio over varying percentage of road segments with parked vehicles (κ).

schemes. As shown in Figure 5.11, as κ increases, the cache hit ratio increases in all schemes. This can be attributed to the fact that the lower the value of κ, the lower the data availability at road segments due to the lack of parked vehicles to cache the data at. This reduces the number of road segments that have the contents. However, PCCMPV exploits the trajectory of vehicles to assess the importance of the road segments they pass by in terms of their traffic density and closeness centrality. Thus, PCCMPV-GC still performs better than CADPC and CADPC-GC, yielding an increase of up to 32% and 20%, respectively. Note that the cache placement scheme in PCCMPV-GC, CCD, CCDLC, and PACD rely on the availability of parked vehicles to provide a stable residence for the exchanged cached content information. This is in order to ensure more diffusion of such information into the network. Thus, the lower the value of κ, the lower the amount of exchanged cached content information

among neighboring vehicles, which reduces data diversity and increases the amount of wasted cache space. These factors can force more data to be acquired from the data center rather than caching nodes, which reduces cache hits.

The lower the value of κ, the higher the reliance on moving vehicles for caching, which makes the mission of locating such nodes during the cache discovery process more crucial. Thus, as depicted in Figure 5.11, CCD yields a higher cache hit ratio than PCCMPV-GC, with an increase of up to 10%. This is due to the ability of CCD to track mobile caching nodes beyond the neighborhood scope, as opposed to GC. However, since the cache discovery scheme used in CCD relies on the breadcrump traces that moving vehicles leave at the CHs they pass by, the decrease in κ can cause such traces to be shortly interrupted, which can reduce cache hits. This also applies to CCDLC. However, since CCDLC further enables the use of the exchanged LCs among neighboring vehicles in the discovery process, it increases the cache hit ratio by up to 6% compared to CCD. Note that as κ decreases, the amount of cached content information about parking clusters that is maintained in LCs decreases, thus the cache hit ratio decreases.

The capability of PACD to predict the location of mobile caching nodes leads to a significant increase in the cache hit ratio, even at low values of κ. In fact, it increases the cache hit ratio by up to 31%, 26%, and 20% compared to PCCMPV-GC, CCD, and CCDLC. In addition, PACD outperforms RPACD by up to 11%, due to its higher prediction accuracy of the location of mobile caching nodes.

The same experiment is conducted in order to asses the average access delay. As depicted in Figure 5.12, as κ increases, the average delay decreases in PCCMPV-GC, CCD, CCDLC, PACD, and RPACD. This is due to the higher cache hit ratio that is

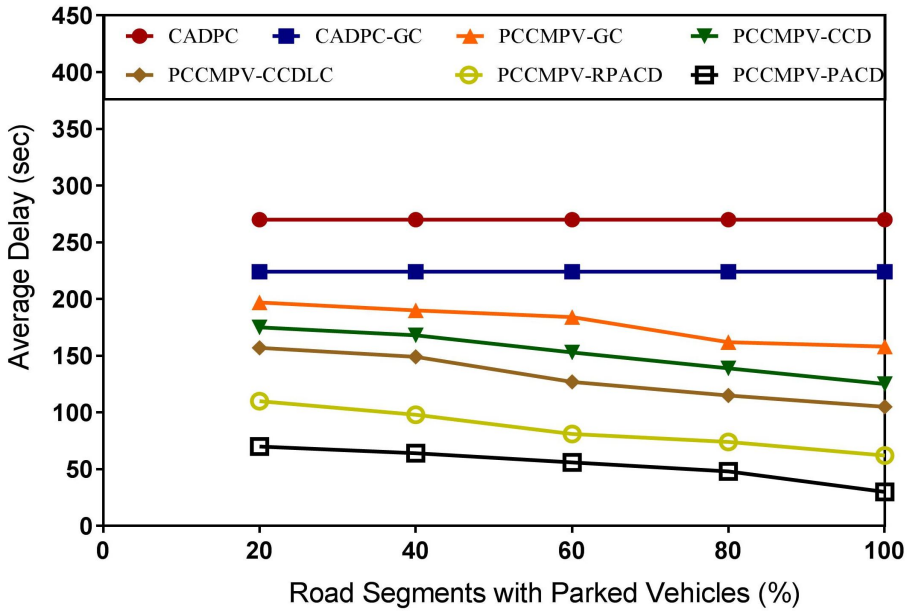


Figure 5.12: Average delay over varying percentage of road segments with parked vehicles (κ).

achieved by them as κ increases, which decreases the risk of acquiring the data from the remote data center. Accordingly, CCD reduces the delay by up to 29% compared to PCCMPV-GC. Also, the higher reliance on moving vehicles for data acquisition as κ decreases, increases the risk of sending the request to the wrong estimated location of a mobile caching node in CCD, thus triggering the need for it to be redirected to another caching node or the distant data center, which increases the delay. The use of LCs in CCDLC during the discovery process reduces this risk by providing information about the cached contents at the available parking clusters. Thus, CCDLC reduces the delay by up to 17% compared to CCD.

PACD further improves the delay by up to 80%, 72%, and 68%, compared to PCCMPV-GC, CCD, and CCDLC, respectively. This is because PACD has a higher chance of locating moving vehicles that have the data cached. Note that as κ decreases

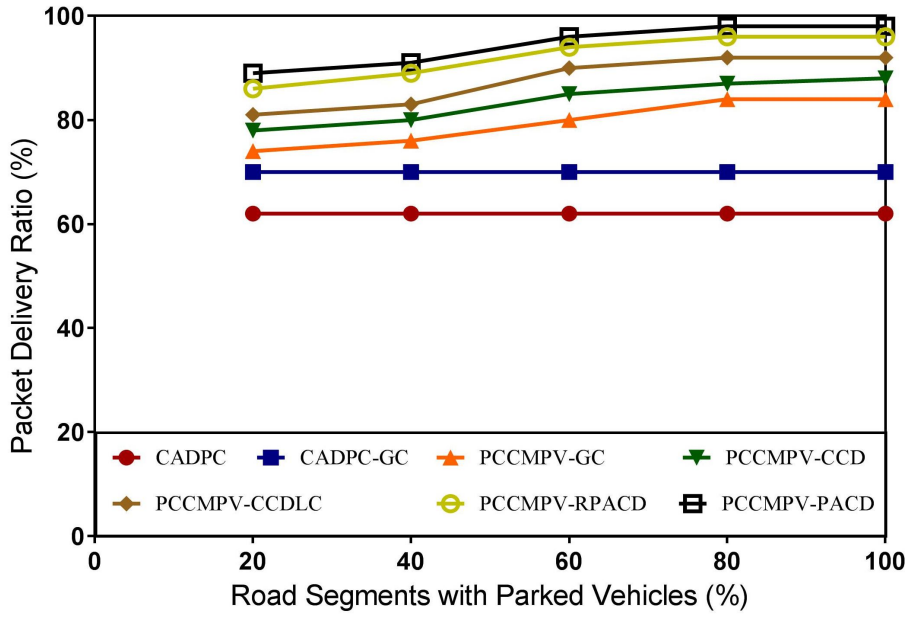


Figure 5.13: Packet delivery ratio over varying percentage of road segments with parked vehicles (κ).

in PACD, the aforementioned risk increases. This problem further exacerbates as the prediction accuracy decreases. Thus, PACD outperforms RPACD by up to 50%.

We assess the packet delivery ratio subject to the same experiment. As demonstrated in Figure 5.13, the packet delivery ratio increases as κ increases. This is due to the significant reduction in the average delay, which reduces the risk of dropping the data packets. As opposed to PCCMPV-GC, CCD provides a tracking procedure of the requester. Thus, it increases the packet delivery ratio by up to 10%. However, the lower the value of κ, the higher the chance that the trails left by the requester at encountered CHs get abruptly interrupted. This increases the risk of failure to find the requester, thus dropping the data packet. The reduced delay achieved by CCDLC as κ increases reduces this risk. Consequently, CCDLC improves the packet delivery

ratio by up to 14% and 8%, respectively.

Along with the ability to predict the requester's location without the need to rely on CHs for tracking, PACD and RPACD also manage to achieve a significant reduction in delay. This increases the chance of reaching the requester before it moves too far away from its location at the time of the request initiation, or too far ahead along its trajectory. Thus, PACD and RPACD perform almost the same, with PACD slightly outperforming the latter by up to 4% due to its higher prediction accuracy. PACD also outperforms PCCMPV-GC, CCD, and CCDLC by up to 20%, 14%, and 10%, respectively. Moreover, PACD outperforms CADPC and CADPC-GC by up to 58% and 40%, respectively.

In order to evaluate the impact of varying κ on the beacon overhead, the same experiment is conducted. As depicted in Figure 5.14, CADPC does not involve any beacon overhead, while the beacon overhead in CADPC-GC is not affected by varying κ since there are no parked vehicles in it. PCCMPV-GC and CCD perform almost the same, since they only require neighboring nodes to exchange their own cached content information. As κ decreases, the beacon overhead slightly decreases in both schemes. This is due to the decrease in the number of exchanged beacon messages, since less vehicles (i.e., parked vehicles) become involved in the information exchange procedure. This decrease reaches up to 9% compared to CADPC-GC. Since the size of the extra information added to each beacon message is not profoundly affected, the ratio between the total extra information exchanged among vehicles to the total amount of information exchanged in beacon messages, does not significantly decrease.

As shown in Figure 5.14, the overhead decreases as κ decreases in CCDLC. This is because as κ decreases, the cached content information maintained in LCs decreases,

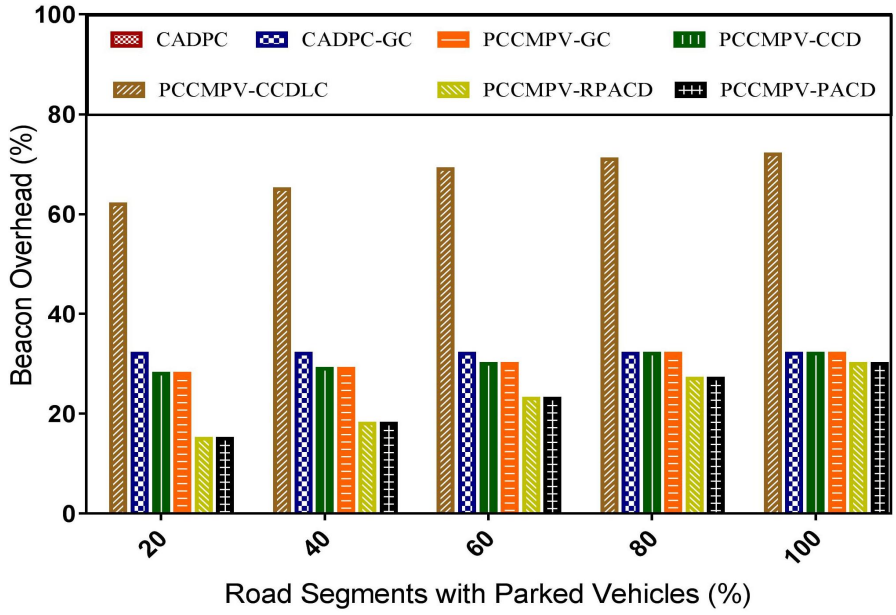


Figure 5.14: Beacon overhead over varying percentage of road segments with parked vehicles (κ).

which in turn decreases the size of the extra information exchanged in beacon messages. However, despite this decrease, CCDLC still yields the highest overhead among all schemes, with an increase of up to 125% compared to PCCMPV-GC and CCD.

PACD outperforms CCDLC even though it also involves the exchange of LCs. This is because PACD adopts the use of bloom filters, which reduces the size of the incorporated extra information. As shown in Figure 5.14, this reduction manifests even further as κ decreases. This can be attributed to the fact that PACD includes a fixed-sized bloom filter to represent the cached content information of each parking cluster in the LCs. Consequently, as κ decreases, the maximum number of parking clusters in LCs decreases, and thus the number of bloom filters that PACD includes in a given beacon message decreases. Hence, the use of bloom filters in PACD enables

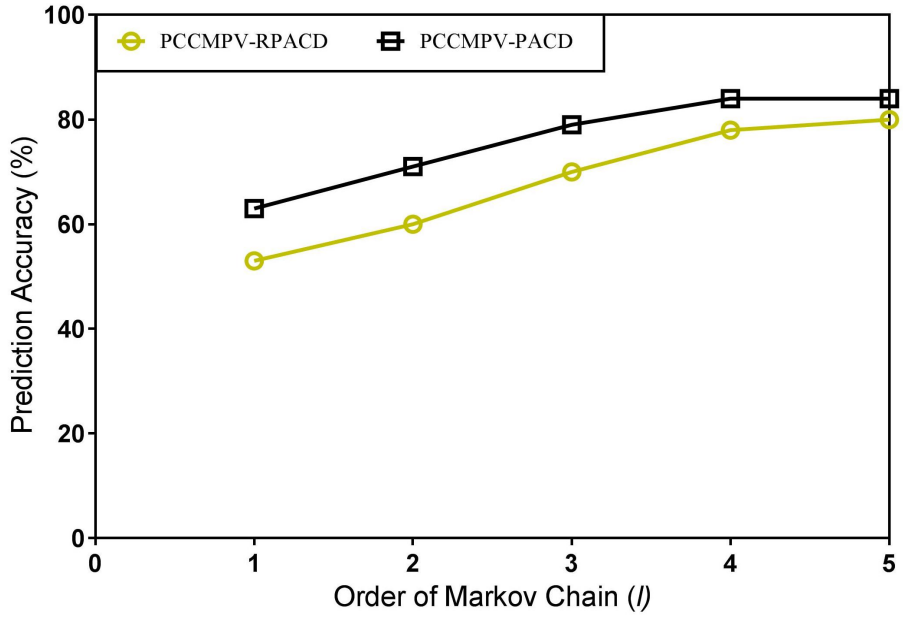


Figure 5.15: Prediction Accuracy over varying order of Markov chain (ℓ).

it to reduce the beacon overhead by up to 76% compared to CCDLC. Also, since less number of bloom filters is used, the gap between PACD and PCCMPV-GC, as well as CCD increases, till it yields a reduction of up to 47% compared to both of them. Note that RPACD and PACD yield the same amount of overhead. This is since they adopt the same information exchange procedure.

4- The Impact of the Order of the Markov Model (ℓ)

In this experiment, we vary the order of the Markov model ℓ from 1 to 5. Note that this variation does not have any effect on CADPC, CADPC-GC, PCCMPV-GC, CCD, and CCDLC, since no prediction is involved in any of these schemes. In contrast, prediction is incorporated in RPACD and PACD only. Thus, they are the only schemes considered in this experiment.

As shown in Figure 5.15, the prediction accuracy increases as ℓ increases, with PACD yielding a significant increase of up to 16% compared to RPACD. This is

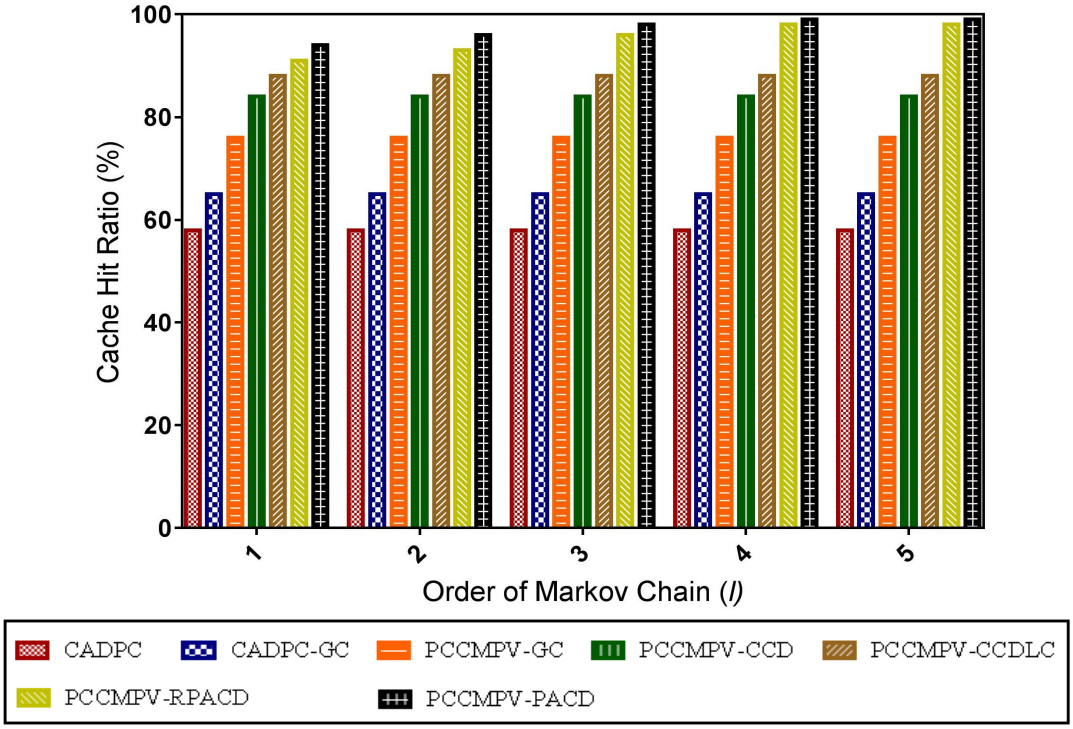


Figure 5.16: Cache hit ratio over varying order of Markov chain (ℓ).

because as ℓ increases, more historical information pertaining to the partial trajectory traversed by vehicles is provided, thus enabling more accurate predictions. Note that ℓ=1 indicates the use of the first-order Markov model. Thus, in this case, the superiority of PACD over RPACD is solely attributed to the leverage of both the XXDice similarity measure and the soft clustering technique ARCA, compared to the MED approach and the hard clustering technique used in RPACD. Such a leverage increases as ℓ increases, since the precision of the XXDice similarity measure, and thus that of ARCA increases. When ℓ is above 1, the MTD-probit model also starts to show some leverage compared to the VLMC model. This is due to the same previously mentioned reasons.

Figure 5.16 demonstrates the effect of varying ℓ on the cache hit ratio. As shown in

the Figure, the cache hit ratio increases as ℓ increases. PACD triggers the highest cache hit ratio, yielding an increase of up to 71%, 52%, 30%, 18%, 13%, and 5% compared to CADPC, CADPC-GC, PCCMPV-GC, CCD, CCDLC, and RPACD, respectively. This is because as ℓ increases, the prediction accuracy increases, which reduces the risk of reaching the predicted location of a caching node only to find that it is not actually there. This can increase the risk of eventually having the interest packet redirected to the remote data center, which could reduce cache hits.

As shown in Figure 5.16, the gap between PACD and RPACD is relatively small. This can be attributed to the fact that the cache discovery process is dynamically performed at every intermediate node along the data delivery path. Accordingly, the interest packet can be redirected to another caching node rather than the distant data center in case of failure to reach another designated caching node. Another reason is the ranking process that enables vehicles to favor a more distant caching node than a closer one based on the entropy (i.e., uncertainty) of the predicted location.

The aforementioned reasons could trigger a higher access delay in RPACD compared to PACD, as depicted in Figure 5.17. Note that this delay is reduced as ℓ increases, due to the higher prediction accuracy. In fact, Figure 5.17 shows that PACD renders the lowest delay among CADPC, CADPC-GC, PCCMPV-GC, CCD, CCDLC, and RPACD, with a reduction of up to 94%, 92%, 88%, 83%, 79%, and 57%, respectively.

As depicted in Figure 5.18, the packet delivery ratio increases as ℓ increases. Note that PACD yields the highest packet delivery ratio, with a significant increase of up to 96%, 60%, 50%, 30%, 25%, and 14% compared to CADPC, CADPC-GC, PCCMPV-GC, CCD, CCDLC, and RPACD, respectively. The leverage gained by PACD over

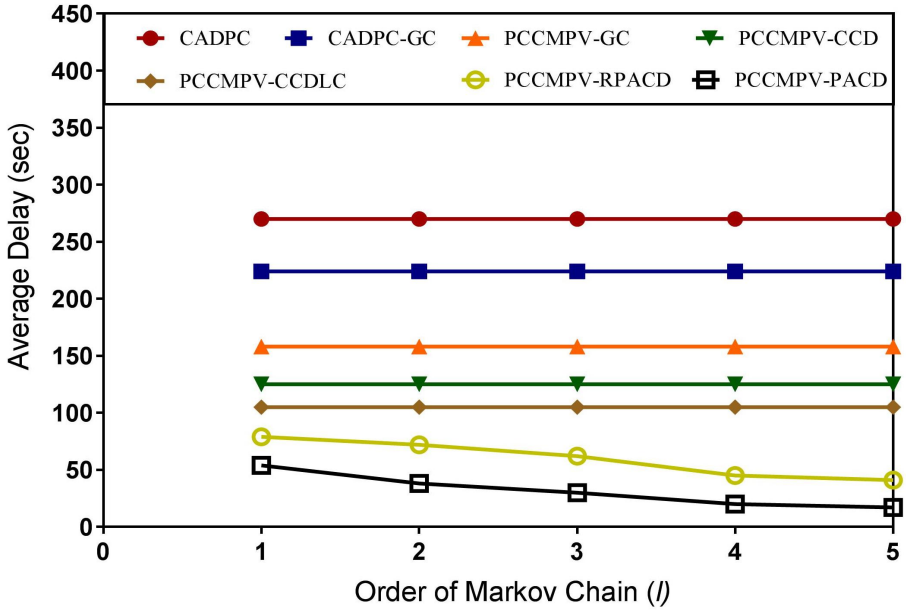


Figure 5.17: Average delay over varying order of Markov chain (ℓ).

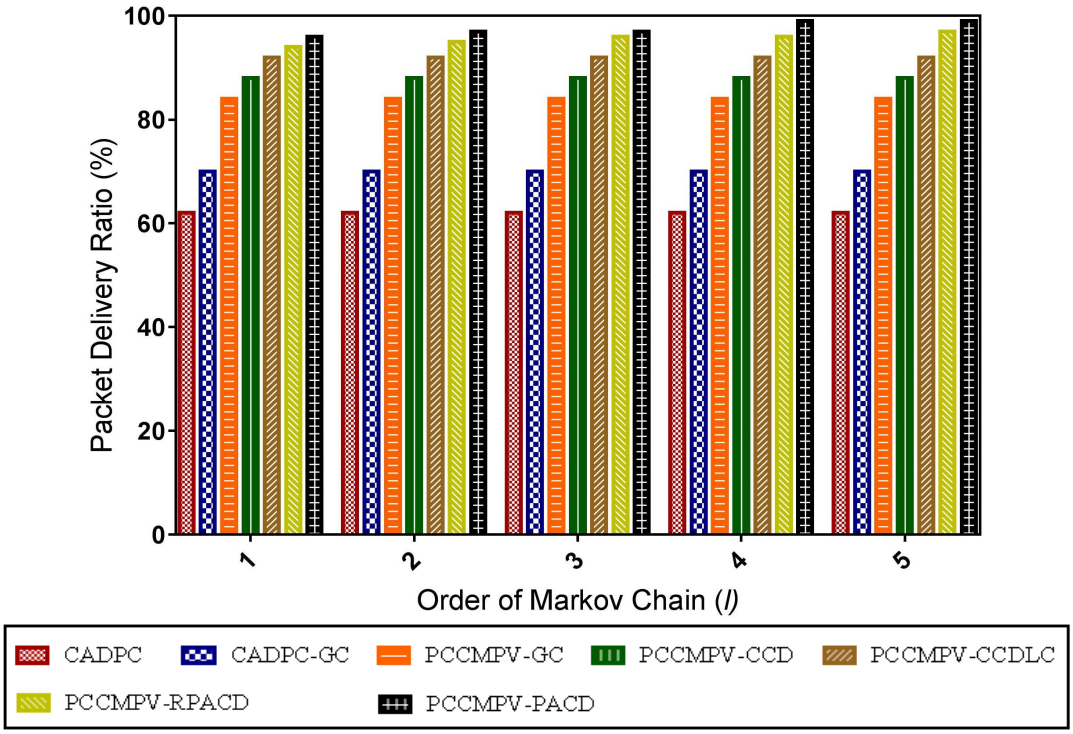


Figure 5.18: Packet delivery ratio over varying order of Markov chain (ℓ).

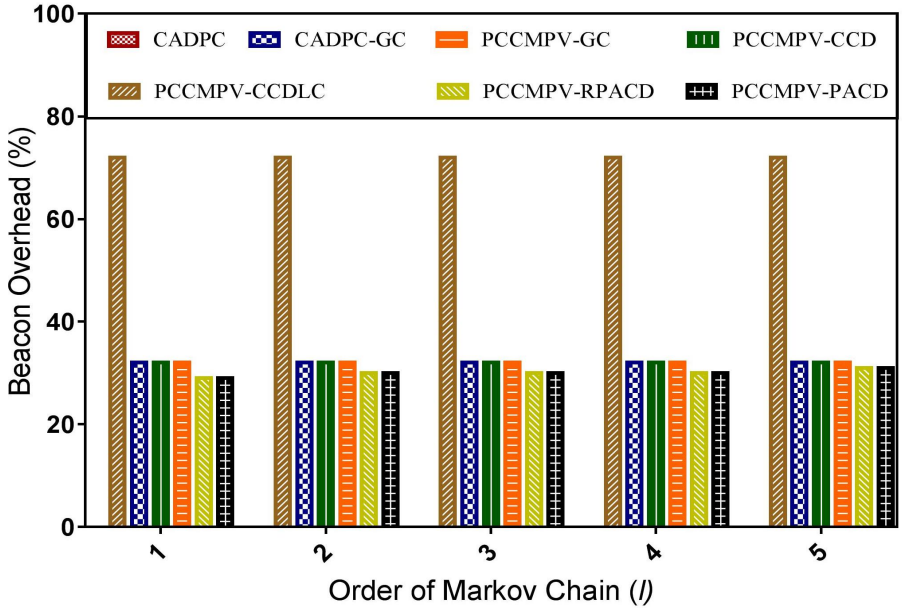


Figure 5.19: Beacon overhead over varying order of Markov chain (ℓ).

RPACD can be attributed to the higher prediction accuracy of the requester's location provided by the former. In addition, the low delay rendered by both PACD and RPACD indicates that more predictions are applied while the requester has only traversed along a few road segments beyond its location at the time of encounter, thus increasing the chance of having an accurate predicted location of the requester.

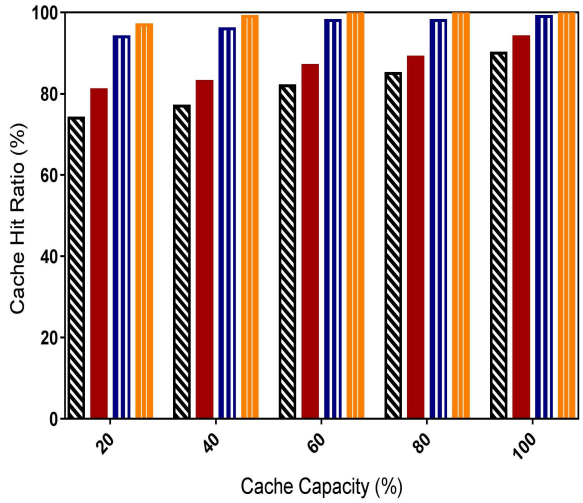
We study the effect of varying ℓ on the amount of overhead. As depicted in Figure 5.19, the amount of overhead remains the same in CADPC-GC, PCCMPV-GC, CCD, and CCDLC as ℓ increases, since none of these schemes involves the use of any prediction model. In contrast, the amount of overhead in PACD and RPACD slightly increases as ℓ increases. This is due to the increase in the number of previously traversed road segments that are exchanged between vehicles. Note that the size of such information is so small compared to the exchanged cached content information. Thus, increasing ℓ leads to a negligible increase in overhead, that can only reach up

to 3%. As previously mentioned, PACD and RPACD generate the same amount of overhead, since the use of bloom filters is incorporated in both of them.

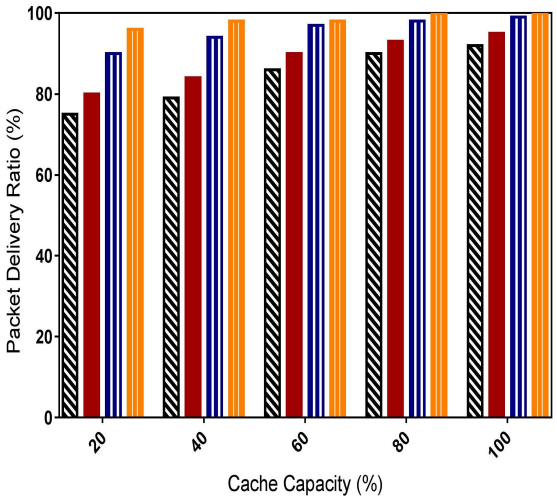
5- Reactive versus Proactive

In Chapter 3, we introduced VOPC-PSO, which acts as an optimal predictive proactive caching benchmark that can quantify the potential gains of predictive proactive caching schemes, evaluate their performance, and certify if there is a possibility for improvement. In the following discussion, we show the performance of our proposed reactive caching schemes compared to VOPC-PSO over varying cache capacity. This is since VOPC-PSO can also be used as a benchmark to evaluate the performance of reactive caching schemes. Thus, we use it to determine the gap between each of PCCMPV-CCD, PCCMPV-CCDLC, and PCCMPV-PACD on the one hand, and the optimal solution on the other. This can show the performance gains of using our proactive over reactive caching schemes, and can further assess the performance of the latter.

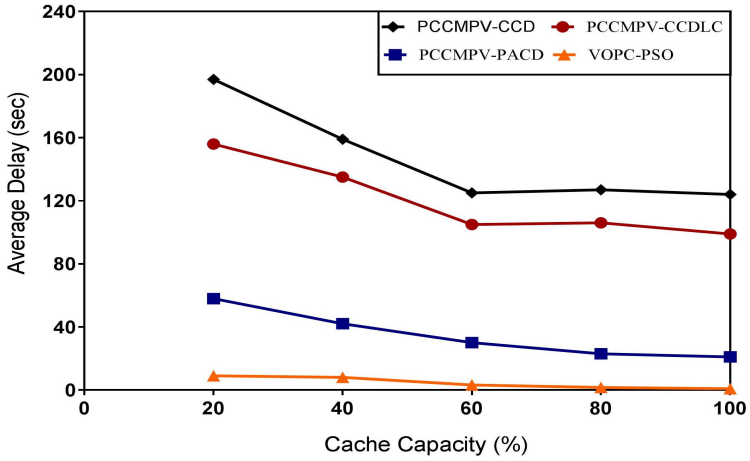
Figure 5.20(a), Figure 5.20(b), and Figure 5.20(c) depict the performance of the aforementioned schemes in terms of cache hit ratio, packet delivery ratio, and average delay, respectively. As shown in each of the three Figures, our proactive caching solution outperforms the reactive caching solution. In particular, as shown in Figure 5.20(a), the gap between each one of the schemes PCCMPV-CCD, PCCMPV-CCDLC, and PCCMPV-PACD on the one hand, and VOPC-PSO on the other, can reach up to 31%, 20%, and 6%, respectively in terms of cache hit ratio. Figure 5.20(b) shows that this gap can reach up to 28%, 22%, and 8%, respectively in terms of packet delivery ratio, whereas it can reach up to 105%, 99%, and 84%, respectively in terms of delay.



(a) Cache hit ratio.



(b) Packet delivery ratio.



(c) Average delay.

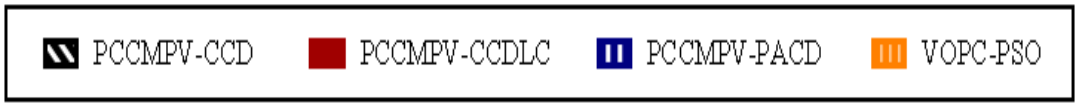


Figure 5.20: Performance results of PCCMPV-CCD, PCCMPV-CCDLC, PCCMPV-PACD, and VOPC-PSO over varying cache capacity.

Note that the superiority of the predictive proactive caching solution compared to the three reactive caching solutions is attributed to two reasons. The first reason is that the former can achieve better utilization of the available cache capacity than the latter. This is since cache placement decisions are made by one centralized entity, and thus it has a global knowledge of the cached content information of each node. The second reason is the fact that VOPC-PSO pre-caches the data at the parked vehicles that the requesters pass by. This reduces the risk of failure to find caching nodes, which increases cache hits. Also, VOPC-PSO strives to satisfy each request before a certain time frame specified by the corresponding user. Thus, it significantly reduces the delay. In contrast, the reactive caching solutions rely on cache discovery in order to find the caching nodes. In addition, they may favor a more distant caching node rather than a closer one if there is more reliability in its estimated position, which increases the delay compared to VOPC-PSO. This explains the significantly lower gap of PCCMPV-PACD compared to the gaps rendered by PCCMPV-CCD, and PCCMPV-CCDLC, since the cache discovery policy of PACD significantly outperforms that of CCD and CCDLC.

VOPC-PSO does not need to track the requester's location, since data is acquired as the requester passes by the caching node. This reduces the risk of dropping the data packet due to failure to locate the requester, which increases the packet delivery ratio. In contrast, each one of PCCMPV-CCD, PCCMPV-CCDLC, and PCCMPV-PACD adopts a specific procedure to track the requester, where the one adopted in PCCMPV-PACD relies on prediction. Thus, the gap between it and VOPC-PSO is lower than the other two. Note that in Figure 5.20(a), Figure 5.20(b), and Figure

5.20(c), the gap between the three reactive caching schemes and VOPC-PSO decreases as the cache capacity percentage increases. This is since the chance of finding the caching nodes increases, due to the same reasons previously mentioned in the discussion of the impact of cache capacity.

5.4 Summary

In this chapter, we proposed the Probabilistic Cooperative Caching at Moving and Parked Vehicles (PCCMPV) scheme. In PCCMPV, we exploit the static and mobile nature of parked and moving vehicles, respectively, to dynamically populate valuable road segments with diverse cached data. To do so, we dynamically assign a probability of caching to nodes along the data delivery path to assess their importance as caching nodes. For parked vehicles, such a probability relies primarily on the traffic density of the corresponding road segment, as well as its closeness centrality, and remoteness from the nearest data holder. PCCMPV provides an implicit form of off-path caching by assessing the trajectory of moving vehicles encountered along the data delivery path to calculate their probability of caching. Performance evaluation of PCCMPV demonstrates significant improvements in terms of delay, packet delivery ratio, and cache hit ratio compared to other caching schemes in vehicular networks.

We also evaluated the performance of CCD and PACD that were proposed in Chapter 4. To do that, we implemented them along with PCCMPV for cache placement and compared them to other cache discovery schemes. Performance evaluation shows that CCD significantly improves delay, packet delivery ratio, and cache hit ratio, compared to the server-based cache discovery approach typically used in VANETs, as well as to a neighborhood-restricted tracking-based cooperative cache discovery scheme that is commonly used in dynamic networks. This is done without

yielding any additional overhead than the latter. We also implemented CCD while adopting the exchange of LCs among neighboring vehicles (i.e., CCDLC). CCDLC has been shown to improve the performance even further. However, this is done at the expense of significant additional overhead. Performance evaluation has demonstrated the superiority of PACD compared to CCD and CCDLC in terms of delay, packet delivery ratio, and cache hit ratio, while significantly reducing beacon overhead compared to CCDLC, due to the use of bloom filters. The prediction technique used in PACD has also been shown to outperform a clustering-based trajectory prediction scheme in terms of prediction accuracy.

Chapter 6

Conclusion and Future Directions

6.1 Summary

In this thesis, we explored the use of VANETs as an edge caching platform for social networking. This was motivated by the need to mitigate the substantial traffic load at backhaul links in 5G networks. This traffic load can be triggered by social media traffic, which is the predominant source of Internet traffic that stems primarily from mobile devices. We proposed caching solutions that aimed at maximizing cache hits and improving the quality of Internet services in VANETs. Note that our work is not restricted to social media applications only. Rather, it can accommodate any application that involves non-real-time, highly popular, and noncritical contents.

Chapter 1 provided an overview of the research problem, as well as our objectives and contributions. Chapter 2 presented an overview of VANETs, and its basic characteristics. It also discussed existing proactive and reactive caching schemes in VANETs. In addition, since cooperative caching has rarely been investigated in VANETs, we reviewed existing cooperative caching schemes in MANETs and ICNs, including cooperative cache discovery and cache placement schemes. Based on this

review, we discussed which ones should be leveraged and which should be avoided within the context of VANETs.

Chapter 3 presented our predictive proactive caching solution that is based on the fact that some users tend to follow a daily routine. This leads to exhibiting a rather consistent and predictable behavior in terms of the route they follow, the time of taking that route, and their social media access behavior. Such a predictable behavior is utilized to pre-cache the data at parked vehicles to be proactively procured by requesters as they pass by. For this purpose, we presented two modules; a prediction module, and a proactive cache placement module. In the former, we proposed a long-term travel time prediction scheme that takes varying weather conditions into consideration. This scheme is based on an LSTM model that uses particle swarm optimization (PSO) in the training procedure rather than the gradient descent (GD) method. This is in order to expand the search space and avoid local minima entrapment. In order to demonstrate the effect of taking the weather into consideration in the prediction procedure, we implemented both GD-LSTM and GD-LSTM-NW, where the weather is considered in the former but not the latter. We compared the three models; LSTM-PSO, GD-LSTM, and GD-LSTM-NW. *We conclude from the results that LSTM-PSO is better than both GD-LSTM, and GD-LSTM-NW, and GD-LSTM is better than GD-LSTM-NW, in terms of prediction accuracy.*

In the proactive cache placement module, we introduced an optimal and a heuristic solution. The Vehicular Optimal Proactive Caching (VOPC) benchmark was introduced as the optimal solution in order to quantify the potential gains of predictive proactive caching in improving the quality of Internet services in vehicular networks. The caching problem in VOPC was formulated as an integer linear programming

(ILP) optimization problem, and can thus act as an upper bound on reachable potential. The objective is to maximize cache hits by assigning replicas to caching spots that yield maximum certainty in their spatiotemporal availability for requesters. This is while sustaining a cache capacity limit. We also proposed a greedy heuristic solution, called the Proactive Caching at Parked Vehicles (PCPV) scheme to solve the caching problem. We evaluated the performance of PCPV compared to the optimal solution VOPC, and implemented VOPC and PCPV using both PSO-LSTM and GD-LSTM. This is to demonstrate the effect of the prediction accuracy on the efficiency of the cache placement procedure. We also compared the proposed predictive proactive caching approach to the broadcast proactive caching (BPC) approach. *We conclude from the results that the predictive approach is better than BPC in terms of delay, packet delivery ratio, cache hit ratio, and satisfaction ratio under different scenarios. However, when the percentage of road segments that have parked vehicles for caching significantly decreases, BPC slightly outperforms PCPV-GD in terms of packet delivery ratio and cache hit ratio. In the same scenario, BPC also outperforms PCPV-PSO in terms of packet delivery ratio, while they both sustain almost the same cache hit ratio. PCPV approaches the optimal solution in most scenarios, but the gap starts to increase, suggesting a room for improvement, under restricted cache capacity, and when only small percentage of road segments have parked vehicles that can be used for caching. We also conclude that the prediction accuracy of the travel time can have a significant impact on the performance of the cache placement procedure. This leads to VOPC-PSO and PCPV-PSO being better candidates for proactive cache placement than VOPC-GD and PCPV-GD, respectively.*

In Chapter 4, we presented two cooperative cache discovery schemes; CCD and

PACD. CCD employs a tracking-based cache discovery scheme to dynamically navigate requests towards caching nodes that are within close proximity to the requester. It utilizes the last encounter information, as well as the static and mobile nature of parked and moving vehicles, respectively, to diffuse cached content information, and track caching nodes. CCD expands the search space beyond the neighborhood scope by relying on such diffusion. It also relies on the static nature of parked vehicles for tracking purposes. In particular, it enables parked vehicles to host the trails left by moving vehicles as they pass by. Such trails act as breadcrumbs that indicate the next road segment to which a moving vehicle is heading. Since the search space is still restricted by such trails in CCD, we further expanded the search space beyond such restrictions by proposing PACD.

PACD resorts to predicting vehicles' trajectory in order to determine the location of moving caching nodes wherever they are. PACD also exploits the static and mobile nature of parked and moving vehicles, respectively, to promulgate cached content information into the network. Such information are pertaining the cached contents of the vehicles themselves, as well as those of parking clusters that have been encountered. The latter is referred to as LCs. In order to reduce the associated overhead, PACD incorporates the use of bloom filters. It dynamically predicts the location of mobile caching nodes in order to locate replicas that are closer to the requester. The Any Relation Clustering Algorithm (ARCA) is employed to cluster trips based on their route similarity using the XXDice similarity coefficient. Each cluster is then trained using the MTD-Probit model to predict the remaining trajectory of vehicles based on partial knowledge of their ongoing trip. Using these predictions, PACD tracks all possible data providers and ranks them based on their proximity to the

requester, as well as their prediction entropy.

In Chapter 5, we presented the proposed cooperative cache placement scheme PCCMPV. PCCMPV expands the cooperation range beyond the neighborhood scope. It does so by exploiting the static and mobile nature of parked and moving vehicles, respectively, to acquire and distribute cached content information among nodes. This is done in the same way as in PACD. Based on the exchanged information, PCCMPV pours diverse data into valuable road segments. This is done by dynamically assigning a probability of caching to vehicles along the data delivery path to evaluate their importance as caching nodes. PCCMPV further exploits the trajectory of moving vehicles to induce a form of off-path caching at road segments.

We evaluated the performance of PCCMPV compared to a combination of an implicit vehicular cooperative caching scheme, called CADD, as well as a non-cooperative caching scheme, called DPC. We referred to this combination of schemes as CADPC. We implemented PCCMPV while using the neighborhood restricted tracking-based cache discovery scheme embedded in GroupCaching (GC). We referred to it as PCCMPV-GC. We also implemented CADPC while using the server-based cache discovery scheme that is typically used in VANETs, and while using GC. We referred to the latter as CADPC-GC. *We conclude from the results that PCCMPV is better than CADPC and CADPC-GC in terms of delay, packet delivery ratio, and cache hit ratio under different scenarios.*

Along with the performance of PCCMPV, we also evaluated the performance of the cache discovery schemes presented in Chapter 4. To do so, we implemented both CCD and PACD while using PCCMPV for cache placement. We also implemented CCD while incorporating the exchange of LCs that are used in PACD. This is in order

to evaluate the effect of such additional information. We referred to it as CCDLC. *We conclude from the results that CCD is better than GC in terms of delay, packet delivery ratio, and cache hit ratio. It manages to achieve such better results while maintaining the same beacon overhead yielded in GC. We further conclude that CCDLC is better than GC, and CCD, in terms of the aforementioned metrics. However, this comes at the expense of a much higher beacon overhead. Meanwhile, PACD performs better than GC, CCD, and CCDLC in all metrics, while significantly reducing beacon overhead compared to CCDLC. This is due to the use of bloom filters in PACD. PACD also manages to reduce the overhead compared to CCD as the cache capacity increases, while rendering higher overhead under low cache capacity. In addition, PACD yields a lower overhead than CCD when the percentage of road segments that have parked vehicles for caching is low.*

We evaluated the proposed trajectory prediction scheme, and assessed the effect of prediction accuracy on the performance of PACD. To do so, we implemented PACD using the proposed prediction model, as well as using the baseline clustering-based prediction scheme RMM. We referred to the latter as RPACD. *We conclude from the results that PACD is better than RPACD in terms of prediction accuracy, and that this superiority leads to significant improvements in terms of delay, packet delivery ratio, and cache hit ratio.*

6.2 Recommendations

The recommendations that stem from this work can be summarized as follows.

1. In order to recruit parked vehicles for the purpose of contributing to the caching process, free parking spaces can be provided as incentives.

2. Reactive caching can serve users who have a predictable behavior, as well as those who do not possess such a behavior. However, predictive proactive caching performs significantly better than reactive caching in serving the former type of users, whereas it is not suitable for the latter.

3. The use of bloom filters can cater for a significant reduction in the amount of beacon overhead associated with tracking-based cache discovery schemes. The number of hash functions, as well as the size of bloom filters should be selected such that the probability of false positive (given by Eq. 4.28 in Chapter 4) is low.

4. In predictive proactive caching, it is recommended to inform the users beforehand of whether or not their specified deadlines (i.e., QoS) can be achieved. This is particularly important when the available cache capacity is too low to accommodate high QoS for all users.

5. The use of a large number of hidden units in LSTMs should be avoided unless the level of improvement in prediction accuracy is worth the computation cost.

6.3 Future Directions

There are several future directions and open issues that can be explored to fully capture the potential of predictive proactive caching, as well as cooperative caching in VANETs. We highlight some of them below.

1. In our proactive caching schemes, we assumed that parked vehicles remain in their parking spaces during the entire time. However, there is some dynamic nature associated with parked vehicles as they enter and leave their parking

spaces. This adds another layer to the cache placement problem and the spatiotemporal availability of replicas, where the duration of availability of parked vehicles needs to be taken into consideration. In addition, there is a need for a migration technique to regulate the way the cached data can be sent from one parked vehicle to another when the time comes for it to leave.

2. In association with the previous point, the movement of parked vehicles opens another possibility for prediction as well. The duration of availability of parked vehicles could be more predictable for some parked vehicles than others. For example, parked vehicles of users at the theater, or at school or work, have a certain degree of stability, and thus predictability. Parked vehicles at Electric Vehicle Charging Stations (EVCSs) provide another important setting where there is a consistent participation pattern. This stems from the fact that electric vehicles in EVCSs also possess a sense of stability in the duration of their availability due to the predictable amount of time it takes to charge electric vehicles. This makes them extremely promising candidates for caching. This applies to both our proactive and cooperative caching solutions.

3. In our proactive and reactive caching schemes, we focused on selecting the best road segments for caching. The parked vehicle that has the largest available cache space at the selected road segment was then used to cache the designated content. Other selection criteria pertaining to the parked vehicles themselves can be explored in the future. For example, the duration of availability, as well as the migration cost can be considered in the selection process. Note that the latter is the communication overhead associated with transmitting the cached data from departing parked vehicles when migration occurs.

4. Our travel time prediction model that was incorporated into the predictive proactive caching framework was based on an LSTM neural network model. However, the prediction capability of such models rely on historical data only, without any regard to real-time information if it differs from the training data. In order to overcome this problem, we plan on using a hybrid model that uses the proposed PSO-LSTM model for offline estimations, and a filtering technique for dynamic adjustments. In this model, kalman filtering can be used for dynamic adjustments in cases when there is a huge difference between offline estimations and real-time information.

5. In our cooperative cache placement scheme, the probability of caching at moving vehicles needs to also involve the probability of encountering another moving vehicle that already has the data in its cache. This is in order to further increase data diversity.

6. A hybrid caching scheme that consists of both our predictive proactive caching, as well as the proposed cooperative caching schemes can be explored, where if the former fails to serve a request, the latter can be triggered.

7. In the future, the proposed caching framework can be extended to accommodate large-sized contents that require multiple transmissions.

8. We relied on a rather primitive scheme for classifying drivers behavior in order to provide a personalized travel time prediction. In the future, more sophisticated classification schemes can be explored.

Bibliography

[1] "The Global State of Digital in 2019 Report". [Online]. Available: https://hootsuite.com/pages/digital-in-2019 (Accessed 2020, August 3rd).

[2] "Number of Social Media Users Worldwide from 2010 to 2021 (in billions)". [Online]. Available: https://www.statista.com/statistics/278414/number-of-worldwide-social-network-users/ (Accessed 2020, August 3rd).

[3] "Mobile's Hierarchy of Needs Revealing Key Insights into Global Mobile Trends". [Online]. Available: https://www.comscore.com/Insights/Press-Releases/2017/3/comScore-Releases-New-Report-Mobiles-Hierarchy-of-Needs (Accessed 2020, August 3rd).

[4] G. S. Paschos, G. Iosifidis, M. Tao, D. Towsley, and G. Caire, "The Role of Caching in Future Communication Systems and Networks", *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1111–1125, June 2018.

[5] S. Ilarri , T. Delot , R. Trillo, "A Data Management Perspective on Vehicular Networks", *IEEE Communications Surveys and Tutorials*, vol. 17, no. 4, pp. 2420–1760, 2015.

[6] O. Attia and T. ElBatt, "On the Role of Vehicular Mobility in Cooperative Content Caching", *in 2012 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pp. 350–354, April 2012.

[7] J. Balen, G. Martinovic, K. Paridel, and Y. Berbers, "PVCM: Assisting Multihop Communication in Vehicular Networks Using Parked Vehicles," *International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT '12)*, pp. 119–122, 2012.

[8] N. Lu, N. Zhang, N. Cheng, X. Shen, J. W. Mark and F. Bai, "Vehicles Meet Infrastructure: Toward CapacityCost Tradeoffs for Vehicular Access Networks," *in IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1266-1277, doi: 10.1109/TITS.2013.2258153, Sept. 2013.

[9] G. Deng, L. Wang, F. Li, and R. Li, "Distributed Probabilistic Caching Strategy in VANETs through Named Data Networking", *in Proc. of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 314–319, 2016.

[10] J. Zhao, Y. Zhang, and G. Cao, "Data Pouring and Buffering on the Road: A New Data Dissemination Paradigm for Vehicular Ad Hoc Networks," in *IEEE Transaction on Vehicular Technology*, vol. 56, no. 6, pp. 3266–3277, 2007.

[11] H. Gong and L. Yu, "Content Downloading with the Assistance of Roadside Cars for Vehicular Ad Hoc Networks", *Mobile Information Systems*, vol. 2017, Article ID 4863167, 2017.

[12] S. Glass, I. Mahgoub, and M. Rathod, "Leveraging MANET-Based Cooperative Cache Discovery Techniques in VANETs: A Survey and Analysis", *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp.2640–2661, 2017.

[13] A. Ioannou, and S. Weber, "A Survey of Caching Policies and Forwarding Mechanisms in Information-Centric Networking", *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp.2847–2886, 2016.

[14] A. Broder and M. Mitzenmacher, "Network Applications of Bloom Filters: A Survey", *Internet Mathematics*, Vol. 1, No. 4, 2004.

[15] H. Hartenstein and K. P. Laberteaux, "A Tutorial Survey on Vehicular Ad Hoc Networks", *IEEE Communications Magazine*, vol. 46, no.6, pp.164–171, 2008.

[16] G. Karagiannis, O. Altintas, E. Ekici, G. Heijenk, B. Jarupan, K. Lin, and T. Weil, "Vehicular Networking: A Survey and Tutorial on Requirements, Architectures, Challenges, Standards, and Solutions", *IEEE Communications Surveys and Tutorials*, vol. 13, no. 4, pp. 584–616, 2011.

[17] N. Lu, N. Cheng, N. Zhang, X. Shen and J. W. Mark, "Connected Vehicles: Solutions and Challenges," *in IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 289-299, Aug. 2014, doi: 10.1109/JIOT.2014.2327587.

[18] S. Al-Sultan, M. M. Al-Doori, A. H. Al-Bayatti, and H. Zedan, "A Comprehensive Survey on Vehicular Ad Hoc Network", *Journal of Network and Computer Applications*, vol. 37, pp. 380–392, 2014.

[19] O. Rehmana, R. Qureshib, M. O-Khaouac, M. F. Niazid, "Analysis of mobility speed impact on end-to-end communication performance in VANETs", *Vehicular Communications*, vol. 26 (2020), June 2020.

[20] U. A. Mughal, J. Xiao, I. Ahmad, K. HiChang, "Cooperative resource management for C-V2I communications in a dense urban environment", *Vehicular Communications*, vol. 26, 2020.

[21] G. Badawy, J. Miic, T. Todd and Dongmei Zhao, "Performance modeling of safety message delivery in vehicular ad hoc networks," *IEEE 6th International Conference on Wireless and Mobile Computing, Networking and Communications*, Niagara Falls, ON, pp. 188-195, doi: 10.1109/WIMOB.2010.5644987, 2010.

[22] S. Biswas, J. Miic and V. Miic, "DDoS attack on WAVE-enabled VANET through synchronization," *IEEE Global Communications Conference (GLOBECOM)*, Anaheim, CA, 2012, pp. 1079-1084, doi: 10.1109/GLOCOM.2012.6503256, 2012.

[23] N. Abani, T. Braun, and M. Gerla, "Proactive Caching with Mobility Prediction under Uncertainty in Information-Centric Networks", *in Proceedings of the 4th ACM Conference on Information-Centric Networking*, pp. 88–97, 2017.

[24] B. Xu, A. Ouksel, and O. Wolfson, "Opportunistic Resource Exchange in Inter-vehicle Ad Hoc Networks", *in Proceedings IEEE International Conference MDM*, pp. 4-12, 2004.

[25] M. Li, Z. Yang, and W. Lou, "Codeon: Cooperative Popular Content Distribution for Vehicular Networks Using Symbol Level Network Coding," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 1, pp. 223–235, Jan. 2011.

[26] K. Liu, J. Kee-Yin Ng, J. Wang, V. C. S. Lee, W. Wu, and S. Hyuk Son, "Network-Coding-Assisted Data Dissemination via Cooperative Vehicle-to-Vehicle-Infrastructure Communications", *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, pp. 1–12, 2016.

[27] D. Grewe, M. Wagner, and H. Frey, "PeRCeIVE: Proactive caching in ICN-based VANETs", *in Proc. of the IEEE Vehicular Networking Conference (VNC)*, pp. 18, 2016.

[28] N. Liu, M. Liu, G. Chen, and J. Cao, "The Sharing at Roadside: Vehicular Content Distribution Using Parked Vehicles", in proceedings of the 31st IEEE Conference on Computer Communications (INFOCOM 2012), Mini-Conference, Orlando, FL, March 2012.

[29] Z. Su, Q. Xu, Y. Hui, M. Wen, and S. Guo, "A Game Theoretic Approach to Parked Vehicle Assisted Content Delivery in Vehicular Ad Hoc Networks", *IEEE Transactions on Vehicular Technology*, vol. 66, no. 7, July 2017.

[30] Y. AlNagar, S. Hosny, and A. A. El-Sherif, "Towards mobility-aware proactive caching for vehicular ad hoc networks", *in Proceedings of the IEEE Wireless Communications and Networking Conference Workshop (WCNCW)*, Marrakech, Morocco, 2019.

[31] Y. AlNagar, S. Hosny, and A. A. El-Sherif, "Proactive caching for vehicular ad hoc networks using the city model", *in Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2019.

[32] C. Bian, T. Zhao, X. Li, X. Du, and W. Yan, "Theoretical Analysis on Caching Effects in Urban Vehicular Ad Hoc Networks", *Wireless Communications and Mobile Computing*, DOI:10.1002/wcm.2651, 2015.

[33] N. Loulloudes, G. Pallis, and M. D. Dikaiakos, "Caching Dynamic Information in Vehicular Ad Hoc Networks", *ser. Euro-Par 2010-Parallel Processing, Springer,* pp. 516–527, 2010.

[34] L. C. Liu, D. Xie, S. Wang, and Z. Zhang, "CCN-based Cooperative Caching in VANET", *in Proceedings of the International Conference on Connected Vehicles and Expo (ICCVE)*, October 2015.

[35] S. Abdelhamid, H. S. Hassanein, G. Takahara, and H. Farahat, "Caching-Assisted Access for Vehicular Resources", *in the IEEE Conference on Local Computer Networks (LCN'14)*, pp. 28–36, Sept. 2014.

[36] S. A. Bitaghsir and A. Khonsari, "Cooperative Caching for Content Dissemination in Vehicular Networks", *Int. J. Commun. Syst.*, vol. 31, no. 8, 2018.

[37] L. Yin and G. Cao, "Supporting Cooperative Caching in Ad Hoc Networks", *IEEE Transactions on Mobile Computing.*, vol. 5, no. 1, pp. 77–89, Jan. 2006.

[38] E. Atsan and O. Ozkasap, "SCALAR: Scalable Data Lookup and Replication Protocol for Mobile Ad Hoc Networks", *Computer Networks*, vol. 57, no. 17, pp. 3654–3672, 2013.

[39] S. Umamaheswari and G. Radhamani, "Enhanced ANTSEC Framework with Cluster based Cooperative Caching in Mobile Ad Hoc Networks", *Journal of Communications and Networks*, vol. 17, no. 1, pp. 40–46, 2015.

[40] A. I. Saleh, "An Adaptive Cooperative Caching Strategy (ACCS) for Mobile Ad Hoc Networks", *Knowledge-Based Systems*, vol. 120, pp. 133–172, 2017.

[41] C. Jayapal, S. Jayavel, and V. P. Sumathi, "Enhanced Service Discovery Protocol for MANET by Effective Cache Management", *Wireless Personal Communications*, DOI: 10.1007/s11277-018-5866-3, 2018.

[42] S. Arianfar, P. Nikander, and J. Ott, "On Content-Centric Router Design and Implications", *in Proc. 2nd Workshop Re-Architect. Internet (ReARCH)*, Philadelphia, PA, USA, no. 5, 2010.

[43] N. Fotiou, D. Trossen, and G. C. Polyzos, "Illustrating a Publish Subscribe Internet Architecture", *Telecommunications System*, vol. 51, no. 4, pp. 233–245, 2012.

[44] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-Network Caching for Information-Centric Networks", *in Proceedings of the ACM Workshop on Information Centric Networking (ICN)*, Helsinki, Finland, pp. 55–60, 2012.

[45] A. Detti, N. B. Melazzi, S. Salsano, and M. Pomposini, "CONET: A Content Centric Internetworking Architecture", in Proceedings of the ACM Workshop on Information-Centric Networking (ICN), Toronto, ON, Canada, pp. 50–55, 2011.

[46] O. Ascigil , V. Sourlas , I. Psaras, and G. Pavlou , "A Native Content Discovery Mechanism for the Information-Centric Networks", *in Proceedings of the ACM Conference on Information-Centric Networking (ICN)*, pp. 145–155, 2017.

[47] R. Chiocchetti, D. Perino, G. Carofiglio, D. Rossi, and G. Rossini, "INFORM: A Dynamic Interest Forwarding Mechanism for Information Centric Networking", *in*

Proceedings of the ACM Conference on Information-Centric Networking (ICN), Hong Kong, pp. 9–14. 2013.

[48] C. J. C. H. Watkins and P. Dayan, "Technical note: Q-learning", *Machine Learning*, vol. 8, no. 4, pp. 279–292, May 1992.

[49] N. E. Majd, S. Misra, and R. Tourani, "Split-cache: A Holistic Caching Framework for Improved Network Performance in Wireless Ad Hoc Networks", *in Proceedings of the IEEE Global Communication Conference (IEEE GLOBECOM14)*, Austin, TX, USA, pp. 137142, Dec. 2014.

[50] M. Fiore, C. Casetti, and C.-F. Chiasserini, "Caching Strategies Based on Information Density Estimation in Wireless Ad Hoc Networks", *IEEE Transactions on Vehicular Technology*, vol. 60, no. 5, pp. 2194–2208, 2011.

[51] M. F. Caetano and J. L. Bordim, "A Cluster based Collaborative Cache Approach for MANETs", *in Proceedings of the International Conference in Network Computers (ICNC)*, Zrich, Switzerland, pp. 104–111, Nov. 2010.

[52] F. Angius, M. Gerla, and G. Pau, "BLOOGO: BLOOm filter based GOssip algorithm for wireless NDN", in Proceedings of the ACM Workshop on Emerging Name Oriented Mobile Networking Design: Architecture, Algorithms, Applications, Hilton Head Island, SC, USA, pp. 25–30, 2012.

[53] G. Rossini and D. Rossi, "Coupling Caching and Forwarding: Benefits, Analysis, and Implementation", *in Proceedings of the ACM Conference on Information-Centric Networking (ICN '14)*, New York, NY, USA, pp. 127–136, 2014.

[54] M. Badov, A. Seetharam, J. Kurose, V. Firoiu, and S. Nanda, "Congestion-Aware Caching and Search in Information-Centric Networks", *in Proceedings of the ACM Conference on Information-Centric Networking (ICN)*, Paris, France, pp. 37–46, Sep. 2014.

[55] M. Takaaki and H. Aida, "Cache data access system in ad hoc networks", *in Proceedings of the 57th IEEE Vehicular Technology Conference (VTC Spring)*, South Korea, pp. 1228–1232, Apr. 2003.

[56] T. Yeferny, S. Hamad and S. Belhaj, "CDP: a Content Discovery Protocol for Mobile P2P Systems", International Journal of Computer Science and Network Security (IJCSNS), vol.18, no.5, pp. 28–35, 2018.

[57] C. Yi et al., "A Case for Stateful Forwarding Plane", *Computer Communications*, vol. 36, no. 7, pp. 779–791, 2013.

[58] J. Cho, S. Oh, J. Kim, H. H. Lee, and J. Lee, "Neighbor caching in Multi-hop Wireless Ad Hoc Networks", IEEE Communications Letter, vol. 7, no. 11, pp. 525–527, 2003.

[59] S. N. Trambadiya, P. A. Ghosh, and J. N. Rathod, "Group Caching: A Novel Cooperative Caching Scheme for Mobile Ad Hoc Networks", Int. J. Eng. Res. Develop., vol. 6, no. 11, pp. 2330, 2013.

[60] D. K. Jain and S. Sharma, "Cooperative Caching Strategy in Mobile Ad Hoc Networks for Cache the Replaced Data Item", Wireless Personal Communication, vol. 84, no. 4, pp. 2613–2634, 2015.

[61] Y. Zhu, M. Chen, and A. Nakao, "CONIC: Content-Oriented Network with Indexed Caching", *in Proceedings of the 29th IEEE Conference on Computer Communications (INFOCOM)*, San Diego, CA, USA, pp. 1–6, 2010.

[62] M. Lee, K. Cho, K. Park, T. Kwon, and Y. Choi, "SCAN: Scalable Content Routing for Content-Aware Networking", *in Proceedings of the 53rd IEEE International Conference on Communications (ICC)*, Kyoto, Japan, pp. 1–5, 2011.

[63] J. M. Wang, J. Zhang, and B. Bensaou, "Intra-AS Cooperative Caching for Content-Centric Networks", *in Proceedings of the ACM SIGCOMM Workshop on Information Centric Networking (ICN)*, Hong Kong, pp. 61–66, Aug. 2013.

[64] "The NS-3 Network Simulator". [Online]. Available: https://www.nsnam.org/ (Accessed 2020, September 20).

[65] Gurobi, "Gurobi optimizer reference manual". [Online]. Available: http://www.gurobi.com (Accessed 2020, October 5).

[66] W. Fan, Z. Gurmu, "Dynamic Travel Time Prediction Models for Buses Using Only GPS Data", *International Journal of Transportation Science and Technology*, vol. 4, no. 4, pp. 353–366, 2015.

[67] C. Bai, Z. Peng, Q. Lu, and J. Sun, "Dynamic Bus Travel Time Prediction Models on Road with Multiple Bus Routes", *Computational Intelligence and Neuroscience*, vol. 2015, pp. 63–72, 2015.

[68] H. Xu, and J. Ying, "Bus arrival time prediction with real-time and historic data", *Cluster Computing*, vol. 20, no. 3, DOI: 10.1007/s10586-017-1006-1, 2017.

[69] J. Myung, D.-K. Kim, S.-Y. Kho, and C.-H. Park, "Travel time prediction using k nearest neighbor method with combined data from vehicle detector system and automatic toll collection system", *Transportation Research Record: Journal of the Transportation Research Board*, no. 2256, pp. 51.59, 2011.

[70] D. Sun, H. Luo, L. Fu, W.Liu, X. Liao, and M. Zhao, "Predicting Bus Arrival Time on the Basis of Global Positioning System Data", *Transportation Research Record*, Washington, DC; National Academy, 2007.

[71] J. Patnaik, S. Chein,and A. Bladihas, "Estimation of Bus Arrival Times Using APC Data", *Journal of Public Transportation*, vol. 7, no. 1, pp. 1-20, 2004.

[72] S. Chien, Y. Ding, and C. Wei, "Dynamic bus arrival time prediction with artificial neural networks", *Journal of Transportation Engineering*, vol. 128, no. 5, pp. 429-438, 2002.

[73] L. Chu, S. Oh, and W. Recker, "Adaptive Kalman filter based freeway travel time estimation", *in Proceedings of the 84th TRB Annual Meeting*, Washington, DC, USA, 2005.

[74] J. Rice and E. van Zwet, "A simple and effective method for predicting travel times on freeways", IEEE Transactions on Intelligent Transportation Systems, vol. 5, no. 3, pp. 200–207, 2004.

[75] D. Billings and J.-S. Yang, "Application of the ARIMA Models to Urban Roadway Travel Time Prediction: A Case Study", in Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC'06), vol. 3, pp. 2529–2534, 2006.

[76] S. I. J. Chien and C. M. Kuchipudi, "Dynamic Travel Time Prediction with Real-time and Historic Data", *Journal of Transportation Engineering-Asce*, vol. 129, no. 6, pp. 608–616, 2003.

[77] H. Dia, "An object-oriented neural network approach to short-term traffic forecasting", *European Journal of Operational Research*, vol. 131, no. 2, pp. 253–261, 2001.

[78] L. Vanajakshi, L. R. Rilett, "Support vector machine technique for the short term prediction of travel time," *in IEEE Intelligent Vehicles Symposium*, pp. 600–605, 2007.

[79] W. D. Fan, "Artificial Neural Network Travel Time Prediction Model for Buses Using Only Global Positioning System Data," *Journal of Public Transportation*, vol. 17, no. 2, pp. 4565, 2014.

[80] M. As, and T. Mine, "Dynamic Bus Travel Time Prediction Using an ANN-based Model", *in Proceedings of the 12th International Conference on Ubiquitous Information Management and Communication (IMCOM'18)*, New York, USA, 2018.

[81] Y. Duan, Y. Lv, F.Y. Wang, "Travel time prediction with LSTM neural network," *in IEEE 19th International Conference onIntelligent Transportation Systems (ITSC)*, pp. 1053–1058, 2016.

[82] I. Islek, and S. G. Oguducu, "Use of LSTM for Short-Term and Long-Term Travel Time Prediction", *CIKM Workshops*, 2018.

[83] A. Singh, S. Dixit, and L. Srivastava, "Particle Swarm Optimization-Artificial Neural Network For Power System Load Flow", *International Journal of Power System Operation & Energy Management*, vol. 1, no. 2, pp. 73–82, 2011.

[84] A. M . Ibrahim, and N. H. El-Amary, "Particle Swarm Optimization Trained Recurrent Neural Network for Voltage Instability Prediction", *Journal of Electrical Systems and Information Technology*, vol. 4, no. 1, DOI: 10.1016/j.jesit.2017.05.001, 2017.

[85] J. Kennedy, and RC. Eberhart, "Particle swarm optimization", *in Proceedings of the IEEE International Conference on Neural Networks*, Vol. 4, pp. 1942–1948, DOI: 10.1109/ICNN.1995.488968, 1995.

[86] "SUMO (Simulation of Urban Mobility)". [Online]. Available:http://sumo-sim.org/ (Accessed 2020, September 20).

[87] G. Potari, Z. Vincellera, Z. Acs, "A method for real-time adaptation of weather conditions within a traffic simulation", *Proceedings of the International Conference on Applied Informatics*, Eger, Hungary, vol. 2. pp. 41–47, DOI: 10.14794/ICAI.9.2014.2.41, 2014.

[88] J. Asamer, "Calibrating VISSIM To Adverse Weather Conditions", *2nd International Conference on Models and Technologies for Intelligent Transportation Systems*, pp. 22–24, 2011.

[89] X. Kong, F. Xia, A. Rahim, Y. Cai, Z. Gao, and J. Ma, "Mobility Dataset Generation for Vehicular Social Networks Based on Floating Car Data", *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, DOI: 10.1109/TVT.2017.2788441,

2018.

[90] S. Elsworth, and S. Gttel, "Time Series Forecasting Using LSTM Networks: A Symbolic Approach", *Manchester Institute for Mathematical Sciences*, The University of Manchester, UK, arXiv preprint:2003.05672, 2020.

[91] R. Atawia, H. Abou-zeid, H. Hassanein, and A. Noureldin, "Joint Chance Constrained Predictive Resource Allocation for Energy-Efficient Video Streaming", *IEEE Journal on Selected Areas in Communications*, vol.34, no. 5, pp. 1389–1404, 2016.

[92] S. Bayhan, L.Wang, J. Ott, J. Kangasharju, A. Sathiaseelan, and J. Crowcroft, "On Content Indexing for Off-Path Caching in Information Centric Networks", *in Proceedings of the ACM Conference on Information-Centric Networking (ICN)*, pp. 102–111, 2016.

[93] M. Lee, J. Song, K. Cho, S. Pack, J. Kangasharju, Y. Choi et al., "Content Discovery for Information-Centric Networking", *Computer Networks*, vol. 83, pp. 1–14, 2015.

[94] V. Sourlas, P. Flegkas, L. Gkatzikis, and L. Tassiulas, "Autonomic Cache Management in Information-Centric Networks", *in Proceedings of the IEEE/IFIP Network Operations and Management Symposium*, pp. 121–129, 2012.

[95] S. Guo, H. Xie and G. Shi, "Collaborative Forwarding and Caching in Content Centric Networks", *in Proceedings of the 11th IFIP Networking*, pp. 41–55, 2012.

[96] M. Fiore, C. Casetti, and C.-F. Chiasserini, "Caching Strategies Based on Information Density Estimation in Wireless Ad Hoc Networks", *IEEE Transactions on Vehicular Technology*, vol. 60, no. 5, pp. 2194–2208, 2011.

[97] N. E. Majd, S. Misra, and R. Tourani, "Split-cache: A Holistic Caching Framework for Improved Network Performance in Wireless Ad Hoc Networks", *in Proceedings of the IEEE Global Communication Conference (IEEE GLOBECOM'14)*, Austin, TX, USA, pp. 137–142, Dec. 2014.

[98] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-Network Caching for Information-Centric Networks", *in Proceedings of the ACM Workshop on Information Centric Networking (ICN)*, Helsinki, Finland, pp. 55–60, 2012.

[99] X. Chen, Q. Fan, and H. Yin, "Caching in Information-Centric Networking: From a Content Delivery Path Perspective", *in Proceedings of the International Conference on Innovations in Information Technology (IIT)*, pp. 48–53, 2013.

[100] Y. Du, S. K. S. Gupta, and G. Varsamopoulos, "Improving On-Demand Data Access Efficiency in MANETs with Cooperative Caching", Ad Hoc Networks, vol. 7, no. 3, pp. 579–598, May 2009.

[101] I. W. Ting and Y. K. Chang, "Improved Group-based Cooperative Caching Scheme for Mobile Ad Hoc Networks", J. Parallel Distrib. Comput., vol. 73, no. 5, pp. 595–607, 2013.

[102] Y.-W. Ting and Y.-K. Chang, "A Novel Cooperative Caching Scheme for Wireless Ad Hoc Networks: GroupCaching", *in Proc. Int. Conf. Netw. Archit. Storage (NAS)*, pp. 62–68, 2007.

[103] N. Chand , R. Joshi , M. Misra , "Cooperative Caching in Mobile Ad Hoc Networks Based on Data Utility", *Mobile Inf. Syst.*, vol. 3, no. 2, pp. 19–37, 2007.

[104] X. Zhu, J. Wang, L. Wang, and W. Qi, "Popularity-based Neighborhood Collaborative Caching for Information-Centric Networks", *in Proceedings of the IEEE 36th International Performance Computing and Communications Conference (IPCCC)*, San Diego, CA, USA, Dec. 2017.

[105] T. Mick, R. Tourani, and S. Misra, "MuNCC: Multi-hop neighborhood collaborative caching in information centric networks", *in Proceedings of the ACM Conference on Information-Centric Networking (ICN)*, pp. 93–101, 2016.

[106] M. Chen, X. Yu, and Y. Liu, "Mining Moving Patterns for Predicting Next Location", *Information Systems*, vol. 54, pp. 156–168, 2015.

[107] G. Xue, Z. Li, H. Zhu, Y. Liu, "Traffic-Known Urban Vehicular Route Prediction based on Partial Mobility Patterns", in:ICPADS, pp. 369–375, 2009.

[108] H. Jeung, Q. Liu, H. T. Shen, X. Zhou, "A Hybrid Prediction Model for Moving Objects", in:ICDE, pp.70–79, 2008.

[109] S. Gambs, M. O. Killijian, and M. N. del Prado Cortez, "Next Place Prediction Using Mobility Markov Chains", *in Proceedings of the First Workshop on Measurement, Privacy, and Mobility. ACM*, 2012.

[110] A. Monreale, F. Pinelli, R. Trasarti, F. Giannotti, "Where Next: A Location Predictor on Trajectory Pattern Mining", in:SIGKDD, pp.637–646, 2009.

[111] M. Morzy, "Mining Frequent Trajectories of Moving Objects for Location Prediction", in:MLDM, pp. 667–680, 2007.

[112] B. Kim, C. M. Kang, S. H. Lee, H. Chae, J. Kim, C. C. Chung, and J. W. Choi, "Probabilistic Vehicle Trajectory Prediction over Occupancy Grid Map via Recurrent Neural Network", arXiv preprint arXiv:1704.07049, 2017

[113] S. Gambs, M. O. Killijian, and M. N. del Prado Cortez, "Next Place Prediction Using Mobility Markov Chains", *in Proceedings of the First Workshop on Measurement, Privacy, and Mobility. ACM*, 2012.

[114] J. J.-C. Ying, W. C. Lee, and T. C. Weng, "Semantic Trajectory Mining for Location Prediction", *in Proceedings of the 19th International Conference on Advances in Geographic Information Systems*, pp. 34–43, NY, USA, 2011.

[115] P. Corsini, B. Lazzerini, F. Marcelloni, "A New Fuzzy Relational Clustering Algorithm based on the Fuzzy C-means Algorithm", *Soft Comput.*, vol. 9, no. 6, pp. 439–447, 2005.

[116] J. Nicolau, "A New Model for Multivariate Markov chains", *Scandinavian Journal of Statistics*, vol. 41, no. 4, pp. 1124–1135, 2014.

[117] P. Rathore, D.Kumar, S. Rajasegarar, M. Palaniswami, and J. C. Bezdek, "A Scalable Framework for Trajectory Prediction", IEEE Transactions on Intelligent Transportation Systems (early access), DOI: 10.1109/TITS.2019.2899179, March 2019.

[118] M. Chen, Y. Liu, and X. Yu, "Predicting Next Locations with Object Clustering and Trajectory Clustering", *in Advances in Knowledge Discovery and Data Mining*, Springer, pp. 344–356, 2015.

[119] G. Yuan, P. Sun, J. Zhao, D. Li, and C. Wang, "A Review of Moving Object Trajectory Clustering Algorithms, Artificial Intelligence Rev., vol. 47, pp. 123–144, 2017.

[120] A. Berchtold and A. Raftery, "The Mixture Transition Distribution Model for High-Order Markov Chains and Non-Gaussian Time Series", Statistical Science, vol. 7, pp. 328–356, 2002.

[121] G. Kondrak, "N-gram Similarity and Distance", *in Proceedings of the 12h International Conference on String Processing and Information Retrieval*, Buenos Aires, Argentina, pp. 115–126, 2005.

[122] B. Damsasio and J. Nicolau, "Combining a Regression Model with a Multivariate Markov Chain in a Forecasting Problem", *Statistics & Probability Letters*, vol. 90, pp. 108–113, 2014.

[123] F. Riedlinger and J. Nicolau, "The Profitability in the FTSE 100 Index: A New Markov Chain Approach", *Asia-Pacific Financial Markets*, DOI: 10.1007/s10690-019-09282-4, 2019.

[124] C. Brew and D. McKelvie, "Word-Pair Extraction for Lexicography", In Proc. of the 2nd Intl Conf. on New Methods in Language Processing, pages 45–55, 1996.

[125] RJ. Hathaway, JW. Davenport, and JC. Bezdek, "Relational Dual of the C-Means Clustering Algorithms", *Pattern Recognition*, vol. 22, no. 2, pp. 205–212, 1989.

[126] M. A. Khalilia, J. Bezdek , M. Popescu, J. M. Keller, "Improvements to the Relational Fuzzy C-Means Clustering Algorithm", *Pattern Recognition*, vol. 47, pp. 3920–3930, 2014.

[127] A. E. Raftery and S.Tavare, "Estimation and Modelling Repeated Patterns in High Oder Markov Chains with the Mixture Transition Distribution Model, *Appl. Statist*, 43, pp. 179–199, 1994.

[128] A. Berchtold, "Estimation in the Mixture Transition Distribution Model", *J. Time Ser. Anal.*, vol. 22, pp. 379–397, 2001.

[129] "GAUSS" [Online].Available:https://www.aptech.com/ (Accessed 2020, August 12).

[130] S. Arshad, M. A. Azam, M. H. Rehmani, and J. Loo, "Recent Advances in Information-Centric Networking based Internet of Things (ICN-IoT)", *IEEE Internet of Things Journal*, vol. 14, no. 8, DOI: 10.1109/JIOT.2018.2873343, 2018.

[131] S. Tarkoma, C. E. Rothenberg, and E. Lagerspetz, "Theory and Practice of Bloom Filters for Distributed Systems", IEEE Communications Surveys and Tutorials, vol. 14, no. 1, pp. 131–155, 2012.

[132] R. G. Gallager, "Information Theory and Reliable Communication", *New York, Wiley*, 1968.

[133] S.H. Bouk, et al., "Hybrid Adaptive Beaconing in Vehicular Ad Hoc Networks: A Survey", *International Journal of Distributed Sensor Networks*, 390360 (2015), 2015.

[134] H. J. F. Qiu, I. W.-H. Ho, C. K. Tse, and Y. Xie, "A Methodolgy for Studying 802. 11p VANET Broadcasting Performance with Practical Vehicle Distribution", *IEEE Transactions on Vehicular Technology*, vol. 64, no. 10, pp. 4756–4769, 2015.

[135] L. Breslau, P. Cao, L. Fan, G. Phillips, S. Shenker, et al., "Web Caching and Zipf-like Distributions: Evidence and Implications", *In IEEE INFOCOM*, pages 126–134, 1999.

Appendix A

Mobile Ad Hoc Networks (MANETs)

Mobile Ad hoc Networks (MANETs) are composed of mobile devices (frequently referred to as mobile hosts-MHs) that form a wireless network and can communicate with each other without the help of any network infrastructure such as access points or base stations [12]. Any MH can directly communicate with its one-hop neighbors that are within the transmission range of its broadcast channel. MHs can also cooperate with each other to transmit/receive data using multihop wireless links to communicate with other nodes that are not within their transmission range. MANETs are characterized by their dynamic topology. This dynamic topology results from the mobility of nodes, or their failure, which is typically caused by energy drainage [51]. Compared to VANETs, the dynamic topology of MANETs tends to be less frequent.

MANETs are characterized by their limited resources in terms of storage, energy, and computing capability [12, 51, 100]. This is in contrast to VANETs, which tend to have a much larger pool of resources to leverage [12]. In addition, MANETs are characterized by their constrained bandwidth. However, the problem of bandwidth limitation is less significant in MANETs than in VANETs, since the nodes that are

within each others communication range tend to be much fewer in MANETs. Furthermore, the mobility patterns of mobile nodes in MANETs are less constrained than those in VANETs, and are thus harder to predict [12]. The aforementioned characteristics trigger several challenges pertaining to the efficiency of data access in MANETs, particularly in terms of data availability and energy efficiency [12]. For instance, when MHs request data from a far-away server, several multi-hop data transmissions are required in order for the request to be satisfied. This involves large delay, energy consumption, and bandwidth usage [51, 100].

Appendix B

Information-Centric Networks (ICNs)

Originally, the Internet has been designed by adopting an end-to-end communication paradigm for the exchange of information between two endpoints; a client and a server. However, the ever-increasing Internet usage and the fact that this usage is mainly governed by content distribution and retrieval rather than connection to a particular server, have rendered such a paradigm inadequate [13]. The mismatch of conventional protocols and current Internet usage patterns have led to some difficulties pertaining to availability, mobility, multi-homing, scalability, and performance [13].

Information-Centric Networks (ICNs) have emerged as a promising alternative to shift the current Internet architecture towards a content-based communication paradigm [13]. To do so, ICNs focus on content dissemination and retrieval. They define a common protocol-corresponding to the network layerthat can be adopted by various applications while utilizing the processing power and storage resources within the infrastructure for content replication [13]. For this purpose, ICN routers are supplied with cache memory and provided with a caching capability.

ICN architectures are based on the use of the publish/subscribe model. In such a model, content providers announce their contents by publishing them to a Content

Notification Service (CNS) and consumers request various contents by subscribing to a CNS [13, 23, 44]. A CNS may be comprised of a name resolution service and/or a name-based routing service [13]. In order to achieve the collective set of future Internet requirements, ICNs employ the following core components [13, 23, 44]:

1) Content Naming: ICNs adopt the concept of content naming. To do that, ICNs use Named Data Objects (NDO) [13]. Each stored and accessed object, such as documents, movies, images, web pages, must be assigned a global unique name. The NDO is completely independent of the content location, storage, and communication method.

2) Routing: In ICNs, routing is handled using one of two approaches; name resolution and namedbased routing. In the former, content routing consists of two steps. First, requests or subscription messages are sent from the consumers to the name resolution system (NRS), indicating the name of the requested content. The NRS resolves the requested content by determining an individual or a set of addresses of content providers or caching nodes that have the requested data. Second, the consumers send request messages to the senders and the requested content is issued back to them. In named-based routing, in just one step, consumers send request packets directly to the original content provider or any caching node, which can resolve the request based on the content name.

3) In-Network Caching: In ICNs, any network node can perform content caching. Upon receiving a content request, a network node can directly respond if the content is available in its local cache. Otherwise, it can direct the request to its peers or to the original content provider. As the data propagates back to the requester, the content can be selectively cached. ICN caching is referred to as universal caching since it has

three characteristics; First, uniform caching is provided for all contents carried by any protocol. Second, any content can be cached regardless of its provider. Third, ICN caching facilitates pervasive caching, since it is enabled by all ICN nodes rather than being restricted to a few specialized caching nodes [13, 23, 44].