Cloud-based Strategies for Robust Connectivity and Efficient Transmission in Vehicular Networks

Ke Xu
Clemson University, kxu1985@gmail.com

Follow this and additional works at: https://tigerprints.clemson.edu/all_dissertations

Part of the Computer Engineering Commons, and the Electrical and Computer Engineering Commons

Recommended Citation
https://tigerprints.clemson.edu/all_dissertations/1507

This Dissertation is brought to you for free and open access by the Dissertations at TigerPrints. It has been accepted for inclusion in All Dissertations by an authorized administrator of TigerPrints. For more information, please contact kokeefe@clemson.edu.
CLOUD-BASED STRATEGIES FOR ROBUST CONNECTIVITY AND EFFICIENT TRANSMISSION IN VEHICULAR NETWORKS

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Electrical Engineering

by
Ke Xu
May 2015

Accepted by:
Dr. Kuang-Ching Wang, Committee Chair
Dr. Harlan Russell
Dr. Daniel Noneaker
Dr. Brian Fralix
Dr. Jim Martin
Abstract

Leveraging multiple wireless technologies and radio access networks, vehicles on the move have the potential to get ubiquitous broadband Internet connectivity. Many studies have put lots of efforts on vehicle-to-vehicle networks for relaying strategies, popular content distribution, etc. However, in dominant infrastructure-based vehicular networks, supporting continuous and fast data transfer for today’s prevalent services, e.g., video streaming, for vehicles anytime and anywhere is still a difficult research problem. By looking into such problem, impacts such as intermittent connectivity, dynamic network topology, fluctuating signal coverage, and inefficient transmissions, all result from two root causes in vehicular network—mobility and limited infrastructure resources. This dissertation investigates three core functions of infrastructure-based vehicular networks in the presence of the two causes: 1) network selection; 2) data forwarding; 3) data retrieval. By leveraging a compute cloud’s abundant computing and data storage resources, three cloud-based strategies are proposed to achieve robust connectivity and efficient transmission for vehicular networks.

First, network selection is essential to maintain robust connectivity for vehicles on the move. Serving a large number of vehicles, today’s centralized and distributed network selection solutions require sophisticatedly designed utility functions and optimization with complex computing, reducing flexibility and efficiency. A fast, game-based network selection scheme is proposed in this dissertation. Vehicles can select best access networks through a coalition formation game approach by leveraging wider scope network information for decision making. A one-iteration fast convergence algorithm is proposed to achieve the final state of coalition structure in the game. Through extensive simulation, the proposed network selection scheme was shown to balance system throughput and fairness without the need of an explicit fairness metric in the utility function. The algorithm efficiency showed eight-fold enhancement over a conventional coalition formation algorithm.

Second, efficient data forwarding strategies are important for transferring data to vehicles.
Previous studies have developed numerous methods to disseminate common content among vehicles using broadcasting or multicasting, however, these methods cannot automatically be applied to multiple unicasts to vehicles on the road. An inter-session network coding scheme for Internet-to-vehicles unicasts is proposed. The proposed scheme makes efficient utilization of network coding by applying the proposed dynamic, optimal grouping algorithm. Leveraging cloud assistance, the grouping algorithm can compute optimal flow routing using real-time network topology information. Extensive simulation showed that the proposed scheme achieved less delay by up to 6+ times than conventional routing for UDP traffic and zero end-to-end packet loss rate. Within a general range in terms of number of vehicles (10 ~ 40) in a $2 \times 2 \text{km}$ area and average speed (10 ~ 50 mph), the proposed scheme maintained an apparent advantage over conventional routing.

Third, reliable and efficient content retrieval on demand from vehicles on the move becomes more and more important. Today’s transport mechanism in vehicular networks still inherits traditional TCP/IP’s client-server manner, working with opportunistic scheduling. However, opportunistic scheduling cannot fundamentally get rid of the vulnerability of client-server architecture using TCP/IP in the presence of fast speed. Content-centric vehicular networking (CCVN) is proposed as a potential framework to achieve efficient data retrieval in vehicular networks instead of patching over TCP/IP. The connection-less, in-network caching and distributed characteristics of CCVN are able to adapt to vehicular environments easily. CCVN uses a cloud-based face management mechanism for enhancement. Emulation results showed CCVN has promising results compared to three selected opportunistic scheduling schemes using TCP/IP- MV-MAX, load-reducing scheduling, and pre-caching scheduling.

The promising performance of the proposed cloud-based strategies validates the usefulness and importance of the cloud-based system. This dissertation is expected to provide guidance for implementing cloud-based applications in vehicular networks in the real world, such as telematics and location-based services.
Dedication

I dedicate this doctoral dissertation to my wife and parents.
Acknowledgments

I would like to thank my advisor Dr. Kuang-Ching Wang. Dr. Wang taught me what research is and how to do research. More importantly, Dr. Wang showed me how to apply technologies in the real world. Without Dr. Wang’s guidance and help, it is impossible to complete this dissertation.

I would like to thank my wife Wei Wang, who lets me know what a true life is. I would like to thank my parents as well. Their selfless love and support keep me going until today.

I also would like to thank my committee- Dr. Harlan Russell, Dr. Daniel Noneaker, Dr. Brian Fralix and Dr. Jim Martin. Thanks for their time and insightful comments to help improve this dissertation.

Last, I want to thank my friends- Rajesh Narayanan, Aaron Rosen, Ryan Izard, Qing Wang, Mengzhe Gu, Rahul Amin, and Adam Hodges. They are those who shared great discussions, shared different cultures with me, and those who offered generous help when I had difficulties.
# Table of Contents

Title Page ................................................................. i
Abstract ............................................................... ii
Dedication .............................................................. iv
Acknowledgments ....................................................... v
List of Tables .......................................................... viii
List of Figures .......................................................... ix

1 Introduction .......................................................... 1
   1.1 Introduction to Vehicular Networks ....................... 1
   1.2 Research Motivations ...................................... 2
   1.3 Research Objectives ....................................... 4
   1.4 Dissertation Outline .................................... 7

2 Background and Related Work ................................... 8
   2.1 Network Selection ......................................... 8
   2.2 Data Forwarding ......................................... 10
   2.3 Data Retrieval .......................................... 12
   2.4 Cloud-based Systems for Vehicular Networks .......... 14

3 The Proposed Cloud-based System ................................ 16
   3.1 System Overview ......................................... 16
   3.2 Distributed Database .................................... 18
   3.3 Key Assumptions ......................................... 19
   3.4 An Envisioned Architecture ............................. 20

4 Coalition Formation Game based Network Selection .......... 23
   4.1 Network Model and Utility Functions .................. 24
   4.2 Game Formulation in Vehicular Networks ............... 28
   4.3 A One-Iteration Network Selection Algorithm ........ 30
   4.4 Results and Analysis ................................... 32

5 Network Coding based Data Forwarding .......................... 40
   5.1 Network Model .......................................... 41
   5.2 Assumptions ............................................. 45
   5.3 Problem Formulation .................................... 46
   5.4 Proposed Dynamic Grouping Algorithm for Inter-Session Network Coding ........ 47
   5.5 Results and Analysis ................................... 52
6 A Content-based Data Retrieval Framework for Vehicular Networks ........ 64
   6.1 Content-Centric Vehicular Networking ..................................... 64
   6.2 CCVN Enhancement for Data Retrieval ...................................... 72
   6.3 Results and Analysis ............................................................... 75

7 Conclusions and Future Work ....................................................... 84

Appendices ....................................................................................... 87
   A Proof of Theorem 4.1 ................................................................. 88
   B Proof of Theorem 4.2 ................................................................. 89
   C Proof of Lemma 5.1 ................................................................. 90
   D Proof of Theorem 5.1 ................................................................. 91

Bibliography ...................................................................................... 92
List of Tables

4.1 The proposed network selection algorithm. ............................................. 31
4.2 The proposed one-iteration coalition formation algorithm. ....................... 31
4.3 The locations and coverage ranges of BSs/APs. ........................................ 33

5.1 The proposed inter-session network coding algorithm. .............................. 48
5.2 The proposed dynamic grouping algorithm for network coding. .................... 49
5.3 Packet loss rate vs. number of vehicles. .................................................. 54
5.4 Packet duplication rate vs. number of vehicles. ....................................... 54
List of Figures

3.1 Overview of the proposed cloud-based system........................................ 17
3.2 Information in the database................................................................. 18
3.3 An envisioned architecture for vehicular networks................................. 21
4.1 RAN coverage of the simulation scenario............................................... 33
4.2 Vehicular scenarios................................................................................ 34
4.3 Comparison with centralized optimization algorithm in [10]. (opt-spec and opt-fair
represent optimization on spectral efficiency and fairness respectively) .......... 36
4.4 Throughput, delay and fairness v.s. arrival bit rate per vehicle.................... 37
4.5 Throughput, delay and fairness v.s. average velocity.................................. 37
4.6 Average running time for one scheduling interval of proposed coalition formation
algorithm and conventional algorithm......................................................... 39
5.1 Intra-session network coding.................................................................. 41
5.2 Inter-session network coding................................................................... 41
5.3 An example of inter-session network coding............................................ 42
5.4 A directed hypergraph $G = (N, H)$ for vehicular networks at time $t.$........ 44
5.5 Technical framework of applying the proposed algorithm in practice............ 51
5.6 Average throughput and average delay for UDP packets........................... 53
5.7 Average throughput and average delay for TCP packets............................ 54
5.8 Average delay vs. average speed (the number of vehicles is 30).................... 56
5.9 Packet loss rate vs. average speed (the number of vehicles is 30)................. 57
5.10 Average delay vs. ratio of forwarding links (the number of vehicles is 30)..... 58
5.11 Packet loss rate vs. ratio of forwarding links (the number of vehicles is 30).... 59
5.12 Average throughput with different network selection algorithms (the number of vehi-
cles is 30). Note: util-1: users’ bandwidth preference (Wi-Fi>LTE>WiMAX>HSPA>EVDO);
util-2: available bandwidth; util-3: $\sqrt{\text{throughput}/\text{delay}}$; util-4: $0.5 \times \text{throughput} +$ 
$0.5 \times \text{delay}$................................................................. 60
5.13 Average delay with different network selection algorithms........................ 61
5.14 Packet loss rate with different network selection algorithms.................... 62
5.15 Processing time of grouping................................................................. 63
6.1 Naming and packet types....................................................................... 65
6.2 CCVN node architecture........................................................................ 66
6.3 Three approaches of layer design for CCVN.......................................... 67
6.4 From connection to non-connection....................................................... 69
6.5 From on-demand caching to in-network caching...................................... 70
6.6 From centralized control to distributed control....................................... 71
6.7 Basic Operations of CCVN................................................................. 73
6.8 Emulation scenarios............................................................................... 75
6.9 Average throughput per vehicle compared to MV-MAX......................... 77
6.10 Fairness compared to MV-MAX. ................................................................. 78
6.11 Download time of a 32 MB file compared to MV-MAX. ............................ 79
6.12 Load compared to LR scheduler with different thresholds. ....................... 79
6.13 Ratio of goodput to load compared to LR scheduler with different thresholds. 80
6.14 Ratio of goodput to load compared to LR scheduler vs. packet loss probability. 81
6.15 Accumulated download size compared to ideal pre-caching. ....................... 82
Chapter 1

Introduction

1.1 Introduction to Vehicular Networks

Vehicular networks are gaining attention as the industries announce plans to bring ubiquitous broadband Internet connectivity to automobiles. Envisioned applications include road safety, driver assistance, infotainment, and vehicle telematics, utilizing a range of wireless communication methods such as Wireless Fidelity (Wi-Fi), Worldwide Interoperability for Microwave Access (WiMAX), Dedicated Short-range Communications (DSRC), or third Generation/fourth Generation (3G/4G) radios. Depending on applications, such networks can be realized as an ad-hoc network, an infrastructure-based network, or a hybrid combination of the two.

Numerous studies have put lots of efforts on vehicle-to-vehicle networks for relaying strategies, popular content distribution, etc. However, in recent dominant infrastructure-based vehicular networks, supporting continuous and fast data transfer for today’s prevalent services, e.g. video streaming, for vehicles anytime and anywhere is still a difficult research problem. Despite significant attention from academia and industry, vehicular networks still face a number of challenging difficulties. Nowadays, people still experience frequent network disconnection and sudden performance degradation when retrieving data on the move, in their private cars or public transit vehicles, despite continuous advancement of the mobile devices, e.g. smartphones integrating both Wi-Fi and 4G interfaces. These challenges are rooted in the way mobile devices address the two root causes in vehicular networks: fast speed and network resource limitations.

Mobility is naturally the most important challenge for vehicular networks. Due to mobility,
vehicles experience extremely dynamic communication channel conditions on the road caused by obstacles such as buildings, trees, and street layout, resulting in unstable received signal strength (RSS). Such influence directly causes intermittent connectivity on the road. Another critical influence by mobility is dynamic network topology. In today’s cellular networks, mobile devices execute handoff automatically across base stations of the same network operator. With multiple radio access technologies available today, devices have more options for network selection, potentially based on their own preferences, such as choosing free roadside Wi-Fi instead of 3G/4G data usage, or choosing one cellular provider over another. Lacking network information, especially when traveling through different regions, users are unable to make best network selection decisions. For network infrastructure, the number of users on the road they see and serve is also changing quickly.

The other critical factor is resource limitations in vehicular networks. A fundamental requirement is signal coverage. Infrastructure-based vehicular network are widely considered due to their high reliability and constant availability where such infrastructure exists. While a wide range of wireless access networks are being deployed rapidly, deploying and maintaining infrastructure remains a time and capital-demanding task. Today, 3G/4G cellular provides good coverage in urban areas despite the mentioned building-caused degradations; beyond urban areas, such coverage degrades quickly. Many have considered the use of vehicle-to-vehicle (V2V) communications to enhance network connectivity; nevertheless, with both transmitting and receiving vehicles moving, transmissions in such networks are even more unreliable. As a result, V2V communications today are often considered usable for only broadcast safety messages and one-to-many gossip-style data dissemination.

### 1.2 Research Motivations

Fast speed and resource limitations result in unsatisfied user experiences and inefficient network resource allocation and performance. The critical limitations reside in three core functions of vehicular networks:

- **Network selection**: Network selection schemes have been a topic receiving significant attention for the objectives of achieving improved vehicle connectivity performance, cost efficiency and overall network efficiency. Numerous studies have proposed different schemes for vertical handoff [82, 75, 14, 46, 70, 72, 76, 39, 10, 37, 54, 65, 64, 68, 66, 17, 89]. The schemes fall into
two typical categories: utility-based optimization and game theoretic frameworks. In [10], an example of the former category, a centralized multi-attribute optimization problem is proposed based on the weighted sum of spectral efficiency, fairness, and battery lifetime utilities. In [68], an example of the latter category, a coalition formation game is proposed to formulate cooperation among roadside units (RSUs) to coordinate the prioritized classes of data transmitted to vehicles. Both forms of solutions require non-trivial computing complexity. With the scarce memory and processing capacity on typical RSUs or a custom provisioned server, practical implementation of such solutions must trade off scope and granularity of network information for algorithmic efficiency.

- **Data forwarding**: It has been shown that forwarding of non-coded data packets over the network suffers from major throughput degradation in high speed networks with even a low packet drop rate [97]. Works on network coding have demonstrated significantly more stable throughput and hence reduced delay in data download over V2V networks (e.g., VANETCODE [4], CodeOn [41], CodePlay [94]) compared with peer-to-peer (P2P) based methods [96, 85]. However, the fundamental problem of large delay still exists in the recent best network coding solution in V2V networks, e.g. the average downloading delay for a 16 MB file is 424 s in 100-vehicle scenario with packet-level network coding [41]. V2V-based network coding solutions have limited performance on reliable data forwarding. To overcome this limitation, more infrastructure resources must be involved [94] and more efforts on vehicle-to-infrastructure networks should be focused on [25]. More importantly, V2V-based network coding is mostly for common content dissemination but cannot directly be applied to today’s unicasts.

- **Data retrieval**: In rapidly varying channel conditions under TCP/IP architecture, opportunistic scheduling is more suitable for data retrieval [5, 88, 73, 29, 99, 36]. In [26], MV-MAX was proposed as a medium access protocol that opportunistically grants wireless access to vehicles with the maximum transmission rate, improving overall system throughput by a factor of almost 4 with respect to the IEEE 802.11 protocol. Load-reducing scheduling was proposed to transmit packets only with high link reliability [90], therefore reducing burst retransmissions caused by packet losses in bad coverage areas. A centralized controller uses an access point coverage map to schedule pre-caching on possible RSUs along the predicted vehicle’s travel path is another mechanism to avoid disconnectivity and reduce download time during data
delivery [2]. However, these methods do not rid the vulnerability of client-server architecture using TCP/IP in the presence of fast speed, e.g. any unexpected signal disturbance may result in connection breaks though scheduler allows packets to be transmitted. The complexity of such algorithms is often constrained by scarce memory and processing capacity on base stations [13].

The common problem across the three functions above is acquisition of more information and, at the same time, the inability to process more information. Recently, cloud computing is becoming increasingly viable for providing any-time, any-place data processing and storage resources. Furthermore, application vendors (e.g., Apple, Google, Microsoft) and network providers (e.g., AT&T, Verizon) are promoting mobile cloud computing services. Researchers have proposed a number of mobile cloud computing platforms. MobiCloud was proposed as a geo-distributed platform, allowing users to connect to cloud resources that are geographically closer to them, reducing communication delay [28]. VehiCloud was proposed as a service-oriented cloud architecture to provide routing service in vehicular networks, by predicting vehicles’ future locations in the cloud using way point messages [61]. Real-road experiments in a building-embraced residential area demonstrated its feasibility and showed a performance more reliable than traditional routing protocol by 80% ~ 100%. Mobile cloud computing technologies as an emerging trend opens up possibilities for achieving more complex computing needs in the cloud.

1.3 Research Objectives

This dissertation analyzes the nature of two root causes of unstable and degraded performance in vehicular networks and explores a novel solution methodology based on the emerging cloud computing technologies. The objective of this dissertation is to achieve robust connectivity and efficient transmission in vehicular networks, by designing cloud-based strategies considering both user-centric and network-centric metrics, expecting higher utilities for both vehicles and networks at the same time. With respect to the three aspects in Sec 1.2, this dissertation proposes three cloud-based strategies to solve these problems, which are described as below.

1. **Coalition formation game based network selection**: usually, selfish individuals always want to get as much bandwidth as possible, or connect to the network with the best signal strength
at some point in time with the lowest cost, even for free. Such selfishness of network selection greatly influences network performance, such as throughput and delay, impairing individuals’ performance consequently in the long run. The authors in [17] proposed a distributed hedonic coalition formation game for network selection based on only RSS, showing the payoffs could be improved by up to 33.37% over the pure individual selection scheme based on RSS. However, the cost function in [17] with respect to RSS does not reflect realistic and specific requirements and performance in networks. The centralized resource allocation algorithm proposed in [10] considers weighted sum of spectral efficiency, fairness, and battery lifetime utilities, however, the algorithm requires hand-tuned weights and non-trivial computing resources. In this dissertation, a coalition formation game, taking account of both user preferences and network information, is proposed to achieve centralized computing and distributed decision making for network selection. The large-scale network information providing for vehicles, such as network availability, traffic load, available bandwidth, etc., is collected, managed, and preserved in clouds. Using the proposed game-based algorithm, not only vehicles could keep robust connectivity, but also networks get traffic loads more balanced and resource allocation more efficient. A fast convergence algorithm for solving a coalition formation game, making possible optimization for larger scale networks, enabling practical implementation of coordinated algorithms in vehicular networks.

2. Inter-session network coding based data forwarding: for data forwarding in vehicular networks, P2P collaboration via V2V communications is considered as a useful way for content distribution, mainly in the areas infrastructure could not cover. The recent CodeOn [41] and Code-Play [94] showed the average download delay could be shortened by up to 10x compared to P2P based approaches [96, 85], by leveraging network coding. Both of the two approaches consider popular content distribution among vehicles in drive-thru based scenarios using broadcasting in a transmitting vehicle’s covered region. However, the fundamental problem of large delay for data forwarding in vehicular network, e.g. the average downloading delay for a 16 MB file is 424 s in 100-vehicle scenario with network coding [41], was not solved by these solutions. Our preliminary work [91] has proposed an intra-session network coding framework across multiple RSUs for vehicular networks using multicasting, and the results show the throughput is improved by 25 ~ 30% but not requiring more bandwidth in core networks. However,
unicasts from Internet to end users are rapidly emerging and will be dominant in the near future (especially for IP video which is predicted to take 79% of the overall Internet traffic by 2018) [20], rather than multicast/broadcast and/or P2P based common content distribution. Furthermore, maintaining low delay and packet loss rate for moving vehicles over wide area links is difficult today. In this dissertation, an efficient and scalable data forwarding approach based on inter-session network coding is designed for vehicular networks by leveraging cloud-based strategies. The proposed solution, involving both Internet core and wireless edge, is expected to maintain reliable and persistent communications for vehicles in the presence of dynamic changing network topologies.

3. A content-based data retrieval framework: today’s routing and transport mechanisms in vehicular networks still inherit traditional Internet Protocol architecture (commonly known as TCP/IP) to achieve data retrieval for vehicles. In the recent decade, researchers have proposed many opportunistic scheduling schemes tailored for vehicular networks, e.g. MV-MAX [26], load-reducing scheduling [90], and pre-caching scheduling [2]. However, opportunistic scheduling alleviates symptoms of unstable performance in vehicular networks but not enough to rid the root cause— the vulnerability of connection-based and client-server architecture using TCP/IP in the presence of fast speed. Even if opportunistic scheduling is applied, any unexpected signal disturbance or any untimely scheduling decision still may result in burst retransmissions, connection suspension or breaks, etc. Furthermore, opportunistic scheduling faces a number of problems that obstruct real implementation. This dissertation proposes a content-centric vehicular networking (CCVN) as a clean-slate solution to achieve efficient data retrieval. The network topology and vehicle’s movement information is maintained in the cloud and provided for face management in CCVN. The proposed framework breaks through limitations of legacy connection-centric and host-centric TCP/IP architecture. Due to the connection-less, in-network caching, and distributed characteristics, CCVN has the ability of adapting to dynamic, short-lived and intermittent connectivity easily by nature, e.g. vehicular environment.

By making connectivity more robust and transmission more efficient using proposed strategies, this dissertation is aimed at improving both network and individual performance as much as possible, in the presence of fast speed and resource limitations, which are two inevitable “root causes”
in vehicular networks. All cloud-based strategies in this dissertation take into account both user-centric and network-centric metrics, as microscopic and macroscopic measurements, respectively. A generic cloud-based system is proposed for supporting vehicular networks. This dissertation is expected to provide guidance for implementing applications in the real world, such as telematics and location-based services.

1.4 Dissertation Outline

The rest of the dissertation is organized as follows. The background and related work is presented in Chapter 2. The cloud-based system for vehicular networks is proposed in Chapter 3. A fast coalition formation game based network selection scheme is proposed in Chapter 4. An inter-session network coding scheme for Internet-to-vehicle unicasts is proposed in Chapter 5. A content-based data retrieval framework for vehicular networking is proposed in Chapter 6. The dissertation is concluded in Chapter 7.
Chapter 2

Background and Related Work

2.1 Network Selection

Different mathematical theories are used in understanding and modeling the network selection problem. In the literature, a large number of studies have considered the network selection problem using utility theory, multiple attribute decision making (MADM), fuzzy logic, game theory, combinatorial optimization, Markov chains, etc. A thorough survey on the prior work was presented in [82]. In this section, a small set that are most related to our work is identified.

MADM is widely studied, mostly considered for network selection problems, and refers to making preference decisions over the available alternative networks characterized by multiple attributes. The most popular MADM algorithms are Simple Additive Weighting (SAW) [75], Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) [14], Multiplicative Exponential Weighting Method (MEW) [46], and Grey Relational Analysis (GRA) [70]. Comparisons between these algorithms are studied in [46, 72]. A set of weights are required by MADM methods to determine the relative importance of each attribute. The assignment for weights has significant impact on the solution space. The choice for weights usually is based on imprecise end-user preferences [44], attributes’ importance from simulation results [39], and conclusions from network providers’ statistics [10]. Analytical Hierarchy Process (AHP) is a popular method to determine the relative weights by dividing a complicated problem into a hierarchy of attributes [37]. Each attribute is weighted according to their relative dominance. In [10], a central global resource controller is used to make an optimal decision based on the weighted sum of spectral efficiency, fairness, and battery lifetime.
utilities. The weights are calculated using empirical values from network providers and AHP method.

In addition to MADM, [82] broadened the survey to include fuzzy logic, Markov chains, and game-theoretic methods for optimizing joint multiple-attribute utility in heterogeneous wireless networks. The work concluded that MADM, fuzzy, and Markov algorithms are more suitable when network traffic is not a key constraint, while game-based algorithms are more suitable when network resources are limited. The reason is that the equilibrium of a game has the tendency to uniformly distribute users into different networks if the utility is highly correlated to the bandwidth obtained by selecting certain networks. However, when compared to the other kinds of algorithms, such equilibrium is not the best solution if the bandwidth is sufficient in the networks.

In a game, players seek to maximize their payoffs by choosing strategies that deploy actions depending on the available information at a certain moment. A widely adopted solution of a game is Nash equilibrium, where each player cannot further benefit by changing his/her strategy while keeping the other players’ strategies unchanged. Game-based network selection algorithms are distributed in nature, since decision-making processes are conducted by each independent player. An extensive survey on game-based network selection algorithms was made in [76], dividing recent works into three categories (users vs. users, users vs. networks, and networks vs. networks), two types (non-cooperative, cooperative) and 16 typical game models. Most approaches involving end users are non-cooperative [86, 11], in which players select his/her strategy individually (typically bandwidth and application quality of service requirements for the users, and profit for the networks). Other work has also proposed cooperative approaches that jointly consider improving payoffs for the other players [12]. Cooperative approaches, which are more frequently studied in networks vs. networks games, are mostly focused on spectral efficiency, resource allocation, admission control, and load balancing problems that consider network metrics such as available bandwidth, network load, etc. [53]. Nearly half of recent game-based works in survey [76] are focused on fair network resource allocation, and the other half are focused on selecting the best network with highest payoffs for users. Although most works consider both users preferences and network performance, payoffs are still biased on one party (either users or networks), with the other party’s metrics performing as constrains.

Recently, coalition formation games have been considered as an effective framework for a distributed and cooperative approach to allocate resources/tasks or select networks [54, 65, 64, 68, 66, 17]. In a coalition formation game, players can self-organize into stable coalition structures with-
out obtaining higher payoffs by switching current coalitions (namely, Nash equilibrium). In these works, the framework of coalition formation games is consistent and the utility functions vary based on different objectives and metrics. Among these models, a typical three-phase algorithm is used, which consists of Phase I - resource discovery, Phase II - an iterative coalition formation loop, and Phase III - data transmission in the ultimate coalitions. The convergence in Phase II, after several iterations, is necessary to achieve Nash equilibrium, which is time-consuming and a critical limitation for implementation in the real world. In [54], a coalition formation game based on Markov chains is proposed to efficiently share bandwidth among rational players (vehicles) in vehicle-to-roadside (V2R) networks, increasing the total utility by 17% compared to the non-cooperative case. In [68], a coalition formation game is proposed to formulate the cooperation among RSUs to coordinate the classes of data being transmitted to vehicles, for which the average payoff is improved by 20.5% to 33.2% compared to the non-cooperative scheme. Although these two models are targeting vehicular networks, no typical vehicular environments and road topologies are considered and measured. Furthermore, the performance comparisons in all the above models use theoretical optimal solutions, cooperative game-based models, and non-cooperative schemes.

To propose an efficient cooperative approach for network selection, this dissertation formulates vehicular networks using a coalition formation game model. To overcome the difficulties of vehicle computing limitations and information acquisition in conventional game-based distributed systems, a cloud-based system for network selection is proposed. A one-iteration algorithm is proposed to break through the limitation of long convergence times in conventional iterative coalition formation algorithms [65, 64, 17]. The performance of the proposed network selection scheme is compared with a centralized optimal solution [10] instead of theoretical analysis and is evaluated in three typical vehicular scenarios, which are rarely considered in the related studies.

2.2 Data Forwarding

P2P collaboration via vehicular communications is widely studied recently for content distribution in vehicular networks. In [50], SPAWN, a pull-based P2P content downloading protocol, was proposed as a first attempt to study cooperative downloading services in vehicular networks. AdTorrent was proposed later which is a semi push-based P2P protocol for vehicles to download advertisements they are interested in [51]. However, the peer and content selection processes have
high overhead and are not scalable due to not considering broadcast paradigm, especially when most vehicles are interested in downloading similar or common contents. In [96], VC-MAC was proposed leveraging the broadcast nature of the wireless medium to maximize the system throughput, particularly for gateway-downloading scenarios. VC-MAC [96] and game-based cooperative P2P approaches [85] improve the system throughput by a factor of 2 to 2.5 approximately, using broadcasting by infrastructure.

Since the peer and content selection mechanisms have high overhead and are not resistant to error-prone channels, network coding based methods are emerging as a commonly considered way for content distribution via V2V communications. To reduce duplicated transmissions and simplify the scheduling, network coding has been adopted in many existing works [4, 41, 94, 25, 51, 59, 93, 38, 40, 32]. In [4], VANETCODE is proposed to enhance cooperative content sharing in VANETs without introducing additional overhead using packet-level network coding (PLNC). Taking advantage of better error tolerance of symbol-level network coding (SLNC), CodeOn [41] and CodePlay [94] were proposed to maximize thedownload rate and minimize the delay for popular content distribution in VANETs based on SLNC, showing the average download delay could be shortened by up to 10x compared to P2P based approaches, by leveraging network coding. Both of the two approaches consider popular content distribution among vehicles in drive-thru based scenarios using broadcasting in transmitting vehicle’s covered region. Even though network coding shows benefits for common content distribution in vehicular networks, the fundamental problem of unstable throughput still exists, resulting in large downloading delay. Meanwhile, these schemes are not able to be automatically applied to today’s dominant traffic-core-to-edge unicast sessions-for vehicles on the move. Therefore, more generic and scalable frameworks, involving both core networks and edge networks, are important and open questions in an under-explored area.

Network coding was originally introduced based on an initial purpose of enhancing throughput [3]. In [3], a commonly considered butterfly network features a multicast from a single source to two destinations. The throughput can get benefits by communicating more information with fewer packet transmissions in the bottleneck link in a typical butterfly network. The major advantages of network coding are throughput improvement [34], robustness to packet losses [74, 98], and routing efficiency [43]. Network coding naturally becomes an efficient solution to enhance throughput and robustness to packet losses or link failures in wireless networks due to inherent broadcast and overhearing capabilities [33]. Intra-session network coding is used for a single communication
session, i.e. unicast communication to one sink node or multicast of common information to multiple sink nodes. Inter-session network coding, i.e. coding among information symbols of different sessions, is more complicated especially for session grouping strategies. Prior works in vehicular networks [4, 41, 94, 91, 51, 59, 93, 38, 40, 32] have used intra-session extensively since common content distribution is considered. Although some works address inter-session network coding for wireless networks [33, 69], operating inter-session network coding in the presence of fast-changing topologies in vehicular network is an open problem.

To propose an efficient data forwarding scheme for vehicular networks, this dissertation takes account of unicasts across the core and the edge for vehicles instead of common content distribution in the edge only. A cloud-based system for network coding is proposed to address a scalable framework for vehicular networks. To overcome the difficulties of operating inter-session for dynamic topologies, a conditionally optimal grouping algorithm is proposed, by using information provided from a distributed cloud. Different network selection algorithms are evaluated as a factor in the proposed system, which is rarely considered in the related studies but is not negligible in vehicular networks.

2.3 Data Retrieval

TCP/IP is a widely adopted communication protocol stack of the Internet. Though such architecture was originally designed for wired networks with fixed topology, it is still commonly applied to vehicular networks. TCP/IP uses a client-server model of communication to establish host-to-host connectivity, in which TCP handles assembling/reassembling of message or file and transmission control, and IP manages address control and packet routing over the Internet. Another popular transport protocol UDP is a connectionless and best-effort datagram protocol. By better distinguishing between error-prone links and network congestions, a TCP-based sublayer is proposed to enhance TCP performance in vehicular networks [15]. Researchers also studied the effects of tuning transmission power on TCP [19] and UDP [35] throughput in multi-hop vehicular ad hoc networks. A scheme of reducing bandwidth wastage due to residue time caused by channel switching is proposed to improve TCP and UDP performance in vehicular networks [84]. However, optimization of TCP/UDP performance is still not sufficient to adapt to fast-changing channel conditions and intermittent connectivity.
Opportunistic scheduling was first introduced using the multiuser diversity to improve the capacity significantly taking account of channel conditions [36]. In nearly twenty years, a large number of studies have explored opportunistic scheduling in wireless networks, being surveyed in [13]. Over the last ten years, researchers have started bringing opportunistic scheduling into vehicular networks since it can be well fit into dynamic varying channel conditions [26, 90, 44]. Existing works are mainly aimed to improve capacity, quality of service (QoS), or a combination of these two. In [5], a link-layer scheduling mechanism for non-real-time, non-safety data transmission in vehicle-to-infrastructure (V2I) systems is proposed for IEEE 802.11e standard, attempting to deliver as much information per flow as possible considering both constrained radio coverage of road segment and vehicle speed. MV-MAX was proposed as a medium access protocol that opportunistically grants wireless access to vehicles with the maximum transmission rate, improving overall system throughput by a factor of almost 4 with respect to the IEEE 802.11 protocol [26]. Load-reducing scheduling was proposed to transmit packets only with high link reliability [90], therefore reducing burst retransmissions, resulting in saving bandwidth resources and improve QoS at the same time. A centralized controller uses an access point coverage map to schedule pre-caching on possible RSUs along the predicted vehicle’s travel path is another mechanism to avoid disconnectivity and reduce download time during data delivery [2].

However, all of these opportunistic scheduling schemes working with TCP/IP protocols do not address the cause of the vulnerability of the client-server model in vehicular scenarios. Furthermore, opportunistic scheduling is difficult to implement because of limitations encountered in practice. First, full knowledge of the channel state of all users at all BSs is widely considered as an assumption. If all users send feedback on every slot for all available subchannels to all associated BSs, such behavior results in large overhead and does not scale. Second, limited computing ability and scarce memory on BSs cannot handle complex scheduling algorithms. Most proposals assume offline scheduling instead of taking account of real-time requirements for simplicity. Today’s deployed BSs cannot satisfy high computation and modeling complexity for online scheduling. Third, difficulty of cross-layer design and compatibility of different RATs result in scarce interest for practical applications.

This dissertation utilizes content-centric networking (CCN) for efficient data retrieval in vehicular networks instead of TCP/IP with opportunistic scheduling. Current TCP/IP stack’s network layer is commonly considered to form the waist of the hourglass with tremendous growth
of number of devices in the Internet. CCN is a new clean-slate architecture of Future Internet which is mainly proposed to break through the limitations of connection-centric and host-centric basis for scalable content distribution [63]. By naming content at the network layer, CCN favors the deployment of in-network caching and multicast mechanisms, thus facilitating the timely and efficient content delivery to users.

Some researchers have already started considering the candidacy of CCN in vehicular networks [83, 6, 81, 7, 22, 9, 8, 80, 95]. In [83], a naming structure is proposed for traffic information dissemination, in which the name components identify the temporal and geographical scopes of traffic information. Many works discuss routing and forwarding strategies in CCN, e.g. next-hop selection [81], path selection [22], provider selection [7, 9], data advertisement [95]. Some works consider the transport layer, e.g. RTT estimation [8], and security [80].

Even though these studies have validated the effectiveness of using CCN in vehicular networks, the common concern from academia and industry has not been cleared yet. The question of whether CCN can fulfill all requirements of data retrieval in vehicular networks which recent proposals based on TCP/IP cannot handle easily still needs to be answered. Existing literatures prove the CCN as a potential solution can answer this question, however, none of them actually systematically measures and compares CCN to recent technologies facing tremendous challenges in today’s vehicular networks, e.g. opportunistic scheduling. This dissertation is focused on exploring and enhancing CCN’s ability in vehicular networks.

### 2.4 Cloud-based Systems for Vehicular Networks

Recently, cloud computing has drawn significant attention as a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources [48]. By extending cloud computing to mobile scenarios with wireless access, mobile cloud computing refers to an infrastructure where both the data storage and the data processing happen outside of the mobile device [47]. The architecture of mobile cloud computing contains three layers: the Application layer, the Platform layer and the Infrastructure layer, and each of these layers provides a specific service for users as Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS), respectively [21]. Evolved from mobile cloud computing, vehicular cloud computing is able to coordinate computing, sensing, communication and physical resources and dynamically
allocate to authorized users among a group of vehicles [56].

As vehicular cloud computing is an emerging and promising area in the near future, academia, vehicle manufacturers and service providers have focused on research and development to bring cloud-based system into vehicles. Ford combines social networks, GPS location awareness, and real-time vehicle data for future innovative applications in connected vehicles by using the cloud [24]. By using Microsoft Windows Azure cloud platform, Toyota vehicles are envisioned to be equipped with the latest technology to access telecommunications information, streaming music, energy management and GPS services while on the move, within 2015 [71]. General Motors is developing a V2V/V2I-based communications system, providing in-vehicle Wi-Fi hot spots for mobile devices, entertainment, and fast downloading services [62]. Besides physical datacenter taking charge of data computation and storage, researchers also considered the cloud formed by resources provided by participating vehicles, or combinations of vehicles and infrastructure-based resources. MobiCloud was proposed as a geo-distributed platform, allowing users to connect to cloud resources that are geographically closer to them, reducing communication delay [28]. VehiCloud was proposed as a service-oriented cloud architecture to provide routing service in vehicular networks, by predicting vehicles’ future locations in the cloud using way point messages [61].

Applying cloud computing technologies to vehicular networks is still a very new direction, which should take account of the features in vehicular networks, such as fast speed. This dissertation designs three critical strategies by leveraging the cloud-based system. The design of an efficient architecture/structure for cloud systems is beyond the scope of this dissertation.
Chapter 3

The Proposed Cloud-based System

As the basis of the proposed strategies in this dissertation, a cloud-based system is proposed in this chapter. Section 3.1 introduces an overview of the proposed cloud-based system. Section 3.2 describes distributed database. Section 3.3 discusses key assumptions. Lastly, Section 3.4 gives an envisioned architecture of the cloud-based system for vehicular networks.

3.1 System Overview

Fig. 3.1 depicts an overview of the proposed cloud-based system. The overall system consists of a distributed cloud, multiple routers in the core network, multiple RSUs in the edge networks, and a set of vehicles in the coverage area. The distributed cloud is a key component in the proposed system, which takes charge of establishing, maintaining the database, computing tasks for proposed strategies, and communicating with routers, RSUs and vehicles. The cloud is geographically distributed and can be flexibly accessed by vehicles according to their locations. Multiple servers at different geographical locations, which could be either real physical machines or virtual machines, reside in the cloud. The cloud connects to the Internet and provides information access to all federated network operators. In the cloud, a distributed database optimizes data maintenance and updates based on geographical location. Database updates are provided by resources including routers’, RSUs’ and vehicles’ information. The database has full knowledge of the network topology, configurations and status, and the cloud is able to compute the cost of network-related metrics and provide optimal strategies. The cloud is used to collect and disseminate necessary information after
computing and processing. All routers send updates of their status to the cloud such as which vehicles’ packets are being forwarded. All RSUs send updates of their status such as which vehicles are being served. The vehicles are equipped with GPS devices, and the on-board device connecting to the network is assumed to be able to import data from GPS and integrate different wireless access technologies. Vehicles periodically report their status, including current location, speed, RSS, and battery status, through associated RSUs to the cloud, about $1 \sim 5 \text{Hz}$ based on GPS’s updating.
rate. The updates will not add much traffic to the network since the packet is typically very short (less than 100 B) [55].

3.2 Distributed Database

The proposed cloud-based system consists of a cluster of distributed database that work together to allow seamless access by vehicles as their locations change. This section first introduces what information is stored in the distributed database, and then it explains how to build and maintain the database.

3.2.1 Information in the database

Each distributed database consists of three parts: vehicle information, RSU information, and a GPS-tagged, grid-based network map, as illustrated in Fig. 3.2. The vehicle and RSU information is based on periodic updates from vehicles and RSUs, respectively. This information reflects the updated and changing status and conditions on individual vehicles and RSUs, by including time stamps. The grid-based network map records historical information, such as RSS, throughput, and traffic flow for each grid corresponding to GPS locations. The statistics are important for designing strategies, such as network selection decisions. For example using average normalized RSS based on...
most recent information in a grid for a certain RSU could indicate the average signal quality level for this RSU, thus avoiding instantaneous fluctuation. There might be situations however where older information can help to improve the accuracy of decision. For example, at 5:00 PM each day at a particular location the RSS of an RSU drops by 20 dB. The network map and vehicle/RSU information sets are able to exchange information with each other.

3.2.2 Database maintenance

The database maintenance includes database establishment and update. Preliminary information collection is used for initially establishing the database, in particular for the network map. During the collection procedure, the testing vehicle records the RSS corresponding to particular RSUs mapped into grids, and then uses software tools to obtain performance samples. The database requests the traffic flow statistics from remote resources periodically and records the network map. Online update is for real-time updates of vehicle/RSU information in the database, which obtains the information from periodic update messages sent by active vehicles on the road and RSUs. Some updating information, e.g. RSS, number of active vehicles, is added to historical records in the map. As an example, the CyberTiger project at Clemson University provides a prototype of a distributed database for a single local area [1].

3.3 Key Assumptions

The common open challenge of proposed cloud-based strategies is the cloud communication latency and inherently intermittent connectivity. Large communication latency and packet losses caused by connectivity discontinuity will make critical information, provided by the cloud, unusable or expired. Hence the vehicles decisions or actions will regress to traditional solutions, getting no benefits from the cloud. This dissertation assumes every vehicle is equipped with GPS devices, which is able to periodically report the vehicle’s location and driving information such as speed to the cloud. Nowadays GPS devices are inexpensive and widely adopted, and more and more cars or mobile devices also already embed GPS components. The requirement for the cloud is powerful computing ability to analyze vehicles movements and predict future locations. So the cloud has the ability of sensing potential high packet loss rate or connectivity breaks beforehand, and then decides the timing of sending any necessary information. The cloud assumed in this dissertation is
push-based, so that the timeliness of pushing all necessary information or decisions to vehicles could be guaranteed based on the analysis and predictions. In the proposed game-based network selection algorithm, once the cloud considers a certain vehicle will need to execute a handoff soon, map-based information for network selection will be sent to the desired vehicle. Though propagation conditions vary, the cloud is able to decide to push information before critical and persistent signal strength drop, ignoring instantaneous disturbances. Such decisions are based on long-term history records in the map-based information database stored in the cloud. In network coding based solution, map-based information for network selection and transmission scheduling guidance will be pushed to vehicles for network selection by the cloud. With the proposed content-based data retrieval framework, the decisions of managing connection points on vehicles will be retrieved before sending a real data request. Though this challenge still needs to be overcome by designing efficient cloud structures and communication mechanisms, it is beyond the scope of this dissertation.

3.4 An Envisioned Architecture

For designing a cloud computing architecture for vehicular networks, a broad survey has been made in [87]. Based on the features in vehicular networks, the authors in [87] proposed an architecture of vehicular cloud computing. The proposed architecture relies on three layers: inside-vehicle layer, communication layer and cloud layer. The design of an efficient architecture for vehicular cloud computing is not discussed in this dissertation. As an example, an envisioned cloud-based architecture for vehicular networks is illustrated in Fig. 3.3. The three layers are described as below.

3.4.1 Inside-Vehicle Layer

This layer contains several different components in a vehicle, which are GPS, internal sensors, radio access technology (RAT) units, storage unit, computational unit, etc. Nowadays, an on-board unit (OBU) is widely equipped with a vehicle including a GPS component, interfaces with vehicle’s internal sensors, RAT interfaces, storage and computation units, or even more. The OBUs have the capability of collecting raw data (driving, location, sensor, etc.) from the vehicle, and have a broadband wireless communication to transfer data through RAT units. The OBUs are normally assumed to have a build-in navigation system, with a map and the location of RSUs.
3.4.2 Communication Layer

This layer includes vehicle-to-vehicle (V2V) system, vehicle-to-infrastructure (V2I) system and/or a hybrid of the former two. The systems in this layer use today’s radio access technologies, such as Wi-Fi, DSRC, LTE, etc. Through this layer, the data/messages could be exchanged between individual vehicle and the cloud storage, or among vehicles. The location updates, driving status or other reports will be sent to the cloud storage via this layer through infrastructure from vehicles, and the data or messages from applications or services will be sent to vehicles through this layer as well. For any safety or emergency messages on the road, data could be directly sent via V2V communication.
3.4.3 Cloud Layer

This layer provides the capability of computing the massive and complex computations in minimal time. Three internal layers are consisted of this layer: applications and services layer, cloud platform layer and cloud infrastructure layer. In the application layer, various applications and services could be accessed remotely by OBU s. The services could be any deployed standard services or operator-customized services, such as Network as a Service (NaaS), Storage as a Service (SaaS), Cooperation as a Service (CaaS), Information as a Service (INaaS), Entertainment as a Service (ENaaS), and Handoff as a Service (HaaS) [89]. The cloud platform layer is in charge of any environment for distributing storage, parallel programming design, the management system for organizing distributed file systems and other system management tools for cloud computing. The cloud infrastructure layer consists of cloud storage and cloud computation. The data gathered from vehicles will be stored in distributed database based on the type of application or services. The computation part could be any computational resources from datacenter, or distributed servers, providing abundant computing capability for services.

By taking advantage of this envisioned cloud-based architecture for vehicular networks, this dissertation proposes three cloud-based strategies in the following chapters.
Chapter 4

Coalition Formation Game based Network Selection

Network selection schemes have drawn great attention for the objective of achieving vehicle connectivity performance, cost efficiency and overall network efficiency. Numerous studies have proposed different schemes for network selection/vertical handoff [82, 75, 14, 46, 70, 72, 76, 39, 10, 37, 54, 65, 64, 68, 66, 17, 89]. In today's heterogeneous wireless networks, decision-making for network selection could be centralized by networks, distributed by vehicles, or cooperative by both. The previous schemes fall into two typical categories: utility-based optimization and game theoretic frameworks. Both forms of solutions require non-trivial computing complexity. With the scarce memory and processing capacity on typical RSUs and OBUs, it is difficult to satisfy computing tasks, especially for optimization with sophisticated utility functions in a large scale of vehicular networks. Practical implementation of such solutions must trade off scope and granularity of network information for algorithm efficiency. Furthermore, complex and dedicated design for utility functions and hand-turned weighting factors in conventional network selection algorithms limit the flexibility to adapt to diverse performance requirements.

In this chapter, a fast and game-based network selection scheme is proposed for vehicular networks by leveraging the cloud system’s abundant computing resources. The proposed scheme provides the framework with flexibility of using customized utility functions. The rest of the chapter is organized as follows. Section 4.1 introduces network model and utility functions. Section 4.2
formulates the coalition formation game in vehicular networks. Section 4.3 proposes a one-iteration network selection algorithm. Section 4.4 analyzes the results.

4.1 Network Model and Utility Functions

4.1.1 Network Model

The system considers a network consisting of $N$ vehicles in the service areas of $K$ RSUs, which are the same or are of different types. The set of vehicles is denoted as $\mathcal{V} = \{v_1, \ldots, v_N\}$. Each vehicle is equipped with at least one type of radio or multi-radio devices to get access to different RANs. RANs are the typical networks widely adopted nowadays. All vehicles are equipped with GPS devices. The on-board device connecting to the network is assumed to contain different wireless access technologies and is able to import data from its GPS. The vehicle periodically reports its status when connecting to a certain network, including location, speed, battery status, etc. to the database. At time $t$ vehicle $v$’s GPS coordinate is $(\text{lat}, \text{lon})_v$, $v \in \mathcal{V}$, where lat and lon are the latitude and the longitude, respectively. The set of RSUs is denoted as $\mathcal{A} = \{a_1, \ldots, a_K\}$. RSU $a \in \mathcal{A}$ has a coverage distance $C_a$. Each RSU $a \in \mathcal{A}$ offers a total bandwidth $B_a$ and a maximum link transmission capacity $\mu_a$, in bits per second. Each queue on a RSU is modeled as an M/G/1 queuing system. The average packet arrival rate for each vehicle $v \in \mathcal{V}$ is denoted as $\lambda_v$.

At time $t$, each vehicle $v \in \mathcal{V}$ makes a decision to select a best network and access the selected network. Each RSU $a \in \mathcal{A}$ serves a set of vehicles $S^a_t \subseteq \mathcal{V}$, consisting of $|S^a_t|$ active vehicles who access RSU $a$’s network. Here, the set of vehicles denoted as $S^a_t$, in which all vehicles join RSU $a$’s network and request for services at time $t$, forms a coalition. At a given time, a vehicle is only allowed to use one single network interface, which implies coalitions are disjoint. In fact, “coalition” is the term in coalition formation game, and more details are introduced in Section 4.2. It is defined here for the ease in understanding utility functions formulated in the next subsection.

4.1.2 Utility Functions

A utility function is quantified for each user to select the best network. In the proposed coalition formation game framework, users can choose any appropriate utility function. The only requirement is the designed utility function must be a non-increasing function in terms of the number
of members in the coalition. Such requirement actually comes from the proposed fast convergence coalition formation algorithm, which is described in Section 4.3. The design of utility functions differs much in terms of different demands and objectives in the networks. The attributes considered in utility functions could be any performance metrics, such as throughput, delay, jitter, etc., or availability metrics, such as average RSS, battery status, sojourn time, or pricing metrics such as per-time-unit cost. Furthermore, the metrics could be user-centric, or network-centric, such that utility functions could be designed as user-oriented or network-oriented, or a combination of both.

In this chapter, two typical performance-driven utility functions are considered, along with two representative utility functions using the SAW and GRA methods respectively.

Before presenting the utility functions, several attributes need to be introduced first. The RSUs’ availability is associated with vehicle’s location. The availability of RSU $a$ is given as

$$
epsilon_v^t(a) = \mathbb{1}_{C_a}((\text{lat}, \text{lon})^t_v) = \begin{cases} 
1 & \text{if } d^t_a \leq C_a \\
0 & \text{if } d^t_a > C_a
\end{cases} \quad (4.1)$$

In (4.1), $d^t_a = \sqrt{(\text{lat}_v^t - \text{lat}_a)^2 + (\text{lon}_v^t - \text{lon}_a)^2}$ is the distance between vehicle $v$’s current location and RSU $a$’s location, where $(\text{lat}_a, \text{lon}_a)$ is the RSU $a$’s location. Therefore, vehicle $v$ is able to get a set of available RSUs at time $t$, which could be represented as $\epsilon^t_v = \{a_i : \epsilon^t_v(a_i) = 1, a_i \in A\}$. The average RSS information of every RSU in the grid is stored in the database. Whenever the vehicle sends the updated GPS locations the server pushes the RSS information back to the vehicle. The location based RSS of RSU $a$ can be given as

$$\delta^v_a = \text{RSS}_a((\text{lat}, \text{lon})^t_v) \quad a \in \epsilon^t_v, v \in V \quad (4.2)$$

Therefore, vehicle $v$ is able to get a set of RSUs’ average RSS at time $t$, which could be represented as $\delta^v_a = \{\delta^v_a, a \in \epsilon^t_v\}$. The actual transmission rate is determined by modulation and coding schemes, which depend on RSS information. So the transmission rate for vehicle $v$ at time $t$ in RSU $a$’s network could be denoted as $\mu^t_v(\delta^v_a)$ which is a function of the RSS value, where $0 < \mu^t_v(\delta^v_a) \leq \mu_a$. The modulation and coding schemes depend on specific radio access technologies and vary between different network operators.

Two typical and generic performance metrics are throughput and delay, which are the
most commonly considered attributes in MADM methods. Given the M/G/1 queuing model, the
normalized throughput in coalition \( S_t \) is calculated as

\[
\eta(S_t) = \min \left( \frac{\sum_{v \in S_{ta}} \lambda_v \cdot L_v}{\mu_a(\delta_{va})}, 1 \right)
\]  

(4.3)

In (4.3), \( L_v \) is the average packet size in bits for vehicle \( v \), and \( \eta(S_t) \) is equal to the utilization \( \rho(S_t) \) for coalition \( S_t \) in RSU \( a \)'s network. The closed form expression of delay in coalition \( S_t \) can be calculated using Pollaczek-Khinchine (P-K) formulas,

\[
\tau(S_t) = \begin{cases} 
\frac{\rho(S_t) \cdot \left( 1 + \frac{\sigma^2_a}{\mu_a} \right)}{\eta_a(1 - \rho(S_t))} & \rho(S_t) < 1 \\
+\infty & \rho(S_t) \geq 1
\end{cases}
\]  

(4.4)

where \( \frac{1}{\eta_a} = \sum_{v \in V} \left( \frac{\lambda_v \cdot L_v}{\lambda_i \cdot L_i} \cdot \frac{1}{\mu_v(\delta_{va})} \right) \) is the average transmission rate in coalition \( S_t \), and \( \sigma^2_a = E \left[ \left( \frac{1}{\mu_v(\delta_{va})} \right)^2 \right] - \frac{1}{\eta_a} \) is the variance of the average service time.

### 4.1.2.1 Utility function I- available bandwidth

The first utility function is available bandwidth. At time \( t \), assume the number of vehicles served by RSU \( a \) is \(|S_a|\), and assume the total bandwidth \( B_a \) is equally allocated among all the vehicles in the coalition \( S_t \). The available bandwidth of RSU \( a \) seen by a vehicle \( v \in V \), which is not in the coalition \( S_t \) currently, is given as

\[
U_b(S_t) = \frac{B_a}{|S_t| + 1} \quad a \in A
\]  

(4.5)

In (4.5), the right-hand equation means the bandwidth allocated to the vehicle \( v \), if it decides to handoff to RSU \( a \)'s network and be a member of \( S_t \).

### 4.1.2.2 Utility function II- power of the network

Two principal performance metrics for any vehicle \( v \) is throughput and delay. To capture the fundamental tradeoff between throughput and delay, the power of a network, is considered to use the ratio of throughput to delay as a metric for evaluating the effectiveness of a resource allocation scheme [64, 60]. The power of RSU \( a \)'s network is the ratio of throughput to packet delay in the
coalition $S_a^t$,

$$U_p(S_a^t) = \frac{\eta(S_a^t)^\beta}{\tau(S_a^t)^{1-\beta}} \quad a \in A \quad (4.6)$$

where $\eta(S_a^t)$ is the aggregate throughput in the network served by RSU $a$; $\tau(S_a^t)$ is the average delay of packets in the same network; and $\beta \in (0, 1)$ is the tradeoff factor between throughput and delay. The factor $\beta$ is chosen based on the relative emphasis placed on throughput versus delay.

In order to compare with different methodologies, two representative MADM methods, SAW and GRA are considered. The next two utility functions are used for these two methods, respectively. In this section, throughput and delay are used as two attributes in the SAW and GRA methods. Both SAW and GRA are based on the weighted sum of multiple attributes, so an appropriate normalization method is necessary. Max-Min, which calculates the proportional ratio between the considered value and the best value for the attribute, is used as the normalization method. If the attribute is larger-the-better, it is called upward attribute, likewise, it is called downward attribute if it is smaller-the-better [82]. For simplicity to be normalized, the attributes are converted to be upward. Therefore, the normalized throughput and delay are given as

$$\tilde{\eta}(S_a^t) = \frac{\eta(S_a^t) - \min_{a \in \epsilon^t} (\eta(S_a^t))}{\max_{a \in \epsilon^t} (\eta(S_a^t)) - \min_{a \in \epsilon^t} (\eta(S_a^t))} \quad (4.7)$$

$$\tilde{\tau}(S_a^t) = \frac{\frac{1}{\tau(S_a^t)} - \min_{a \in \epsilon^t} \left(\frac{1}{\tau(S_a^t)}\right)}{\max_{a \in \epsilon^t} \left(\frac{1}{\tau(S_a^t)}\right) - \min_{a \in \epsilon^t} \left(\frac{1}{\tau(S_a^t)}\right)} \quad (4.8)$$

4.1.2.3 Utility function III- simply additive weighting (SAW)

SAW is the simplest and most widely used MADM method. The utility function is calculated as a weighted average of normalized attributes,

$$U_{SAW}(S_a^t) = w_{\eta} \cdot \tilde{\eta}(S_a^t) + w_{\tau} \cdot \tilde{\tau}(S_a^t) \quad a \in A \quad (4.9)$$
4.1.2.4 Utility function IV- gray relational analysis (GRA)

GRA considers the distance from the evaluated value of an attribute to the best reference value, which is calculated as

\[ U_{\text{GRA}}(S_t^a) = \frac{1}{w_\eta \cdot |\tilde{\eta}(S_t^a) - \tilde{\eta}_0^t| + w_\tau \cdot |\tilde{\tau}(S_t^a) - \tilde{\tau}_0^t| + 1} \]  

(4.10)

where \( \tilde{\eta}_0^t \) and \( \tilde{\tau}_0^t \) are the best reference values of normalized throughput and delay, respectively.

The four utility functions given in this section are all larger-the-better, and are calculated for a given coalition \( S_t^a \), \( a \in A \). Note that the utility functions used in the proposed network selection system are not limited to these four functions or throughput-delay related functions. Utility functions can be designed based on specific demands, such as performance based, power consumption based, price based, or mixed metrics. The next two sections discuss the utility function requirements and the proper use of the utility functions for network selection.

4.2 Game Formulation in Vehicular Networks

Coalitional games have been widely explored in different disciplines to study the behavior of rational players when they cooperate to form coalitions. A fundamental tutorial on coalitional games is given in [67]. Coalition formation games as an important class in coalitional games are frequently used for network selection in the field of networking [54, 65, 64, 68, 66, 17]. Similar to that of previous works, in this section, a coalition formation game for network selection in vehicular network is formulated.

4.2.1 Coalition Formation Games

In the considered vehicular network, \( N \) vehicles are players in a coalition formation game. Since \( S_t^a \subseteq V \) is a coalition formed by \( |S_t^a| \) active vehicles who access RSU \( a \)'s network (\( a = a_1, \ldots, a_K \)) at time \( t \), all the coalitions in the networks form a coalition structure, referred to as a coalition partition, which is defined as

**Definition 4.1.** A coalition partition is defined as the set \( \Pi^t = \{ S_t^{a_1}, \ldots, S_t^{a_K} \} \), \( |\Pi^t| = K \) which partitions the players set \( V \) at time \( t \), i.e., \( \forall a, S_t^a \subseteq V \) are disjoint coalitions such that \( \bigcup_{a=1}^{a_K} S_t^a = V \).
In Definition 4.1, the number of coalitions is constant with respect to time if there are a fixed number of RSUs in the particular service area.

For every coalition $S^t_a \subseteq V$, the utility function $U(S^t_a)$ represents the total utility value that can be achieved in the coalition. In this section, the four utility functions given in Section 4.1.2 are considered as $U(S^t_a)$ (denoted as $U_b(S^t_a)$, $U_p(S^t_a)$, $U_{SAW}(S^t_a)$, and $U_{GRA}(S^t_a)$, respectively, for ease of identification). In the coalition formation game, $U(S^t_a)$ is a transferable utility shared by all members in the coalition $S^t_a$, which could be arbitrarily apportioned between the coalition members. Define $\phi_v(S^t_a)$ as the payoff received by a vehicle $v \in V$ after dividing the utility $U(S^t_a)$ with any payoff division rule. In this section, a fair allocation rule is used for payoff division, resulting in the payoff of any vehicle $v \in S^t_a$ to be

$$\phi_v(S^t_a) = \frac{U(S^t_a)}{|S^t_a|}$$

### 4.2.2 Hedonic Coalition Formation

Hedonic coalition formation is a popular method to study the conditions of forming coalitions in the game [65, 64, 68, 66, 17], which is used in this section. Two conditions must be satisfied if a coalition formation game is to be classified as hedonic-1) the payoff of any player depends solely on the composition of members of the coalition to which this player belongs, and 2) the coalitions form as a result of the preferences of the players over their possible coalitions’ set [65, 17]. As such, the first condition can be satisfied given the utility functions in Section 4.1.2 in the formulated game in Section 4.2.1. To satisfy the second condition, a preference function is first defined for any vehicle $v \in V$,

$$f_v(S^t_a) = \begin{cases} 
\phi_v(S^t_a) & a \in \epsilon^t_v \\
0 & \text{otherwise}
\end{cases}$$

Second, each vehicle must build preferences over its own set of possible coalitions from the available RSU set $\epsilon^t_v$. Each vehicle must be able to compare the coalitions, and order them based on which coalition the vehicle prefers to join as a member. A preference relation is defined for each vehicle to evaluate its preferences over the coalitions.

**Definition 4.2.** For any player $i \in V$, a preference relation or order $\succeq_i$ is defined as a complete, reflexive, and transitive binary relation over the set of all coalitions that player $i$ can possibly form,
i.e., the set \( \{ S^t_{a_k} \in \Pi^t : i \in S^t_{a_k} \} \).

For any vehicle \( v \in \mathcal{V} \), the preference relation of the proposed hedonic coalition game is denoted as

\[
S^t_{a_i} \succeq_v S^t_{a_j} \iff f_v(S^t_{a_i}) \geq f_v(S^t_{a_j})
\]  

(4.13)

where \( S^t_{a_i}, S^t_{a_j} \subseteq \mathcal{V}, S^t_{a_i}, S^t_{a_j} \in \Pi^t \) are any two coalitions that contain vehicle \( v \), and \( a_i, a_j \in \epsilon^t_v \).

In (4.13), \( S^t_{a_i} \succeq_v S^t_{a_j} \) indicates that player \( v \) prefers to be part of coalition \( S^t_{a_i} \), over being part of coalition \( S^t_{a_j} \), or at least \( v \) prefers both coalitions equally. Further, using the asymmetric counterpart of \( \succeq_i \), denoted by \( \succ_i \), \( S^t_{a_i} \succ_v S^t_{a_j} \), indicates that player \( v \) strictly prefers being a member of coalition \( S^t_{a_i} \) over being a member of coalition \( S^t_{a_j} \).

Therefore, the formulated hedonic coalition formation game is defined by the pair \( (\mathcal{V}, \succ) \) where \( \mathcal{V} \) is the set of players, and \( \succ \) is a profile of preferences, i.e., preference relations, \( (\succeq_1, \ldots, \succeq_N) \) defined for every player in \( \mathcal{V} \).

### 4.3 A One-Iteration Network Selection Algorithm

The remaining problem is to design an algorithm to form coalitions in the formulated game. Given a similar network switch rule is followed by all the players in the game, a typical three-phase coalition formation algorithm was proposed in these existing works [65, 64, 68]: Phase I- resource discovery and information collection, Phase II- iterative coalition formation loop, and Phase III- data transmission in the ultimate coalitions. Phase II is the most important step to form a converged final coalition partition. The convergence of a final partition \( \Pi^t_{final} \) in Phase II requires a loop, although the existing algorithms are all designed for distributed decision-making by players. The convergence time is not trivial especially involving thousands of vehicles in heterogeneous systems, which is challenging with fast-changing topologies.

Motivated by this limitation, a three-step network selection algorithm is proposed in Table 4.1. In Step 1, by updating locations of all vehicles the necessary information could be retrieved from the cloud database, used for utility calculation; in Step 2, a one-iteration coalition formation algorithm is conducted to form a final coalition partition and vehicles execute handoffs according to the results; in Step 3, data is transmitted in the networks and the cloud updates the database. The most important procedure in Algorithm 4.1 is the coalition formation algorithm in Step 2,
Table 4.1: The proposed network selection algorithm.

**Algorithm 4.1** Network selection algorithm for duration \([t, t + \Delta t]\)

<table>
<thead>
<tr>
<th>Step 1: Information update and collection</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Vehicle information update, such as ((\text{lat}, \text{lon})_v^t, v \in \mathcal{V}), etc.;</td>
</tr>
<tr>
<td>2: Information retrieval in the database;</td>
</tr>
<tr>
<td>3: Utility function (U(\cdot)) calculation;</td>
</tr>
</tbody>
</table>

**Step 2: Coalition formation**

| 4: Run **Algorithm 4.2** forming a final coalition partition; |
| 5: Notify vehicles with the best coalition to join; |
| 6: Vehicles make handoff decision and execute handoff; |

**Step 3: Data transmission and database update**

| 7: Data transmission between RSUs and vehicles; |
| 8: Information update, such as traffic load, bandwidth usage, etc. |

Table 4.2: The proposed one-iteration coalition formation algorithm.

**Algorithm 4.2** Coalition formation algorithm for duration \([t, t + \Delta t]\)

| Input: \(\mathcal{V}, A, U(\cdot)\) |
| Output: \(x^t, \Pi^t\) |

| 1: Initialization: \(\Pi^t = \{S_{a_1}^t, \ldots, S_{a_K}^t\}; S_{a_k}^t = \{\emptyset\}, a_k \in A; x_v^t = 0, v \in \mathcal{V}\); |
| 2: for \(v \in \mathcal{V}\) do |
| 3: for \(a \in A\) do |
| 4: \(f_v(S_a^t \cup \{v\}) \leftarrow U(S_a^t \cup \{v\});/*\text{following (4.11) and (4.12)*}/|
| 5: end for |
| 6: \(x_v^t = \arg\max_a f_v(S_a^t \cup \{v\})\); |
| 7: \(S_x^t = S_{x_v}^t \cup \{v\}\); |
| 8: end for |
| 9: \(S_{a_k}^t = S_{a_k}^t, a_k \in A; \Pi^t = \{S_{a_1}^t, \ldots, S_{a_K}^t\}; x_v^t = x_v^t, v \in \mathcal{V}\). |

which is illustrated in **Algorithm 4.2**. First, Nash equilibrium is introduced for understanding the convergence of **Algorithm 4.2**.

Nash equilibrium is a widely used solution in game theory. Generally speaking, Nash equilibrium is a strategy profile, where each player’s strategy is the best response to others’ strategies in the profile. A best response represents the optimal strategy of a player that maximizes the utility given others’ strategies. Since all players have their best responses, none of them have the incentive to deviate from their current strategies. For the formulated game, Nash equilibrium is defined as,

**Definition 4.3.** Denote \(x_v^t \in \mathcal{E}_v^t\) as the decision of selecting networks by any vehicle \(v \in \mathcal{V}\), and \(\mathcal{X}_v\) is the decision set of all other vehicles except \(v\). Nash equilibrium is the decision profile \(x^t = \{x_v^t, x_{v_2}^t, \ldots, x_{v_N}^t\}\), where \(v\)’s best response at time \(t\) is \(BR_v(x^t_{\mathcal{V}_v}) = x_v^t, \forall v \in \mathcal{V}\). The coalition partition \(\Pi^t = \{S_{a_1}^t, \ldots, S_{a_K}^t\}\) resulting from \(x^t\) is Nash-stable.

As a result of **Definition 4.3**, **Theorem 4.1** is proven to state the sufficient and necessary
conditions in the formulated \((V, \succ)\) game.

**Theorem 4.1.** At time \(t\) a decision profile \(x^t = \{x^t_{v_1}, x^t_{v_2}, \ldots, x^t_{v_N}\}\), resulting in a coalition partition \(\Pi^t = \{S^t_{a_1}, \ldots, S^t_{a_K}\}, \forall k \in A, S^t_{a_k} \subseteq V\), is a Nash equilibrium, if and only if

\[
f_v(S^t_{a_i}) \geq f_v(S^t_{a_j} \cup \{v\})
\]

\(\forall v \in V, v \in S^t_{a_i}, v \notin S^t_{a_j}, a_i, a_j \in \epsilon_v\) (4.14)

**Proof.** Refer to Appendix A. \(\square\)

Motivated by sequential Chinese restaurant games [78, 79], a fast algorithm is proposed in Table 4.2 to construct a Nash-stable coalition partition in the formulated \((V, \succ)\) game. Algorithm 4.2 is simple and short. For each time step, the algorithm only needs to examine the values of the preference function for all players sequentially, and then it goes to the final coalition partition after one iteration. In Theorem 4.2, the final coalition partition \(\Pi^\ast = \{S^\ast_{a_1}, \ldots, S^\ast_{a_K}\}\) obtained by Algorithm 4.2 is proven to be Nash-stable.

**Theorem 4.2.** Given the condition that \(U(S^t_a)\) is a non-increasing function in terms of \(|S^t_a|, \forall a \in A\), the output decision profile \(x^\ast = \{x^\ast_{v_1}, x^\ast_{v_2}, \ldots, x^\ast_{v_N}\}\) of Algorithm I is a Nash equilibrium and the corresponding \(\Pi^\ast = \{S^\ast_{a_1}, \ldots, S^\ast_{a_K}\}\) is Nash-stable.

**Proof.** Refer to Appendix B. \(\square\)

Since the sequence of calculating a player’s payoff is fixed in each iteration, the Nash equilibrium obtained by Algorithm 4.2 is unique. The important condition which guarantees Algorithm 4.1 is able to obtain a Nash equilibrium is the utility function \(U(\cdot)\) must be a non-increasing function in terms of the number of members in the coalition. It is easy and straightforward to prove that the four utility functions in Section 4.1.2 are all non-increasing functions, thus satisfying such condition.

### 4.4 Results and Analysis

Simulation is conducted to evaluate the performance of the proposed network selection approach.
Figure 4.1: RAN coverage of the simulation scenario.

Table 4.3: The locations and coverage ranges of BSs/APs.

<table>
<thead>
<tr>
<th>BS/AP location</th>
<th>Coverage radius (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVDO (750,1000)</td>
<td>1500</td>
</tr>
<tr>
<td>HSPA (1250,1000)</td>
<td>1500</td>
</tr>
<tr>
<td>WiMAX (750,1000)</td>
<td>1000</td>
</tr>
<tr>
<td>LTE (1250,1000)</td>
<td>1000</td>
</tr>
<tr>
<td>Wi-Fi AP1:(650,1250) AP2:(650,750)</td>
<td>250</td>
</tr>
<tr>
<td>AP3:(1000,1250) AP4:(1000,750)</td>
<td></td>
</tr>
<tr>
<td>AP5:(1250,1250) AP6:(1250,750)</td>
<td></td>
</tr>
</tbody>
</table>

4.4.1 Simulation Description

In the simulation scenario as shown in Fig. 4.1, a $2 km \times 2 km$ grid area (the square with dash line) is considered, covered by 5 types of RANs- HSPA(3G), EVDO (3G), WiMAX (4G), LTE (4G), and IEEE 802.11g (Wi-Fi). There is one base station for each cellular network, and six access points for Wi-Fi networks. The locations of the BSs/APs and the coverage range of each RAN refer to Table 4.3. Data rates and modulation and coding schemes (MCS) refer to the parameters of Table A-1-A-5 in the Appendix in [45]. The five types of RANs provide large diversity in terms of
Figure 4.2: Vehicular scenarios.

data rates and coverage ranges. Note this section does not consider differentiated data services for different RANs, and vehicles also do not have preferences on RAN types. So a decision for selecting a certain network only depends on calculated value of the defined utility function. The total simulation time is 10000 seconds, and the time step is 1 second, which means the cycle of forming a new coalition partition or centralized scheduling is 1 second. Three scenarios are considered: highway, intersection
and Manhattan, shown in Fig. 4.2. In each scenario, the total number of vehicles is 100. Each road has two bi-directional lanes, shown as the gray lines in Fig. 4.2. The width of lanes and size of vehicles are neglected in the simulation. In the highway and intersection scenarios, all vehicles keep a constant velocity towards one direction, indicating a constant traffic flow on the road. In the Manhattan scenario, each vehicle uniformly chooses a velocity below a maximum speed. The probability of going straight is 0.5 and taking a left or right is 0.25 at each intersection. The simulation time is sufficiently long to obtain performance statistics, especially for the Manhattan scenario with random turning for vehicles at each intersection. In all scenarios, if a vehicle moves out of the considered square region, this vehicle is assumed to re-enter into the region from the other end of that road for simplicity. Since the proposed game-based scheme is focused on selecting the best network out of all available networks, a vehicle can see at least one available network in the considered region. The methods of extending coverage, such as relaying or using ad-hoc mode, is beyond the scope of this dissertation.

The performance metric evaluated in this section needed to be defined beforehand is the fairness, given as

\[ \gamma = \frac{\left( \sum_{v \in V} \theta_v \right)^2}{|V| \cdot \sum_{v \in V} (\theta_v)^2} \]  

(4.15)

The fairness \( \gamma \) defined in (4.15) follows Jain’s definition of fairness [31]. Here, \( \theta_v \) represents vehicle \( v \)’s throughput, indicating the fairness is throughput-based.

### 4.4.2 Comparisons with Centralized Optimal Algorithm in [10]

The centralized optimal algorithm in [10] is selected to as a comparison to the proposed coalition formation game. The original optimization in [10] considers a weighted sum of three attributes, which are spectral efficiency, fairness, and power consumption. Instead of directly comparing with an optimal multiple-attribute utility, this subsection considers optimal algorithms in terms of a single attribute, in order to avoid the weighting effects. In particular, spectral efficiency and fairness using centralized optimization in [10] are considered, respectively. For both optimization, three minimum data rate requirements for vehicles’ quality of service (512 Kbps, 768 Kbps, 1024 Kbps) are measured in the simulation, similar to [10]. The average velocity in all cases is 13.4 m/s. The system throughput and fairness comparisons are shown in Fig. 4.3. As shown, there is a tradeoff between throughput and fairness. Once one vehicle sees good signal quality, it will try to get as
much bandwidth as possible for data transmission assuming spectral efficiency optimization, which leaves other vehicles in the same network with less bandwidth. Likewise, with fairness optimization, the bandwidth is allocated fairly, sacrificing some vehicle’s large data rates in good coverage areas. In Fig. 4.3, the results of the coalition formation game show that the performance is neither biased on spectral efficiency nor fairness optimization, rather it naturally captures the tradeoff between the two. The reason is the four utility functions in Section 4.1.2 contain throughput related metrics, and the utility division rule in (4.11) also guarantees fairness among vehicles. In all three scenarios, the performance has similar features for both optimization and the coalition formation game in Fig. 4.3.

The fairness metric used by the centralized optimal algorithm is defined explicitly and quantitatively in [10] (the combined utility function (equation (11a)) includes fairness functions defined by (4) and (5) in [10]). Furthermore, weights are needed to be determined if using MADM methods, e.g. AHP method. Compared to such algorithm, the proposed game-based scheme does not include fairness functions in utility functions. The fairness is embedded in the payoff division rule of coalition formation game in (4.11). The proposed game-based scheme provides a framework with
the feature of embedded fairness, and readers have flexibility to define their own utility functions for coalitions according to their own demands or requirements. The framework always assures fairness of host utility defined by users. The four throughput-based utility functions in this chapter are used as an example to evaluate fairness performance.

4.4.3 Comparisons among Different Utility Functions

Fig. 4.4 shows throughput, delay, and fairness performance among four utility functions discussed in Section 4.1.2 versus arrival bit rate. The average velocity in all cases is 13.4 m/s. The tradeoff factor in (4.6) is $\beta = 0.5$, and the weights in (4.9) and (4.10) are $w_\eta = w_\tau = 0.5$. In Fig. 4.4, the throughput linearly increases first and then saturates when the arrival bit rate per vehicle increases and reaches 2 Mbps. In all three scenarios, the throughput of the utility function $U_b$ is slightly higher than the other three. Similarly, the delay performance of the utility function, $U_b$,
is also slightly better than the other three in all scenarios. This is reasonable because the utility function $U_b$ tries to select the network with the most available bandwidth, so that the delay could be reduced with higher bandwidth. The price for getting higher throughput and lower delay with $U_b$ is to sacrifice fairness by $5 \sim 10\%$ as shown in Fig. 4.4, which is approximately the range of improvement for the throughput. $U_{SAW}$ and $U_{GRA}$ have very close performance in Fig. 4.4, and $U_p$ gets slightly better performance in terms of fairness. The fairness around an arrival bit rate of $1 \, Mbps$ is higher than that with saturation, because non-saturation data for all vehicles’ packets is more likely to be sent out instead of being backlogged. With more complex road topology and dynamic movement pattern in the Manhattan scenario, the throughput deteriorates by up to about $5\%$ and $15\%$ compared to the highway and intersection scenarios, respectively. The intersection scenario could yield the highest throughput because vehicles have a chance to see more networks compared with the highway scenario but with constant movement compared with the Manhattan scenario. Fig. 4.5 shows throughput, delay, and fairness performance vs. average velocity. The arrival bit rate per vehicle is set as $1000 \, Kbps$ in the simulation. In Fig. 4.5, with the increase of the average velocity, the throughput is nearly constant, the delay decreases, and the fairness slightly decreases for all cases. The delay’s change is more significant in the Manhattan scenario, which is because vehicles have a high probability of traversing high bandwidth networks with a high velocity in a more dynamic road topology.

4.4.4 Comparison with Conventional Coalition Formation Algorithms in [65]

Fig. 4.6 compares the average running time for forming an ultimate coalition partition in one coalition formation cycle (1 second) between the proposed algorithm and conventional algorithm in [65]. The two algorithms both run on the same virtual machine with $3.4 \, GHz$ CPU and $10.5 \, GB$ memory. The utility function $U_b$ is used. In Fig. 4.6, with the increase of the number of vehicles, the running time for both algorithms linearly increases, but the slope for the conventional algorithm is about 8 times steeper than the proposed algorithm. Hence, in one formation cycle, the running time of the conventional algorithm is 8 times as that of the proposed algorithm, leaving less time for data transmission if being implemented in practice. If the overhead of the coalition formation is too large, the coalition formation game is not usable in the real world because after finishing computation for
coalition partition, many vehicles will have already moved to other locations, resulting in the final coalition partition being outdated. From the point of view of network scalability, under the same hardware and software conditions, with the same running time limit for formation, the proposed algorithm actually is able to handle a network whose size is 8 times that of the one handled by the conventional coalition formation algorithm.

The running time of optimization on spectral efficiency in Chapter 4.4.2 is much longer. AMPL software is used for linear programming in the same virtual machine with the same simulation setting. When the number of vehicles is 100, the running time for one cycle is already about 4 seconds (the total running time for 10000-second simulation is more than 10 hours), which actually is beyond the scheduling cycle duration itself.

Figure 4.6: Average running time for one scheduling interval of proposed coalition formation algorithm and conventional algorithm.
Chapter 5

Network Coding based Data Forwarding

In the past decade, numerous works have proposed P2P-based collaboration via V2V communications for content distribution in vehicular networks [50, 51]. To reduce high overhead and adapt to error-prone channels, network coding has been adopted in many existing works [4, 41, 94], and has provided significant throughput improvement compared to P2P solutions. Both recent P2P and network coding approaches mostly consider common content distribution among vehicles, however, such solutions cannot be automatically applied to multiple unicast sessions from infrastructure to vehicles on the move. Therefore, more generic and scalable framework, involving both core networks and edge networks, are still open questions for supporting broadband media access for vehicles. Furthermore, an efficient grouping strategy of unicasts for coding is a difficult research problem due to dynamic network topology on the road.

Considering multiple unicasts from Internet to vehicles, this chapter proposes an inter-session network coding scheme for efficient data forwarding. To overcome the difficulties of operating inter-session for dynamic topologies, a conditionally optimal grouping algorithm is proposed. The rest of the chapter is organized as follows. Section 5.1 introduces network model in an extended butterfly network. Section 5.2 introduces inter-session network coding operations and key assumptions. Section 5.3 formulates the considered problem in vehicular networks. Section 5.4 proposes a dynamic grouping algorithm for inter-session network coding. Section 5.5 analyzes the simulation
results and discusses the limitations.

5.1 Network Model

There are two classes of network coding according to the way packets are grouped for encoding. Intra-session network coding applies to a single communication session, i.e. unicast communication to one sink node or multicast of common information to multiple sink nodes. Fig. 5.1 shows a single source multicasts to two destinations using intra-session network coding. The source node sends two packets along different paths and the intermediate relaying node conducts coding (binary sum in the figure) if it sees these two packets. After destination receives coded packets, it decodes using the native packets from the other links (commonly referred to as remedy links in network coding literature). On these remedy links there are no other operations of packets besides forwarding only [27]. Inter-session network coding has a similar coding and decoding operation as the intra-session network coding, shown in Fig. 5.2, except native packets for coding are from different sessions or source nodes. Inter-session network coding is more complicated especially for
Figure 5.3: An example of inter-session network coding.

5.1.1 Inter-session Network Coding Operation

Fig. 5.3 shows an example of inter-session network coding, consisting of two unicast sessions between two sources and two destinations. \( S_x \) are application servers. \( R_x \) and \( P_x \) both refer to routers - the different symbols are only used to indicate their different roles when network coding is applied. Sessions from source nodes \( S_1 \) and \( S_2 \) are for vehicles \( V_1 \) and \( V_2 \) respectively. \( D_1 \) and \( D_2 \) are neighboring BSs in the edge. The solid lines are wired links with a same bandwidth \( r \). The dashed lines towards vehicles are wireless links with bandwidth \( r \).

At time \( t \), assume \( V_1 \) and \( V_2 \) are connected with \( D_1 \) and \( D_2 \) respectively. Packets from \( S_1 \) are forwarded to \( V_1 \) following the path \( S_1 - P_1 - D_1 - V_1 \), and packets from \( S_2 \) are forwarded to \( V_2 \) following the path \( S_2 - P_2 - P_3 - D_2 - V_2 \). Since the two vehicles are mobile, at time \( t+1 \), they move out of the previous BSs’ coverage and are connected to the other BS, e.g. \( V_1 \) is connected to \( D_2 \) and \( V_2 \) is connected to \( D_1 \). With traditional routing there are two possible paths for each vehicle. For \( V_1 \) the two paths are \( S_1 - R_1 - R_2 - D_2 - V_1 \) and \( S_1 - P_1 - D_1 - R_2 - D_2 - V_1 \). For \( V_2 \) the two paths are \( S_2 - R_1 - R_2 - D_1 - V_2 \) and \( S_2 - P_2 - P_3 - D_2 - R_2 - D_1 - V_2 \).

At this point, either the link \( R_1 - R_2 \) or the links \( R_2 - D_1 \) and \( R_2 - D_2 \) carry packets from both sessions, forming a bottleneck due to the limited bandwidth. The bottleneck results in decreasing throughput and increasing end-to-end packet delay for vehicles at time \( t+1 \).

Inter-session network coding is introduced to overcome the above problem. On the router \( R_1 \), inter-session network coding is operated. In each time period, the cloud computes the network coding grouping for each router. After \( R_1 \) receives grouping results from the cloud, it conducts network coding for the two sessions, where binary-sum XOR (exclusive or) coding across unicasts is
used. The router $R_2$ forwards coded packets to BSs. The link $R_1 - R_2$ is a bottleneck link in the above case with traditional routing, while bandwidth usage is improved using network coding in Fig. 5.3. The BSs $D_1$ and $D_2$ decode packets using native packets from forwarding links $S_1 - P_1 - D_1$ and $S_2 - P_2 - P_3 - D_2$. These packets carried by forwarding links are used for decoding coded packets. Since binary-sum coding is used in this chapter, the destination only needs to operate XOR using a coded packet and a native packet for decoding. The ratio of forwarding links is defined as the ratio of the number of forwarding links for a source node to the number of destination nodes, e.g. 0.5 for both sessions in Fig. 5.3. There are other types of network coding, such as random linear network coding, which does not require native packets for decoding but only sufficient number of coded packets. For simplicity, this chapter discusses XOR coding only.

The benefit of using inter-session network coding in the example of Fig. 5.3 is the ability to guarantee stable throughput and packet delay in the presence of mobility. Such benefit comes from encoding multiple packets into one packet, resulting in saving bandwidth on bottleneck links. The requirements of implementing such topology are 1) common bottleneck links for neighboring BSs and alternative forwarding links for each BS, 2) corresponding bandwidth for links carrying encoded packets and links carrying native packets, 3) disjoint multipath routing in the network. A more complicated model based on the example shown in Fig. 5.3 is introduced in the next section.

5.1.2 An Extended Butterfly Network Model

The butterfly network is widely studied in the literature, see e.g. [49, 42]. An extended butterfly network is considered in this section, shown in Fig. 5.4. The network is represented by a hypergraph $G = (N, \mathcal{H})$, where $N$ is the set of nodes and $\mathcal{H}$ is the set of hyperarcs. The set of source nodes is represented as $S = \{s_1, s_2, \ldots, s_M\}$ in the core network; the set of destination nodes is represented as $V = \{v_1, v_2, \ldots, v_M\}$, which are vehicles on the move in the edge. The set of RSUs is represented as $A = \{a_1, a_2, \ldots, a_K\}$, which is a part of both the core network and the edge network. The core network contains two relay nodes $R = \{r_1, r_2\}$, forming a bottleneck link. Multiple forwarding links (simply forwarding without coding) from sources nodes to RSUs also exist in the core network. A hyperarc is a pair $(i, \mathcal{J})$, where $i$ is the start node of multiple directed arcs, $i \in N \setminus V$, $i \notin \mathcal{J}$ and $\mathcal{J}$ is the set of end nodes, $\mathcal{J} \subset N \setminus S$, $N = S \cup V \cup A \cup R$. An arc is a pair $(i, j)$, $i \neq j$, $j \in \mathcal{J}$ which is a wired link in the core network and wireless link in the edge network. A link $(i, j)$ has the capacity of $c_{i,j}$, $i \in N \setminus V$, $i \notin \mathcal{J}$, $i \neq j$, $j \in \mathcal{J}$. The solid and dash-dotted lines with
The edge network consists of $M$ vehicles, which are in the service areas of $K$ RSUs, which can potentially be of different types of RANs. Each vehicle is equipped with either a single radio or multi-radio devices to connect to the corresponding type of RANs. All vehicles are equipped with GPS devices. At time $t$ vehicle $v$’s GPS coordinate is $(\text{lat}, \text{lon})^t_v$, $v \in \mathcal{V}$, where lat and lon are the latitude and the longitude, respectively. RSU $a \in \mathcal{A}$ has a coverage distance $D_a$. Each RSU $a \in \mathcal{A}$ offers a total bandwidth $B_a$ and a maximum link transmission capacity $\mu_a$, in bits per second. At time $t$, the RSU’s availability is associated with vehicle’s location. The availability of RSU $a$ is given
\[\epsilon_t^v(a) = 1_{D_a}((\text{lat}, \text{lon})_a^v) = \begin{cases} 
1 & \text{if } d_t^a \leq D_a \\
0 & \text{if } d_t^a > D_a \end{cases} \quad a \in A, v \in V \] (5.1)

In (5.1), \(d_t^a = \sqrt{(\text{lat}_v^t - \text{lat}_a)(\text{lon}_v^t - \text{lon}_a)^2}\) is the distance between vehicle \(v\)'s current location and RSU \(a\)'s location, where \((\text{lat}_a, \text{lon}_a)\) is the RSU \(a\)'s location. Therefore, vehicle \(v\) is able to get a set of RSU’s availability at time \(t\), which could be represented as \(\epsilon_t^v = \{a_i : \epsilon_t^v(a_i) = 1, a_i \in A\}\). At time \(t\), each vehicle \(v \in V\) makes a decision to select a best network and access the selected network. Each RSU \(a \in A\) serves a set of vehicles \(E_t^a \subseteq V\), consisting of \(|E_t^a|\) active vehicles who access RSU \(a\)'s network. At a given time, a vehicle is only allowed to use one single network interface, so that \(E_t^i \cap E_t^j = \emptyset, i, j \in A, i \neq j\). The edge network partition is denoted as \(\Pi_t = \{E_t^1, \ldots, E_t^K\}\).

Hence, at time \(t\) the set of hyperarcs in the edge network is \(\mathbb{H}_{\text{edge}}^t = (i, J_t^i), i \in A\), where \(J_t^i = E_t^i\).

In the core network, there are \(M\) unicast sessions shown in Fig. 5.4. There are \(M\) source processes originating at the source node \(s_1, s_2, \ldots, s_M\), respectively, and destinated at the sink node \(v_1, v_2, \ldots, v_M\), respectively. For notational convenience, \(X_{i,i} = 1, \ldots, M\) refers to a single packet (or a chunk segmented from an incoming message) of the source process on source node \(s_i\); \(Y_{i,j}\) refers to a single packet of the process on the arc \((i, j)\), and \(Z_{i,i}, i = 1, \ldots, M\) refers to a single packet of the output process on destination node \(v_i\). \(F(\cdot)\) is defined as the packet operation function on an arc. The data rates of the source process on source node \(s_i\) and the process on the arc \((i, j)\) are represented as \(x_i\) and \(y_{i,j}\), respectively. The set of hyperarcs in the core network is denoted as \(\mathbb{H}_{\text{core}} = (i, J_i), i \in S \cup R\), where \(J_i \subseteq r_1 \cup A\), \(i \in S\); \(J_{r_1} = r_2\); \(J_{r_2} = A\).

### 5.2 Assumptions

Due to the fast-changing topology in the edge network, the grouping of sessions must be dynamic instead of being constant in prior works. This chapter proposes a dynamic grouping strategy in terms of a snapshot view. During a short time period, traffic flows are considered to be static so that the grouping is constant. In a next time period, grouping needs to be adjusted according to changed flows. In the long term, the grouping is time-variant. For network coding to gain the best performance, optimal session grouping strategies must be computed by cloud servers based on knowledge of the edge network topology. The grouping strategies are computed considering the link...
topologies of RSUs, and the current served sessions at routers and RSUs.

Federated networks are assumed able to control packet forwarding from end to end based on agreements among operators. Since today’s IP core networks are based on Border Gateway Protocol (BGP), control of packet forwarding is a non-trivial task on legacy switches/routers. Emerging software-defined networking (SDN) techniques are expected to provide more flexibility of manipulating forwarding rules. While the scheme in this chapter is orthogonal to the underlying SDN approach, the chapter bases its architectural discussion on OpenFlow [57]. The core network is assumed to consist of a set of OpenFlow enabled switches controlled by remote controller(s).

5.3 Problem Formulation

The formulated problem is a nonlinear optimization problem of maximizing a utility function by optimally choosing data rates of unicast flows $x^t_s$, packet operation function $F(\cdot)$, and the edge network partition $\Pi^t$ in the network $G$, given as below.

Problem 5.1.

\[
\begin{align*}
\text{maximize} & \quad \sum_{s \in S} U_s(x^t_s) \\
\text{subject to} & \quad y^t_{s_1, r_1} \leq x^t_s, \quad \forall s_i \in S, \\
& \quad y^t_{s_1, a_j} \leq x^t_{s_1}, \quad \forall s_i \in S, a_j \in J_{s_1}, \\
& \quad Y^t_{r_1, r_2} = F(X^t_1, \ldots, X^t_M), \\
& \quad y^t_{s_i, r_1} \leq c_{s_i, r_1}, \quad \forall s_i \in S, \\
& \quad y^t_{s_i, a_j} \leq c_{s_i, a_j}, \quad \forall s_i \in S, a_j \in J_{s_i}, \\
& \quad y^t_{r_1, r_2} \leq c_{r_1, r_2}, \\
& \quad y^t_{r_2, a_i} \leq c_{r_2, a_i}, \quad \forall a_i \in A, \\
& \quad \sum_{j \in E_{a_i}} y^t_{a_i, v_j} \leq B_{a_i}, \quad \forall a_i \in A, E^t_{a_i} \in \Pi^t.
\end{align*}
\]

In Problem 5.1, the first two constraints are the data rate constraints for outgoing arcs on each source node $s_i \in S$. The third constraint is the packet operation on the arc $(r_1, r_2)$. The next four constraints are the capacity constraints for each arc in the core network. The last constraint is the capacity constraints for each RSU’s network on the edge. The last constraint with respect to $\Pi^t$
is another major research problem on resource allocation [10] or network selection [68], so that this chapter does not emphasize this part.

The proposed model is generic and under certain conditions it can transform to models studied in the literature, e.g. two forwarding link in [49] and \( N - 1 \) forwarding links in [42]. The case in [49] is the base of pairwise network coding, and the case in [42] is the ideal case in which the advantage of network coding grows proportionally as \( \Theta(|\mathcal{N}|) \) compared to routing. Both the cases do not consider the fast-changing topology on the edge, resulting in the impossibility of the requirement of forwarding links for decoding in terms of time. Generally the topology of forwarding links is not time variant in the core network, so \( \mathcal{H}_{core} \) is assumed to be constant in the proposed model. With the three unknown arguments \( x_t^s, F(\cdot), \Pi^t \), it is non-trivial to solve Problem 5.1. This paper emphasizes the design of \( F(\cdot) \) to optimize the utility function, given the data rates of source processes \( x_t^s \) and the fast-changing edge network partition \( \Pi^t \).

5.4 Proposed Dynamic Grouping Algorithm for Inter-Session Network Coding

The remaining problem is to design an algorithm to make optimal grouping strategies for routers. In the considered network \( \mathcal{G} = (\mathcal{N}, \mathcal{H}) \) shown in Fig. 5.4, the relay node \( r_1 \) uses XOR coding across multiple sessions according to the grouping strategy provided by the cloud. The ideal case is to group all the sessions at the node \( r_1 \), resulting in the coding gain as \( \Theta(|\mathcal{N}|) \) against routing [42]. Such gain comes from the ability of carrying packet information from multiple sessions in one encoded packet. Although the constant forwarding links and the fast-changing topology on the edge make the ideal case impossible, the objective is to group as many sessions as possible, in order to obtain a maximum utility in Problem 5.1.

5.4.1 Algorithm Description

A three-step inter-session network coding algorithm is proposed in Table 5.1. In Step 1, by updating locations of all vehicles, serving vehicles of RSUs, and traffic on the core the necessary information is used for make grouping strategies; in Step 2, the cloud makes an optimal grouping strategies for network coding, and routers and RSUs are notified by such grouping strategies; in


Table 5.1: The proposed inter-session network coding algorithm.

<table>
<thead>
<tr>
<th>Algorithm 5.1</th>
<th>Network coding algorithm for duration $[t, t + \Delta t]$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1: Information update and collection</strong></td>
<td>1: Vehicle information update, such as $(\text{lat, lon})_v^t$, $v \in V$, etc.;</td>
</tr>
<tr>
<td></td>
<td>2: RSU information update, such as $\Pi^t$;</td>
</tr>
<tr>
<td></td>
<td>3: Traffic information of core networks update, such as $x_s^t$;</td>
</tr>
<tr>
<td><strong>Step 2: Grouping strategies for network coding</strong></td>
<td>4: Run Algorithm 5.2 to form grouping strategies for network coding;</td>
</tr>
<tr>
<td></td>
<td>5: Notify routers and RSUs with the optimal grouping strategies;</td>
</tr>
<tr>
<td><strong>Step 3: Encoding and decoding</strong></td>
<td>6: Routers operate inter-session network coding according to the strategies;</td>
</tr>
<tr>
<td></td>
<td>7: RSUs decode packets and deliver original packets to corresponding vehicles.</td>
</tr>
</tbody>
</table>

**Step 3**, according to the grouping strategies, routers and RSUs operate encoding and decoding respectively, and original packets are sent to corresponding vehicles after decoding at the RSUs. The most important procedure in Algorithm 5.1 is the grouping strategies in **Step 2**, which is illustrated in Algorithm 5.2.

In Algorithm 5.2, $\Omega$ is an output of grouping strategies which contains multiple sets of grouped vehicles. In line 2-9 of Algorithm 5.2, the unicasts for vehicles are examined to keep eligible ones for network coding in the set $\Lambda_a^t$, by excluding those that have forwarding links to the associated RSUs. The eligible vehicles are candidates for the elements of grouping. $\Phi_{RSU}$ is a set of RSUs who have non-empty sets of eligible vehicles for network coding. The loop from line 10 to line 38 is to search for groups in terms of how many vehicles should be grouped together for encoding. As such, the examination starts from the maximum number of possible grouping is $|\Phi_{RSU}|$, and descends the number of grouping for each iteration. The loop from line 14 to line 33 is to find the most groups with the examined number of grouping. The set $\Gamma_a^t$ keeps a set of eligible vehicles after $\Lambda_a^t$ intersects with forwarding links of other RSUs involved in network coding, by excluding those that cannot use forwarding links to decode. In line 34-37, the found groups are included in the output set $\Omega$ as elements and $\Lambda_a^t$ exclude corresponding vehicles in the found groups. Afterwards, the algorithm continues with the next examined number of grouping.

In order to prove the output of Algorithm 5.2 is optimal, Lemma 5.1 and Condition 5.1 are introduced first.

**Lemma 5.1.** In Algorithm 5.2, for any give number of grouped vehicles $n$, $n = 1, \ldots, |\Phi_{RSU}|$, the number of the found grouping set $|\Omega_n|$ is the most among all subsets $\{\Psi_n | \forall \Psi_n \in \Phi_{RSU}, |\Phi_n| = n\}$ of $\Phi_{RSU}$.
Table 5.2: The proposed dynamic grouping algorithm for network coding.

**Algorithm 5.2** Dynamic grouping algorithm for duration \([t, t + \Delta t]\)

| Input: \(S, A, R, V, \Pi_t\) |
| Output: \(\Omega\) |
| 1: Initialization: \(\Phi_{RSU} = \emptyset, \Omega = \emptyset\) |
| 2: for \(a \in A\) do |
| 3: \(\Lambda_t^a = E_t^a \{ v_i | s_i \in S, a \in J_t^i, v_i \in E_t^i \} \) |
| 4: if \(\Lambda_t^a \neq \emptyset\) do |
| 5: \(\Phi_{RSU} = \Phi_{RSU} \cup \{a\} \) |
| 6: else |
| 7: continue |
| 8: end if |
| 9: end for |
| 10: for \(n = |\Phi_{RSU}|; n \geq 2; n = n - 1\) do |
| 11: \(tmp_{min} = \infty\) |
| 12: \(tmp_{max} = 0\) |
| 13: \(\Omega_n = \emptyset\) |
| 14: for \(\Psi \in \{\Psi_n | \forall \Psi_n \subseteq \Phi_{RSU}, |\Psi_n| = n\}\) do |
| 15: for \(a \in \Psi\) do |
| 16: for \(b \in \Psi \setminus \{a\}\) do |
| 17: \(\Gamma_t^a = \Lambda_t^a \cap \{ v_i | s_i \in S, b \in J_t^i \} \) |
| 18: end for |
| 19: if \(\Gamma_t^a = \emptyset\) do |
| 20: No grouping found; continue with next \(\Psi\); |
| 21: end if |
| 22: if \(tmp_{min} > |\Gamma_t^a|\) do |
| 23: \(tmp_{min} = |\Gamma_t^a|\) |
| 24: end if |
| 25: end for |
| 26: if \(tmp_{min} > tmp_{max}\) |
| 27: \(tmp_{max} = tmp_{min}\) |
| 28: \(\Omega_n = \emptyset\) |
| 29: for \(m = 1; m \leq tmp_{max}; m = m + 1\) do |
| 30: \(\Omega_n = \Omega_n \cup \{ (w^m_i, w^m_i, \ldots, w^m_i) | z_i \in \Psi, w^m_i \in \Gamma_t^i, i = 1, 2, \ldots, |\Psi| \}\) |
| 31: end for |
| 32: end if |
| 33: end for |
| 34: if \(\Omega_n \neq \emptyset\) do |
| 35: \(\Omega = \Omega \cup \Omega_n\) |
| 36: \(\Lambda_t^a = \Lambda_t^a \setminus \{ \Lambda_t^i \cap \Omega_n \}, a \in \Phi_{RSU}\) |
| 37: end if |
| 38: end for |

**Proof.** Refer to Appendix C. \(\square\)

**Condition 5.1.** With the found maximum number of a grouping set \(|\Omega_n|\), the element selection for a group \((w^m_{z_1}, w^m_{z_2}, \ldots, w^m_{z_{|\Psi|}})\), \(z_i \in \Psi, w^m_{z_i} \in \Gamma_t^i, i = 1, 2, \ldots, |\Psi|\), follows a pre-defined rule.
The pre-defined rule can be any rule defining the order for selecting elements from all eligible
vehicles in a specific RSU’s serving list. For example, the algorithm always selects the first eligible
vehicle or the vehicle with the smallest identity number in the list.

**Theorem 5.1.** Given Condition 5.1 and Algorithm 5.2, let $\Upsilon = \mathbb{V} \setminus \{v_\Omega | v_\Omega \in \text{certain group in } \Omega_n, n = 2, \ldots, |\Phi_{RSU}|\}$ be the remaining vehicles which are not involved in network coding, whose packets
are forwarded directly. The ratio $\left( \sum_{\Omega_n \neq \emptyset} |\Omega_n| \cdot n + |\Upsilon| \right) / \left( \sum_{\Omega_n \neq \emptyset} |\Omega_n| + |\Upsilon| \right)$ is the largest with
the output against all other possible grouping sets in Algorithm 5.2.

*Proof.* Refer to Appendix D. □

In the ratio of Theorem 5.1, the denominator represents how many transmissions per unit
time and the numerator represents how many transmissions per unit time. Hence, from Theorem 5.1 the highest throughput and the most gain of inter-session network coding
can be achieved by Algorithm 5.2. Note Theorem 5.1 holds under Condition 5.1. Under Condition
5.1 and Theorem 5.1, the computation complexity is $O \left( |\mathbb{A}|^2 \cdot 2^{|\mathbb{A}|} + |\mathbb{A}| \cdot |\mathbb{V}| \right)$. In the complexity, $2^{|\mathbb{A}|}$ comes from the sum of combinations in line14 of Algorithm 5.2, and $|\mathbb{A}|^2$ is from line15 and
line16. Had a globally optimal solution been adopted instead, exhaustive combinations of
group elements had to be searched for all $n$ in Algorithm 5.2. The complexity would have become
$O(|\mathbb{V}|^O(|\mathbb{A}|)) \cdot O(|\mathbb{V}|)^O(|\mathbb{A}|)$, where the number of iteration is $O(|\mathbb{A}|) - 1$. The running time on a
modern PC could be considered as infinity with such complexity in the algorithm. Instead, a con-
ditionally optimal algorithm is proposed in this section. The design of a feasible, globally optimal
algorithm is still an open question.

### 5.4.2 Technical Framework in Practice

This subsection discusses a technical framework of applying the proposed algorithm in prac-
tice. Since all nodes in the network are assumed to be OpenFlow-enabled, the framework is designed
in the OpenFlow domain. Each switch maintains a flow table, which performs packet lookup and
forwarding. The flow table contains a set of flow entries defining header fields (from Layer 1 to Layer
4), activity counters, and a set of zero or more actions to apply to matching packets. All packets
processed by the switch are compared against the flow table. If a matching entry is found, any
actions for that entry are performed on the packet, but if no match is found, the packet is forwarded

50
to the controller. The controller is responsible for determining how to handle packets without valid flow entries, and it manages the switch flow table by adding and removing flow entries.

Fig. 5.5 illustrates the technical framework of applying the proposed algorithm in practice. Suppose there are newly incoming packets from four flows arriving at Switch 1, but Switch 1 does not have any valid flow entries in its flow table. So Switch 1 forwards the packets to Controller 1, and Controller 1 provides the cloud infrastructure information from Layer 1 to Layer 4 of each flow by parsing packet headers. The cloud uses the received flow information, along with the flow table database and network topology database maintained in the cloud, to perform algorithm computing (Algorithm 5.1 and 5.2 in Sec. 5.4.1). Once computing is finished, the coding set database will be updated according to the output, and Controller 1 will be notified of the results. Controller 1 will inform Switch 1 of inserting flow entries and actions in the flow table. In this case, packets from Flow 2 and 3 will be directly forwarded to port 2 and 3, respectively. Flow 1 and 4 need to be
encoded first and the coded packets need to be forwarded to both port 2 and 3. Since a common action of a flow entry is to forward to specific port(s), the entries for Flow 1 and 4 are inserted as to forward to a virtual port 1 (vport 1). Virtual ports are easy to be created and removed using today’s software-based switch, e.g. Open vSwitch [77]. Network coding programs could listen to vport 1 and then perform encoding once receiving all corresponding packets. The coded packets can be sent over vport 1, and Switch 1 will forward these coded packets to port 2 and 3 by looking up the flow table. The usage of virtual ports is one option, while other means could be also deployed in practice.

After multiple hops, Switch 4 in the edge receives these coded packets from port 1; however, it cannot find any matched flow entries for these packets, so it forwards these packets to Controller 2. Controller 2 then passes the header information of the coded packet to the cloud. The cloud returns the decoding instructions by looking up the coding set database and flow table database. The decoding ability of Switch 4 was already considered when computing the algorithms for Switch 1, so the decoding information should be accurate. Switch 4 decodes packets for Flow 1 and 4 using native packets received from other ports, e.g. port 2 and port 3, respectively. After decoding, packets will be forwarded to BSs and then corresponding vehicles.

Since flow topology is dynamic, switches communicate with controllers in a periodical basis to update the existing flow entries, besides to request actions for newly incoming flows. Flow table database, network topology database, and coding set database are also updated periodically once there is any change to the corresponding parties.

### 5.5 Results and Analysis

The performance of the proposed network coding scheme is evaluated and analyzed through results from extensive simulation.

#### 5.5.1 Simulation Description

The network topology in the simulation follows Fig. 4.1. The locations of the BSs/APs and the coverage range of each RAN refer to Table 4.3. A Manhattan scenario is used on the edge, shown in Fig. 4.2. In the Manhattan scenario, each vehicle uniformly chooses a velocity below a maximum speed. The density of vehicles in the considered region is up to 40 [16]. The probability
of going straight is 0.5 and taking a left or right is 0.25 each at each intersection. The network selection algorithm uses available bandwidth as a utility function in a game-based scheme [92]. In the core network, there are 2 relay nodes and multiple source nodes. The number of source nodes is equal to the number of vehicles. From each source node to the corresponding vehicle UDP traffic (constant bit rate) or TCP traffic is considered, with the traffic arrival rate 2 \( Mb/s \) at the source. Each wired link on the core has a bandwidth capacity 50 \( Mbps \) (This value is chosen for simplicity of creating a bottleneck link with limited number of vehicles in the simulation. The conclusions in this chapter do not depend on this value and could be applied to any value, e.g. 10 \( Gbps \), if the bottleneck link satisfies a similar feature). The size of queuing buffer on all the nodes is 50 \( Mbits \).

The ratio of forwarding links is set as 0.5. The total simulation time is 500 seconds, and the time step is 1 second, which means the cycle of forming groups for network coding is 1 second. The simulation code is written in JAVA. The code was developed and run in user space under Ubuntu 14.04.

Two baseline approaches are considered in this section- 1) conventional routing: the packets in the bottleneck link are directly routed; 2) modified pairwise network coding: based on [49] and Algorithm 5.2 is used to form groups consisted of pairwise vehicles for network coding (line10 in Table 5.2: \( n \) is only set to 2). Pairwise grouping still follows the algorithm, but every group only groups two sessions. The reasons of choosing pairwise grouping as a baseline because it has the minimal time cost for computing grouping and encoding/decoding packets.
5.5.2 TCP or UDP?

Fig. 5.6 and Fig. 5.7 show average throughput per vehicle and average delay per packet for UDP packets, and TCP packets, respectively. The traffic generate rate at each source node is 2 Mbps. In both figures, throughput with routing, proposed network coding and pairwise coding has no significant difference when the number of vehicles changes. This is because that the throughput actually highly depends on packet arrival rate when the bottleneck link is not congested. When the bottleneck link forwards more packets, despite that each coded packet can contain more information, coordination time by encoding process on relaying nodes sacrifice throughput compared to simply forwarding packet with routing. Although the throughput is close, there exist apparent differences for delay among these three cases, for both UDP and TCP packets. For UDP packets in Fig. 5.6,
when the number of vehicles is 10, in all the three cases the delay is small enough (so not visible in the figure). For UDP, as the number of vehicles increases, the delay with routing goes up quickly, which is larger than the delay with proposed network coding by 3090.38%, 221.59% and 126.02%, for 20 vehicles, 30 vehicle, and 40 vehicles, respectively; and larger than the delay with pairwise network coding by 389.95%, 108.32% and 77.77%, respectively. The reason is, given the congested link in terms of the number of packets traversed, the proposed network coding is able to encode as many packets as possible, which improves the usage efficiency of bandwidth. For TCP packets, before the bottleneck link is congested, delay with network coding is much higher than that with routing, which expresses effects, caused by network coding, towards in-order and reliable protocols like TCP. For network coding, a relay node must check its buffer if packets belonging to eligible vehicles (the set calculated by the proposed grouping algorithm) are ready for encoding. If all corresponding packets are waiting in the buffer, the relay node will go ahead with operating encoding; otherwise encoding process will be deferred until all corresponding packets arrive in the buffer. TCP’s sliding window gets affected by some lagged packets using network coding, and thus the average delay becomes much higher than using routing. Such effect becomes alleviated as the bottleneck link being congested, e.g. when there are 30 and 40 vehicles shown in Fig. 5.7, since packets are waiting back to back in the buffer. But network coding does not yield much advantage on delay performance for TCP packets when the link becomes congested.

Table 5.3 shows packet loss rate for both TCP and UDP packets, and Table 5.4 shows packet duplication rate for TCP packets. Packet loss rate is calculated using the total number of lost packets divided by the total number of transmitted packets, and packet duplication rate is calculated using the total number of retransmitted packets divided by the total number of transmitted packets. For UDP, with network coding there is no packet loss for all the four cases, which validates network coding is extremely reliable for best effort delivery. However, for TCP, packet loss rate with network coding is always higher than that with routing, which is still caused by the wait-encode-send behavior of using network coding. This means in the presence of a fast-changing vehicular network topology, reliable transmission like TCP suffers from such kind of coding method. As shown in Table 5.3 and 5.4, packet loss rate and packet duplication rate when there are 10 vehicles is higher than when 20 vehicles because the waiting time for encoding process is longer and thus packets are more likely to be lost or get retransmitted since a vehicle has high probability of moving to a new location or associate with another AP/BS.
5.5.3 Effect of Speed

Fig. 5.8 shows average end-to-end delay per packet vs. average speed. The curve shown in Fig. 5.8 indicates the average delay decreases first and then increases as average speed goes up. Under very low speed, e.g. 10 mph, every vehicle spends a long period in one AP/BS’s coverage, and when the speed increases, vehicles have more chances to meet better signal coverage, yielding less delay for packet delivery. As the speed continues increasing, even if more networks are experienced, data rate will fluctuate more frequently. Thus the delay increases after the speed reaches 30 mph.

In Fig. 5.8, the proposed network coding is able to achieve less delay than routing and pairwise network coding by a range of $215.98\% \sim 640.06\%$ and $57.01\% \sim 202.42\%$, respectively.

Fig. 5.9 shows packet loss rate vs. average speed for the three cases. As speed increasing, the packet loss rate climbs up overall. Packet losses come from 1) buffer overflow at relay nodes in the bottleneck link, and 2) vehicles moving out of current RSU’s coverage and associating with
other RSUs. As expected, there is a trough at 30 mph since vehicles have more chances to see better signal coverage statistically than low speeds. However, if speeds are too high, more packets are lost due to too frequent handover. The promising finding in Fig. 5.9 is no matter what the speed is, with network coding there is no packet loss (zeros in Fig. 5.9), meaning packet loss rate performance with network coding is robust in terms of speed.

### 5.5.4 Effect of Forwarding Links

Fig. 5.10 shows average delay vs. ratio of forwarding links. For all the three cases, the average delay decreases with increasing ratio of forwarding links. From 0.1 to 0.5 for the ratio of forwarding links, the proposed network coding obtain less delay by from 8.53\% to 640.06\%, compared to routing. After that the delay are close for all the three cases. As the ratio being small (< 0.3), there are not so many vehicles that are involved in network coding because a lack of forwarding links
Figure 5.10: Average delay vs. ratio of forwarding links (the number of vehicles is 30).

For decoding. While the ratio is too large (> 0.5), most of vehicles are able to receive routed packets directly from forwarding links instead of going through the bottleneck link. In these two ranges, the gain of network coding becomes small. The maximum delay difference happens when the ratio is equal to 0.5, where the proposed network coding achieves less delay by 640.06% and 202.42% over the routing and the pairwise network coding respectively.

Fig. 5.11 shows packet loss rate vs. ratio of forwarding links when the number of vehicles is 30. For routing, with increasing the ratio, traffic loads are switching from the bottleneck link to forwarding links, thus for some RSUs handling a large amount of packets in a short period yields packet loss. Again, packet loss rate for the two network coding schemes still remains at zero.

Fig. 5.10 and Fig. 5.11 validates the occurrence of bottlenecks resides in the link $r_1 - r_2$ in Fig. 5.4. As explained in the simple example given in Fig. 5.3, if a source node has a forwarding link to the BS connected to the corresponding vehicle, packets are directly forwarded to such forwarding
link. If such forwarding link does not exist, with routing native packets from different sessions are directly forwarded to the link $r_1 - r_2$, but with network coding packets are encoded first on $r_1$ according to the grouping strategy and then forwarded to this link. In the simulation the only difference between the case with routing and the cases with network coding is the number of packets carried on the link $r_1 - r_2$. Given a ratio of forwarding links in Fig. 5.10, the difference of average delay among the three cases results from the number of carried packets on this link. Given a ratio of forwarding links in Fig. 5.11, all the packet losses with routing come from the link $r_1 - r_2$ because there is no packet loss with the two cases of network coding.

### 5.5.5 Effects of Network Selection Algorithms

Fig. 5.12, 5.13, 5.14 show average throughput, delay and packet loss rate with different network selection algorithms, respectively. Util-1 is distributed network selection algorithms. From
Figure 5.12: Average throughput with different network selection algorithms (the number of vehicles is 30). Note: util-1: users' bandwidth preference (Wi-Fi > LTE > WiMAX > HSPA > EVDO); util-2: available bandwidth; util-3: $\sqrt{\text{throughput/delay}}$; util-4: $0.5 \times \text{throughput} + 0.5 \times \text{delay}$.

Util-2 to Util-4 game-based network selection algorithms are used, and more details could be referred to [92]. In Fig. 5.12, there is no significant difference for throughput among these three cases. For delay performance in Fig. 5.13, network coding schemes perform best for all the four network selection algorithms. The maximum delay difference occurs when using Util-3, where the proposed network coding achieves less delay by 831.98% and 234.58% over the routing and the pairwise network coding respectively. In Fig. 5.14, not surprisingly, with network coding, the packet loss rate keeps at zero. For purely distributed network selection algorithm without any aid from cloud, the packet loss rate is the highest with routing. The design of an optimal network selection algorithm is out of the scope of this dissertation and more related information could be found in [92].
5.5.6 Processing Time for Grouping

Fig. 5.15 compares the average processing time for running grouping algorithm in one grouping cycle (1 second) between the proposed network coding and the pairwise network coding. The simulation ran on the same virtual machine with 3.4 GHz CPU and 10.5 GB memory. In Fig. 5.15, both curves increase linearly with increasing the number of vehicles, where the complexity of the proposed grouping algorithm is $O\left(\left(|A|^2 \cdot 2^{|A|} + |A| \right) \cdot |V|\right)$, and that of the proposed pairwise grouping algorithm is $O\left(\left(|A|^4 + |A| \right) \cdot |V|\right)$ (in line10 of Algorithm 5.2 the complexity reduces to $|A|^2$ instead of $2^{|A|}$). However, since the absolute value is very small for pairwise network coding, even if the number of vehicles is 40, the processing time still keeps at a low level, which is 0.0174 s.

The processing time of grouping is introduced by network coding as computing overhead on relay nodes. The processing time shown in Fig. 5.15 indicates that the proposed grouping algorithm has potential to be implemented in practice, especially using pairwise network coding. Given the
complexity of the proposed algorithm, if more computing resources are available for computation, the network is able to scale with increase of the bottleneck capacity or traffic demands.

5.5.7 Discussions

Throughput is not improved in the simulation shown in Fig. 5.6 and Fig. 5.7, which is sacrificed by wait-encode-forward behavior on relay nodes. The latency caused by such behavior consists of two parts: 1) latency of waiting for all packets of eligible sessions to group; 2) latency of encoding. One baseline chosen in the simulation is pairwise network coding, which has the smallest overhead for the grouping and waiting. The results in Fig. 5.6 and Fig. 5.7 show throughput does not have significant difference from the optimal grouping. Hence, the number of sessions for grouping has minor effects on throughput performance, resulting in the latency of encoding is a dominant factor. On the relay node, encoding packets is operated in user space under Linux in
the simulation, which is much slower than direct forwarding of routing. To solve this problem, low-layer implementation of network coding is needed, which can leverage data-plane programmable router/switch, e.g. Dell split data-plane switch [52] or FLARE [23]. If encoding rate is able to catch up or approach forwarding rate, enhanced throughput is expected and delay will be reduced further. Performance of network coding with TCP is also expected to be improved since many packets will arrive before timeout. This dissertation considers inter-session network coding only, however, intra-session usually has significant throughput improvement [91]. Another potential way to improve is to conduct intra-session network coding for each session.
Chapter 6

A Content-based Data Retrieval Framework for Vehicular Networks

In the presence of dynamic channel conditions, opportunistic scheduling under TCP/IP architecture has been considered as an effective way to improve data retrieval on the move. In the past two decades, a huge number of studies have explored opportunistic scheduling schemes for data delivery in mobile wireless networks [5, 88, 73, 29, 99]. However, all of these opportunistic scheduling schemes working with TCP/IP protocols do not rid the cause of the vulnerability of the client-server model in vehicular scenarios. Furthermore, opportunistic scheduling does not inspire great efforts on implementation because of difficulties and limitations encountered in practice.

Instead of patching over TCP/IP, this chapter proposes content-centric vehicular networking (CCVN) as a clean-slate framework for efficient and reliable content retrieval on the move. The rest of the chapter is organized as follows. Section 6.1 presents the basic components of the proposed CCVN. Section 6.2 describes the enhancement for data retrieval using CCVN. Section 6.3 compares and analyzes the performance of the proposed framework.

6.1 Content-Centric Vehicular Networking

This section introduces the proposed framework of CCVN. First, as the foundation of CCN data naming and node structure with key components are introduced. Second, three approaches of
layer design for CCVN are illustrated. Third, differential features of CCVN in vehicular networks are explained. Last, an essential requirement of content caching is discussed.

6.1.1 Naming and Packet Types

CCVN is built based on secure name-based content forwarding. Content has a persistent, unique, and hierarchical name. Content naming is still an active research area, while CCN provides a basic convention for the hierarchical name structure. For example, a typical content name can be like:

/clemson.edu/ccvn/introduction/v1

The above name is composed of multiple segments separate by slash symbols. The first segment can be a globally routable name managed by an organization and can be derived from a DNS name assigned to that organization. The remaining segments depend on specific applications and/or services. Since an entire content is divided into multiple chunks for forwarding, the name can also include chunk number. Cached content also can be expired or updated, so version number can be included in the name as well. Hashing is used to matching names efficiently.

There are two types of packets in CCVN communications- Interest and Data. As shown in Fig. 6.1 vehicle asks for content by broadcasting its Interest specifying a particular content name over all available connectivity. Any node having data matching the same content name can respond
with a Data packet after hearing the Interest.

6.1.2 Node Architecture

Any machine or device running CCVN is considered as a CCVN node, shown as Fig. 6.2. Nodes send and receive packets over faces. A face is a connection point to an application, or to a type of service, or to another node over a specific channel.

Each CCVN node maintains three data structures: (i) a Content Store (CS); (ii) Forwarding Information Base (FIB); and (iii) Pending Interest Table (PIT).

The CS maintains a table of previously incoming Data packets indexed by the Name field of the packet. CS serves as both communication buffering and a content cache. Data packets are self-identifying and self-authenticating and can potentially be useful to many users. Thus, in order to optimize the use of network bandwidth, and reduce user-perceived latency, nodes store the Data packets in their CSs.

The FIB is used to forward Interest packets towards potential data sources. The FIB maintains a set of entries, and each entry containing a name prefix and a list of the faces that might
The Pending Interest Table (PIT) is used to keep track of Interests forwarded upstream towards content source so that Data packets later received can be sent back to their requestor(s). In the PIT each entry contains a previously seen Interest packet and a list of the faces that received the interests. There may be additional information, such as timeout values, that affect the entries. In CCVN, only Interest packets are routed, thus matching Data packets follow the reverse-path of the corresponding Interest packet according to the PITs on nodes along the path.

6.1.3 Layer Design

CCVN provides flexibility of implementation in current TCP/IP layer model. Fig. 6.3 shows three approaches, which are explained in the following.

- Overlay approach:
  CCVN performs as an overlay on top of network layer. Each CCVN node has a bunch of interfaces assigned with IP addresses. Transport and application layers call CCVN’s APIs and the underlying communication for CCVN uses regular routing protocols on network layer, such as BGP, OSPF, RIP, etc. This approach is the easiest way to be compatible with current Internet architecture and needs minimal software requirement. The downside is the overhead of packet forwarding is the maximum among the approaches.

- Independent approach:
CCVN replaces the network layer directly sitting on top of link layer. CCVN has a module of interacting with link layer information, e.g. MAC address learned by ARP protocol, for packet forwarding. Any MAC address for a specific interface, e.g. Ethernet or Wi-Fi interface, should be bundled with a face in CCVN, instead of an IP address in TCP/IP. This approach is a clean-slate approach to use content-centric networking but requires a large amount of software development, especially in the operating system architecture and kernel implementation.

- Hybrid approach:

  This hybrid approach is a combination of the first two. Each CCVN node can operate in two modes: CCVN mode and TCP/IP mode, depending on the requirements and demands of flows. This approach provides the most flexibility in practice.

  CCVN has the ability of co-existing with current TCP/IP architecture. Recently, the overlay approach is already implemented and validated in industry [18], and researchers are implementing the hybrid approach on research router platforms, e.g. Penn and Teller [58].

6.1.4 Differential Features

Three differential features enable CCVN to suit data retrieval in vehicular networks by fulfilling the requirements easily, compared to TCP/IP based opportunistic scheduling.

6.1.4.1 Connection-less Manner

In legacy TCP/IP, if a vehicle requests some content, between the content provider and the vehicle a socket is established, e.g. a TCP or UDP session. Multiple sessions go through the same BS to different vehicles associating with the BS. An example of two sessions is shown in Fig. 6.4. These sessions could be static, long-distance, and long-lived if the requested content is large in size. Intermittent connectivity and packet losses could make data delivery via some sessions inefficient, for example, Vehicle 2 is driving through an area with bad signal coverage. But these sessions are still needed to be maintained. Opportunistic scheduling could let vehicles in good coverage or with high transmission rate transmit as much as possible, temporarily preventing from sending data to those vehicles in bad coverage. Hence a better system throughput could be achieved, and at the same time traffic load could be reduced since lots of retransmissions could be avoided.
In CCVN, such session-based delivery is not required. No CCVN nodes need to maintain and manage sessions. The content providers and any intermediate node only need to simply forward to neighboring nodes if it has the matching content. There is no need to know the state of the remote nodes like TCP. If any vehicle experiences signal drop or packet loss, the Interest will not be successfully arrive at the BS, so that the BS will not take out any bandwidth for sending packets to this vehicle. Naturally, the bandwidth could be saved for sending packets to those vehicles in good coverage. Therefore, CCVN is able to adapt to varying channel condition automatically.

6.1.4.2 In-network Content Caching

In legacy TCP/IP, pre-caching is on demand and requires computation and prediction from opportunistic scheduling scheme. For example, initially a vehicle is associating with BS 1 in Fig. 6.5. BS 1 transmits data from the content provider to the vehicle. Packets with solid color mean they are successfully received by the vehicle. After computation and prediction based on vehicle’s
Figure 6.5: From on-demand caching to in-network caching.

location and movement information, the scheduling server knows the vehicle is driving away from BS 1 and about to enter into BS 2’s coverage. Then the server decides to let BS 2 pre-cache packets (packet in white in Fig. 6.5) for transmissions right after the vehicle executes a handoff from BS 1 to BS 2. These pre-cached packets could be sent from BS 1 according to the queuing buffer or directly from the content provider.

In CCVN, any node receiving an Interest forwards to next hop(s) towards possible source, if it does not have a matching content. In Fig. 6.5, BS 1 could also forwards the vehicle’s Interest to all nearby BSs including BS 2. BS 2 will forwards the Interest and cache any incoming content from the content provider. In this case, once the vehicle handoffs from BS 1 to BS 2, BS 2 possibly already cached multiple chunks of the content (including already transmitted and/or buffered packets at BS 1). After the handoff, if BS 2 does not cache the chunk the vehicle requests in an Interest, it forwards the Interest over possible faces including the one pointing to BS 1. Assuming BS 1 buffered such chunk but does not transmit before the handoff, it will reply BS 2, so that every Data packet coming from the nearest source when the vehicle is moving. The complexity for pre-caching is relaxed and
data retrieval delay after handoffs could be greatly reduced natively.

6.1.4.3 Distributed Control

In legacy TCP/IP, most opportunistic scheduling schemes require centralized control of transmissions at a BS. An additional scheduling server could be assigned or scheduling function could be integrated on the BS. To satisfy accuracy of scheduling and QoS for vehicles, scheduling schemes have to access a large database which could provide sufficient information, e.g. location information and network topology. Powerful computing resources are also required, e.g. for predicting moving path of vehicles in a large network.

In CCVN, every node only needs to keep the required CS, PIT, and FIB on its system, including the BSs and routers, shown in Fig. 6.6. The additional requirement of a CCVN node might be more storage space for caching compared to a node in TCP/IP networks. The distributed control of CCVN relaxes the rigid requirements of computation load and large database on the BS side.

6.1.4.4 Discussions on Content Caching

An essential requirement for CCVN is content object caching. Caching happens in the network via CSs. Layered caching architecture is envisioned in content-centric networking design, e.g. in the CCNx protocol. There are three typical layers- 1) L0- memory cache, 2) L1- physical attached
storage, and 3) L2- network attached storage. The difference between these layers is the tradeoff of storage and latency. Memory cache is able to run at close to line rate but is limited by memory size. L1 cache provides more local storage for using SSD, flash card, etc., sacrificing read/write latency. L3 cache has the most storage resources from network, such as the cache cluster [58], but has the largest latency.

The caching mechanism of CCN itself is an under-explored area in academia. Since caching is optional policy-driven mechanisms for caching is suggested usually. CCVN leverages in-network caching to provided fast data retrieval from neighboring nodes, so this chapter assumes that every node enables at least L0 and L1 cache.

6.2 CCVN Enhancement for Data Retrieval

For improving data retrieval performance in vehicular networks, this section proposes a cloud-based face management strategy to enhance CCVN, and the basic operations are introduced.

6.2.1 Cloud-based Face Management

Faces provide connectivity to applications, services, or other nodes. Conventional CCN does not provide dynamic face management, which cannot suit vehicular networks. A cloud-based face management mechanism is proposed in CCVN. This mechanism has functionality of adding and deleting faces dynamically, by leveraging GPS, information of distributed database, and computing resources in the cloud.

Face management is still an in-band mechanism under CCVN architecture. Vehicle imports data from GPS device and send out interests containing GPS location and speed over current existing faces. Since databases are geographically distributed BSs are configured statically to include the faces pointing to the associated database. Once BSs get these interests from vehicles they will be forwarded to the database over corresponding faces by checking FIB. The database server has full knowledge of network topology of its managing area, e.g. the number of vehicles, vehicles' locations and speed, BSs locations and channels, etc. After the database server receives the interests, it computes and predicts the possible traversing path of this vehicle. The server sends back the obtained optimal face management solution.

The frequency of sending interests of face management on vehicles can be based on the up-
dating rate of GPS, or based on grid size of the map maintained in the database. The requirement for the cloud is powerful computing ability to analyze vehicles’ movements and predict future locations. As a result, the cloud has the ability to sense potential high packet loss rates or connectivity breaks beforehand, and then decides if send necessary information before breaks.

Figure 6.7: Basic Operations of CCVN.
6.2.2 Basic Operations

To explain basic operations in CCVN, Fig. 6.7 is used as an example to illustrate the entire process. There are two vehicles, two BSs in this scenario. The operations shown in Fig. 6.7 are sequential which are marked by numbers. Vehicle 1 is the one that needs to retrieve data.

1. Face management:

Both vehicles periodically send out interests of face management and get data from the distributed database (sequence (1) in Fig. 6.7). The distributed database has full information (historical and recent) of both vehicles locations and speeds. The database server computes the traversing path of Vehicle 1 in the cloud and finds out Vehicle 2 is in Vehicle 1’s communication range, and Vehicle 1 will enter into the LTE BS's coverage shortly. At a point of time, Vehicle 1 receives a Data packet from the database, which includes the face information of the LTE BS and Vehicle 2 (sequence (2) in Fig. 6.7). So Vehicle 1 knows adding faces pointing to Vehicle 2 and the LTE BS can achieve better performance for retrieving data. Thus Vehicle 1 adds both faces according to the received Data packet (sequence (3)). Similarly, Vehicle 2 adds face pointing to Vehicle 1 as well. Note that both vehicles may update their FIBs accordingly after updating their faces.

2. Content retrieval:

Vehicle 1 requests content by sending an Interest packet (sequence (4) in Fig. 6.7), which specifies the name of the requested content, e.g. /clemson.edu/ccvn/intro. The Interest packet will be sent over faces by looking up FIB, so currently the WiMAX BS and Vehicle 2 is able to receive this Interest (Vehicle 1 has not entered into the LTE BS's coverage). Any CCVN node that receives an Interest has the named data in its CS that satisfies the Interest responds with a Data packet. Content satisfies an Interest if the name in the Interest packet matches the name of the content object. If the node does not already have a copy of the requested content, it will check FIB and forward the Interest towards a possible source node. If a node receives a second Interest from a different face but with the same name already existing in the PIT, it updates the face information of that entry but will not forward it again, thus reducing unnecessary traffic load in the network. In Fig. 6.7, the WiMAX BS and Vehicle 2 forward Vehicle 1’s Interest to their next hop(s) and update their PITs because they do not have any cached copy of the request content. Finally, the Interest packet arrives at the content provider.
which has the exact matching content. The provider then replies with a Data packet (sequence (5)), containing the request data, the full name, a signature for the packet, signer identification information, and supporting details. The Data packet could be just one chunk if the request data is separated into multiple chunks. After Vehicle 1 receives the first chunk, it will request the following chunks correspondingly. Any intermediate node received the Data packet can optionally cache this content in its CS for future reply with an Interest with the same name. In Fig. 6.7, the LTE BS and Vehicle forwards the Data packet can choose to cache the content.

6.3 Results and Analysis

Experiments were conducted to evaluate the performance of the proposed CCVN compared to different types of opportunistic scheduling under TCP/IP architecture.

6.3.1 Experiments Overview

All vehicles and RSUs are emulated by virtual machines (VMs) hosted on a Dell R620 server (one VM represents one entity). The proposed CCVN is implemented using CCNx protocol (release 0.8.2) [18]. Transmission rates and packet losses of wireless links between RSUs and vehicles are emulated using tc tools under Linux. To capture packets through interfaces and filter/analyze packets, tshark is used in VMs.
MV-MAX, load-reducing (LR) scheduling, and pre-caching scheduling- are chosen as comparisons to the proposed CCVN scheme in the following three subsections, respectively.

Fig. 6.8 shows the scenarios used for emulation in this section. Scenario 1 is for the comparison to MV-MAX and LR-scheduling, in which there are one BS and ten vehicles (maximum number). The length of considered road is 3200 m. The BS is located in the middle of this highway. The vehicles form a traffic flow with a constant speed at 17.78 m/s. The distance between two vehicles is 200 m. Assume the BS uses LTE technology, data rates refer to the parameters of Table A-3 in the Appendix in [45]. Scenario 2 is for the comparison to pre-caching scheduling, in which there is one vehicle and three BSs. Each BS’s signal covers 200 m on the road. The vehicle drives through at a speed of 17.78 m/s and execute handoff from one to another one it reaches the edge of one BS’s coverage. The width of lanes and size of vehicles are neglected in the emulation. The emulation time is 200 seconds. For simplicity, a vehicle is assumed to re-enter into the road from the other end once it moves out of the road.

6.3.2 Comparison to MV-MAX

Fig. 6.9 shows average throughput per vehicle versus the number of vehicles. For MV-MAX, iperf is used for transmitting data from the BS to vehicles. Both TCP and UDP are evaluated using MV-MAX. In MV-MAX, only when data rate is the largest (29.34 Mbps) the data will be transmitted. In Fig. 6.9, the throughput of CCVN is obviously higher than both cases of MV-MAX. The reason is that all vehicles receive data from the BS at the same time in CCVN, though they have to share bandwidth with more vehicles. Due to the chunk-based mechanism using Interest-Data mode, CCVN will not experience inefficient transmissions with impacts between sessions. Both TCP and UDP throughput of MV-MAX are close to each other and keep flat with the increase in the number of vehicles. The stable throughput is an important characteristic of MV-MAX due to that at a time point there are only few vehicles are allow to transmit/receive in order to avoid collisions and time occupancy by low-rate transmissions. However, MV-MAX’s throughput is still much lower than CCVN, for example, 24.21% and 38.78% less for 10 vehicles with TCP and UDP respectively, which is the minimum difference in the emulation.

Though there is some concern of fairness on short-term throughput or with more random movement in complex road topologies, MV-MAX still can achieve excellent long-term fairness for a drive-thru like road topology. In this section, Jain’s Fairness [31] is defined for evaluating fairness
The number of vehicles
Average throughput per vehicle (Mbps)

CCVN
MVMAX with TCP
MVMAX with UDP

Figure 6.9: Average throughput per vehicle compared to MV-MAX.

The performance, given as:

\[ \gamma = \frac{(\sum_{v \in V} \theta_v)^2}{|V| \cdot \sum_{v \in V} (\theta_v)^2} \]  \hspace{1cm} (6.1)

where \( V \) is the set of all vehicles, and \( \theta_v \) on the right-hand side of (6.1) is vehicle \( v \)'s throughput, indicating the fairness is throughput based.

Fig. 6.10 shows the fairness of CCVN, MV-MAX with TCP and UDP respectively. MV-MAX is able to constantly yield fairness above 0.95 with the increase in the number of vehicles. However, CCVN almost achieves perfect fairness. The reason is still the Interest-Data mode can guarantee all vehicles have the same chance to transmit and receive, even if they use different data rates.

Next, performance of transferring an actual file is compared in Fig. 6.11. In the experiments, download time of a 32 MB file is measured. For MV-MAX with TCP, \( nc \) tool was selected for transferring the file since it has better performance for copying a large file with long-live TCP
connections than other tools like *iperf*, *wget*, and *scp*. Since the diverseness of download time with MV-MAX, all values are shown in Fig. 6.11. For total ten vehicles, three vehicles are able to download the file with less time using MV-MAX, and the remaining seven has larger download time. These three vehicles are the first three in the coverage meeting the highest data rate so that they are allowed to receive data first. For other vehicles, TCP client side at the BS keeps opening but are only allowed to transmit until they enter into the range with the highest data rate (*tc* is used to change/limit the bandwidth). From Fig. 6.11, the time before transmitting actually is the major factor impacts the download time for MV-MAX. Even if a vehicle waits for several seconds and then starts receiving data, the long-waiting TCP socket actually starts transmitting at a very low sending rate, and slowly increasing the window size. That is the main reason that Vehicle 1-4 and Vehicle 10 has much more download time. For CCVN, all the download times are close, which indirectly prove the fairness is almost perfect shown in Fig. 6.11. Again, the Interest-Data mode of CCVN
expresses its advantage by achieving efficient transmission.

6.3.3 Comparison to Load Reducing Scheduler

Link reliability is needed to be defined first. In this section, link reliability is considered as packet successful transmission probability. Similar as [90], four threshold values of packet loss probability is set as \(\{0.03, 0.08, 0.15, 0.5\}\), which represents for links states of excellent, good, moderate, very low. The threshold is set to block transmission to/from vehicles whose link state is worse than the state with the threshold. The Markov chain of link transition is followed [90] and the
corresponding stationary distribution of the above link states are \{0.1127, 0.3803, 0.2535, 0.2535\}. Eight vehicles are considered in this emulation, so at any time point the corresponding distribution of vehicles with above link states is \{1, 3, 2, 2\}. The bandwidth is set as 11.74 Mbps. `iperf` is used for transferring data from the BS to vehicles.

Fig. 6.12 shows load of CCVN compared to LR scheduler with different thresholds. With relaxing the threshold from 0.03 to 0.15, the load of LR scheduler increases obviously. Not only successful transmissions from the BS to vehicles, but also retransmissions due to packet loss are included in the load. With threshold of 0.15, LR scheduler nearly use up the entire bandwidth, however, CCVN only uses partial of the bandwidth. Note that CCVN allows all vehicles transmit/receive packets, no matter which link state a vehicle sees.

In order to measure the percentage of data successfully transmitted out of the total load, a metric of ratio of goodput to load is shown in Fig. 6.13. CCVN can get a very stable ratio around 1, which indicates very few bandwidth is wasted. Once a vehicle sees a bad link state, its Interest packets will be lost on the way to the BS so that the BS will not reply anything. Thought after a timeout the vehicle will re-send the Interest, such traffic is relatively light (327 bytes for an Interest packet) compared to traffic of content (1471 bytes for a regular Data packet). From the BS side, each Data packet sent is based on an Interest packet received, which means the corresponding vehicle is experiencing good coverage with high probability. For LR scheduler with threshold of 0.03, the ratio is stable since few packets are lost; however, with threshold of 0.08 and 0.15, the ratio varies dramatically in the time domain. Due to more retransmission, the instantaneous ratio could be higher than 1 because successfully received packets could be those transmitted but got lost earlier.

Fig. 6.14 shows a quantitative comparison between CCVN and TCP in terms of packet loss
probability. The packet loss probability from 0.01 to 0.15 is considered since there are few packets got successfully received beyond 0.15. The ratio of goodput to load of TCP linearly decreases with respect to increase of the packet loss probability. CCVN has ability of resisting packet loss much better than legacy TCP shown in the figure. Beyond loss probability of 0.1, the ratio of CCVN keeps about 10% larger than that of TCP, up to 12.32% as the loss probability is 0.14. The advantage of CCVN is the BS side will not waste too much bandwidth like retransmissions what TCP does if a vehicle experience high packet losses.

6.3.4 Comparison to Pre-caching Scheduling

Fig. 6.15 shows accumulated download size of CCVN compared to ideal pre-caching using TCP and UDP in scenario 2 in Fig. 6.8. TCP/UDP pre-caching assumes the new BS after handoff already caches the next bunch of packets for continuous transmission for a vehicle. These pre-cached
Figure 6.15: Accumulated download size compared to ideal pre-caching.

packets include the packets in buffer waiting for transmission at the old BS before vehicle’s handoff, and packets whose sequences are following the last received packets at the vehicle. Since it is non-trivial to achieve pre-caching in practice dealing with protocols and Linux kernel, this emulation only considers ideal cases for pre-caching as the baseline (using one continuous TCP/UDP session to mimic the “ideal” case). For CCVN, regular case and pre-caching case are considered. There is one content source behind three BSs and provide a 250 MB file. CCVN with pre-caching means the BSs already caches the content the vehicle requests so that it can directly retrieve the content from its memory and reply. In the CCVN regular case, BSs need to forward vehicle’s Interests and forward any reply from the nearest source to it. The nearest source could be the content source, earlier BS which caches the chunks before handoff. In Fig. 6.15, the download size using CCVN linearly increases through the entire duration, without any interrupt or delay. Compared to the ideal cases, both CCVN downloads less size by around 13% at a time point. The regular
CCVN case achieves closed performance as the CCVN with pre-caching. Though CCVN still has
differences with the ideal case, it could relax the dramatic complexity of pre-caching using legacy
TCP/UDP, yielding consistent downloading performance without any interrupt after handoff. Even
if no advanced mechanism like pre-caching is adopted, regular CCVN is sufficiently robust.
Chapter 7

Conclusions and Future Work

This dissertation considers a cloud-based system supporting robust connectivity and broadband data delivery for vehicles on the move. The common information in the system is discussed for designing cloud-based strategies for vehicular networks. Network selection, data forwarding and data retrieval in the presence of high speed and limited resources, as three core functions in vehicular networks, are investigated in this dissertation.

Chapter 4 presented a cloud-based network selection scheme using a coalition formation game in vehicular networks. The proposed scheme leverages the database maintained in the cloud to assist vehicles on the move to select the best networks. Vehicles are able to make decisions based on the information provided by the cloud in a wider network-awareness scope. Given the considered four throughput-based utility functions, the proposed coalition formation game is able to tradeoff the network and individual vehicles’ performance with the built-in utility division rule. Not limited by throughput-based fairness, the proposed scheme offers a built-in fair framework with the flexibility of defining any type of utility functions. The framework always assures fairness of host utility defined by users. Furthermore, the proposed coalition formation algorithm accelerates the convergence to the final coalition partition, indicating a great potential to support a larger-scale network of up to 8 times in size compared to the one using the conventional algorithm. Through extensive simulation under different conditions in terms of road topology, network availability and mobility pattern, the above two features (a fair framework and fast convergence) are validated as two main contributions in this area, especially for practical implementation.

Chapter 5 presented a cloud-based inter-session network coding scheme for Internet unicasts
in vehicular networks. The proposed scheme leverages the database maintained in the cloud to collect information from networks and provide optimal grouping strategies for inter-session network coding at relay nodes in core networks. A generic model considering an extended butterfly network is proposed and an optimization problem is formulated. A conditionally optimal algorithm for network coding is proposed. Through extensive simulation, the proposed network coding is proven to achieve better delay and packet loss rate than routing for UDP traffic, while keeping comparable throughput. Compared to routing and pairwise network coding, the proposed network coding can achieve less delay by up to 6+ times and 2+ times, respectively. The packet loss rate with network coding remains zero for UDP in the simulation, which is a promising observation. Furthermore, deploying an appropriate network selection algorithm enhances the delay performance of network coding. The processing time of grouping is proven to be possible for implementation in practice. To enhance throughput and performance of network coding with TCP, low-layer implementation of encoding is needed to accelerate encoding rate.

Chapter 6 proposed CCVN as a potential architecture of vehicular networking. By its properties of connection-less, in-network caching, and distributed control, CCVN has the ability to achieve efficient data retrieval in vehicular networks by nature. Compared to MV-MAX, CCVN achieves better throughput by at least 24.21% and almost perfect fairness at the same time. For the comparisons to LR scheduling with packet losses, CCVN shows great robustness resisting lossy links, but still generates much less load than legacy TCP. By leveraging in-network caching, CCVN yields consistent and robust content delivery with frequent handoffs. All these benefits are from the Interest-Data exchanging mechanism in CCVN, which easily adapts to dynamic, intermittent, and short-live connectivity. CCVN provides a promising architecture for vehicular networks with attractive performance, compared to opportunistic scheduling with TCP/IP. The main body of the proposed CCVN is implemented using CCNx protocol under Ubuntu 14.04, providing preliminary validation for future deployment in practice. More interestingly, CCVN can be implemented as an overlay on top of TCP/IP, providing more flexibility of deployment in the future.

Future work of the investigated functions is envisioned in the following.

Network selection: 1) Implement the proposed network selection scheme combining the CyberTiger database and mobile devices (ORBIT based nodes) [30]; 2) Investigate more utility functions based on specific users’ or network operators’ expectations and demands.

Data forwarding: 1) Design a globally optimal algorithm so that the throughput could
obtain more benefits from network coding; 2) Design schemes combining intra-session [91] and the proposed inter-session network coding, enabling throughput robustness in the presence of lossy links.

CCVN: Optimize face management and strategy layer, allowing more efficient and dynamic control on face and FIB.
Appendices
Appendix A  Proof of Theorem 4.1

Proof of Theorem 4.1. Necessary condition: suppose that at time $t$ the decision profile of all players is $x^t = \{x^t_{v_1}, x^t_{v_2}, \ldots, x^t_{v_N}\}$, resulting in a coalition partition $\Pi^t = \{S^t_{a_1}, \ldots, S^t_{a_K}\}$, which satisfies (4.14). Assume a vehicle $v \in V$ is a member of coalition $S^t_{a_i}$ at time $t$, i.e., $x^t_v = a_i$, then $v$'s value of the preference function is $f_v(S^t_{a_i})$. If this vehicle joins any other coalition $S^t_{a_j}$ ($j \neq i$), then the value of the preference function changes to $f_v(S^t_{a_j} \cup \{v\}), \forall a_j \in \epsilon^t_v, j \neq i$. Since $f_v(S^t_{a_i}) \geq f_v(S^t_{a_j} \cup \{v\}), v \in S^t_{a_i}, v \notin S^t_{a_j}, a_i, a_j \in \epsilon^t_v$ is satisfied by (4.14), $v$’s best response at time $t$ is $BR_v(x^t_{-v}) = x^t_v = a_i, \forall v \in V$, therefore, $x^t = \{x^t_{v_1}, x^t_{v_2}, \ldots, x^t_{v_N}\}$ is a Nash equilibrium.

Sufficient condition: suppose that at time $t$ the Nash equilibrium is $x^t_v = \{x^t_{v_1}, x^t_{v_2}, \ldots, x^t_{v_N}\}$, resulting in a coalition partition $\Pi^t = \{S^t_{a_1}, \ldots, S^t_{a_K}\}$. Assume a vehicle $v \in V$ is a member of coalition $S^t_{a_i}$ at time $t$, i.e., $x^t_v = a_i$, then $v$’s value of the preference function is $f_v(S^t_{a_i})$. If this vehicle joins any other coalition $S^t_{a_j}$ ($j \neq i$), then the value of the preference function changes to $f_v(S^t_{a_j} \cup \{v\}), \forall a_j \in \epsilon^t_v, j \neq i$. Since $x^t_v$ is a Nash equilibrium, i.e. $BR_v(x^t_{-v}) = x^t_v = a_i$, the preference relation between coalitions is $S^t_{a_i} \succeq_v S^t_{a_j} \cup \{v\}, \forall a_j \in \epsilon^t_v$, therefore, $f_v(S^t_{a_i}) \geq f_v(S^t_{a_j} \cup \{v\}), v \in S^t_{a_i}, v \notin S^t_{a_j}, a_i, a_j \in \epsilon^t_v, \forall v \in V$ according to (4.13). So (4.14) holds. \qed
Appendix B  Proof of Theorem 4.2

Proof of Theorem 4.2. According to (4.11) and (4.12), $f_v(S^t_a)$ is non-increasing given that $U(S^t_a)$ is a non-increasing function in terms of $|S^t_a|$, $\forall a \in A$. For any coalition $S^t_{a_i}$, suppose vehicle $v$ is the last member joining $S^t_{a_i}$ (not including $v$ yet), then the preference relation satisfies $f_v(S^t_{a_i} \cup \{v\}) \geq f_v(S^t_{a_i} \cup \{v\}), a_i, a_j \in \epsilon_v$. Since vehicle $v$ is the last one joining $S^t_{a_i}$, the output coalition $S^t_{a_i} = S^t_{a_i} \cup \{v\}$, and then $f_v(S^t_{a_i}) = f_v(S^t_{a_i} \cup \{v\}) \geq f_v(S^t_{a_j} \cup \{v\})$. Since $|S^t_{a_j}| = |S^t_{a_i}|$ and $f_v(S^t_{a_i})$ is a non-increasing function in terms of $|S^t_{a_i}|, \forall a \in A, f_v(S^t_{a_i} \cup \{v\}) \geq f_v(S^t_{a_j} \cup \{v\})$.

Therefore, $f_v(S^t_{a_i}) \geq f_v(S^t_{a_j} \cup \{v\}), \forall a_j \in \epsilon_v$. According to Theorem 4.1, the output decision profile $x^* = \{x^*_{v_1}, x^*_{v_2}, \ldots, x^*_{v_N}\}$ of Algorithm 4.1 is a Nash equilibrium and the corresponding $\Pi^* = \{S^r_{a_1}, \ldots, S^r_{a_K}\}$ is Nash-stable. \hfill $\square$
Appendix C  Proof of Lemma 5.1

Proof of Lemma 5.1. Assume there exists a subset $\Psi_n'$ which has a grouping set $\Omega_n(\Psi_n')$ satisfying $|\Omega_n(\Psi_n')| > |\Omega_n(\Psi_n^*)|$, where $\Omega_n(\Psi_n^*)$ is the found grouping set with subset $\Psi_n^*$ from Algorithm 5.2, $\Psi_n' \neq \Psi_n^*$. According to line 22-24, $|\Omega_n(\Psi_n')| = \min(|\Lambda_a'|), a \in \Psi_n'$; and $|\Omega_n(\Psi_n^*)| = \min(|\Lambda_a|^*), a \in \Psi_n^*$, such that $\min(|\Gamma_a'|) > \min(|\Gamma_a|^*)$. According to line 26-27, $\text{tmp}_{\text{max}} = \min(|\Gamma_a'|)$, resulting in the output of the loop from line 14-33 as $\Omega_n(\Psi_n')$, given the number of grouped vehicles $n$. Such output conflicts with the assumption that $\Omega_n(\Psi_n^*)$ is the found grouping set from Algorithm 5.2 where $\Psi_n' \neq \Psi_n^*$. So Lemma 5.1 holds. \hfill \Box
Appendix D  Proof of Theorem 5.1

Proof of Theorem 5.1. Let $\Omega^* = \{\Omega^*_n | n = 2, \ldots, |\Phi_{RSU}|; \Omega^*_n \neq \emptyset\}$ be the output of Algorithm 5.2, and $\Upsilon^* = V \setminus \{v_0 | v_0 \in \text{certain group in } \Omega^*_n, n = 2, \ldots, |\Phi_{RSU}|\}$ accordingly. Assume there exists exactly one vehicle $v$ such that if leaving the current group in $\Omega^*$ or the forwarding set $\Upsilon^*$ and joining another, the ratio $\frac{\sum_{\Omega^*_n \neq \emptyset} |\Omega^*_n| \cdot n + |\Upsilon^*|}{\sum_{\Omega^*_n \neq \emptyset} |\Omega^*_n| + |\Upsilon^*|}$ is larger than $\frac{\sum_{\Omega^*_n \neq \emptyset} |\Omega^*_n| \cdot n + |\Upsilon^*| + 1}{\sum_{\Omega^*_n \neq \emptyset} |\Omega^*_n| + (|\Upsilon^*| + 1)}$, which conflicts with the assumption.

Case III: Suppose $v' \in \Omega^*_m$, in $\Omega^*$ for Algorithm 5.2 and $v' \in \Omega^*_m'$ in $\Omega'$ for the assumption. In this case $\Omega^*_m' > \Omega^*_m$, so $\Omega^*_m'$ cannot exist because it will be found according to Lemma 5.1. Case II: Suppose $v' \in \Omega^*_m$, in $\Omega^*$ for Algorithm 5.2 and $v' \in \Upsilon^*$ in $\Omega'$ for the assumption. In this case, $\Omega^*_m > \Omega^*_m'$ so $\Omega^*_m'$ cannot exist because it will be found according to Lemma 5.1. For more than one vehicle having attempts to deviate from the output of Algorithm 5.2, it is straightforward to find extended cases based on Case I-III. Case I ext.: $\exists$ at least one $v' \in \Omega^*_m$, in $\Omega^*$ for Algorithm 5.2 and $v' \in \Omega^*_m'$ in $\Omega'$ for the assumption satisfying $\Omega^*_m' > \Omega^*_m$. Case II ext.: $\exists$ at least one $v' \in \Omega^*_m$, in $\Omega^*$ for Algorithm 5.2 and $v' \in \Upsilon^*$ for the assumption. Case III ext.: $\exists$ at least one $v' \in \Upsilon^*$ for Algorithm 5.2 and $v' \in \Omega^*_m'$ in $\Omega'$ for the assumption. Using the similar analysis, the assumption cannot stand. Note there is one another case for more than one deviating vehicle. Case IV: There are even number of vehicles and they pairwise switch so that $|\Omega^*_n| = |\Upsilon^*_n|, n = 2, \ldots, |\Phi_{RSU}|$. In this case, the ratios are equal so the assumption cannot stand. Therefore Theorem 5.1 holds. \boxed
Bibliography


