

SMART CONTENT CACHING FOR DEVICE-TO-DEVICE DATA DISSEMINATION

By

Rui Wang

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Electrical Engineering-Doctor of Philosophy

2020

ABSTRACT

SMART CONTENT CACHING FOR DEVICE-TO-DEVICE DATA DISSEMINATION

By

Rui Wang

Wide popularity of wireless devices and their data-enabled applications have created an evolving marketplace for digital content ecosystems. A common operation in those ecosystems is to disseminate content in a cost-optimal manner. With the conventional download model, a user downloads content directly from a Content Provider's (CP) server via a Communication Service Provider's (CSP) network. Downloading content through CSP's network involves a cost, which must be paid either by End Consumers (EC) or the CP. The main objective of the thesis is to provide caching mechanisms that minimizes the overall provisioning cost in different network topologies. This is implemented by caching right objects in data-enabled mobile devices such as smartphones, smart pads, vehicles and novel edge devices. In this thesis, several number of existing caching strategies are studied. Then, an incentive based cooperative content caching framework is developed for both fully-connected Social Wireless Networks (SWNETs) and mobile wireless networks in which content demands are hierarchically heterogeneous. Furthermore, a D2D cooperative caching framework is proposed for streaming video with heterogeneous quality demands in SWNETs. This caching framework contains two main components: a value-based caching strategy in which the value of caching a streaming video segment is defined for given pricing and video sharing models, and an Adaptive Quality (AQ) provisioning algorithm that minimizes the overall video content provisioning cost within an SWNET. Additionally, a vehicular content caching mechanism is developed for disseminating navigational maps while minimizing cellular network bandwidth usage. The key concept is to collaboratively cache the dynamic

components of navigational maps in roadside units (RSUs) and vehicles such that the majority of dissemination can be accomplished using V2V and V2I communication links. Moreover, a novel caching mechanism is proposed which is based on Connectionless Edge Cache Servers in vehicular networks. The goal is to intelligently cache content within the vehicles and the edge servers so that majority of the vehicle-requested content can be obtained from those caches, thus minimizing the amount of cellular network usage needed for fetching content from a central server. A notable feature of the cache servers in this work is that they do not have backhaul connectivity. This makes the connectionless servers to be relatively less expensive compared to the usual Roadside Service Units (RSUs), and potentially moveable in response to specific events that are expected to generate content in large volumes. Finally, a list of future work on this topic is compiled that includes: 1) developing machine learning models for predicting content demand and spatiotemporal localities of node movements, 2) developing mechanisms for edge cache server placement for performance optimization, and 3) analyzing the impacts of selfishness on the performance of caching.

ACKNOWLEDGEMENTS

My sincere gratitude to my advisor, Dr. Subir Biswas, for his extraordinary support, encouragement and guidance during my work. It was through his mentorship that I learned how to start and own a research problem, think independently, and find hope in failure.

I would also like to thank my committee Dr. Richard Enbody, Dr. Nihar Mahapatra and Dr. Jian Ren for their time and support.

Thanks must be given to my lab mates Faezeh Hajiaghajani, Saptarshi Das, Bo Dong, Brandon Harrington, Henry Griffith, Dezhi Feng and Yan Shi for all of the brainstorming and implementation discussions. I would like to thank my son as his presence in my life has been a motivation for me. Finally, I would like to give thanks to my beloved wife and my parents for their unconditional love and support and encouragement throughout my life. It would not have been possible to stand where I am now without them.

TABLE OF CONTENTS

LIST OF TABLES	viii
LIST OF FIGURES	ix
LIST OF ALGORITHMS	xii
Chapter 1: Introduction	1
1.1 Caching for Cellular Bandwidth Cost Reduction	2
1.1.1 Traditional Infrastructure-based Caching	2
1.1.2 Device-to-Device Cooperative Caching	3
1.1.3 Connectionless Edge Cache Servers	4
1.2 Content Search	5
1.3 Cache Replacement	6
1.4 Incentives, Security Issues and User Selfishness in D2D Caching	6
1.5 Dissertation Objectives	7
1.6 Scope of Thesis	8
Chapter 2: Related Work	11
2.1 Caching in Stationary Networks	11
2.2 Caching in Mobile Networks	12
2.2.1 Traditional Infrastructure-based Caching	12
2.2.2 Device-to-Device Cooperative Caching	14
2.3 Summary	16
Chapter 3: Cooperative Caching in Social Wireless Networks	18
3.1 Introduction	18
3.2 Hierarchically Heterogeneous Requests	18
3.2.1 Global Popularity of Object	19
3.2.2 Local Popularity of Category	19
3.2.3 Local Popularity of Object	20
3.2.4 Request Generation Process	21
3.3 Network Model and Problem Formulation	21
3.3.1 Content Search Model	21
3.3.2 Pricing Model	21
3.3.3 Cost under Heterogeneous Request Model:	22
3.3.4 Problem Definition	23
3.4 Heterogeneous Split Caching Algorithm	23
3.5 Performance Evaluation	26
3.5.1 Impacts of Zipf parameter (α) and Rebate cost ratio (β)	27
3.5.2 Comparison with Traditional Caching Strategies	30
3.5.3 Object Density	31
3.5.4 Convergence of Baseline HSC to Prefilling HSC	33

3.6	Summary	33
Chapter 4: Distributed Caching in Mobile Wireless Networks		35
4.1	Introduction.....	35
4.2	Content Search with Tolerable Access Delay (TAD).....	35
4.3	Heterogeneous Split Caching (HSC) for Mobile Networks	35
4.4	Performance evaluation	36
4.4.1	Monolithic Mobility.....	38
4.4.2	Community-based Mobility	43
4.4.3	Helsinki-map Mobility.....	46
4.5	Summary.....	46
Chapter 5: Caching for Streaming Video in Social Wireless Networks.....		48
5.1	Introduction.....	48
5.2	Content Search and Pricing Model	48
5.3	Streaming Video Play Model.....	48
5.3.1	Play Buffering and Caching.....	48
5.3.2	Video Play Model	49
5.4	Value-based D2D Caching	52
5.4.1	Value of Caching a Video Segment.....	52
5.4.2	Value-based D2D Caching Algorithm at the ECs Devices	54
5.5	Adaptive-Quality Content Provisioning by Content Provider	56
5.6	Performance Evaluation.....	60
5.6.1	Experimental Settings	60
5.6.2	Impacts of Cache Availability and Terrain Size on Cost Saving.....	62
5.6.3	Evolution of Provisioning Cost and Hit Rates over Time	66
5.6.4	Content Quality Density of the Cached Objects.....	67
5.6.5	Content Delivery Latency	68
5.6.6	Impacts of Video Play Sequence	69
5.7	Comparison with Reactive User Preference Profile Algorithm.....	70
5.7.1	Impacts of Cache Size.....	71
5.7.2	Impacts of Rebate-to-Download-Cost Ratio.....	72
5.7.3	Impacts of Different Quality Preference Distributions	73
5.8	Summary.....	74
Chapter 6: Caching for Dynamic Map Dissemination in Vehicular Networks		76
6.1	Introduction.....	76
6.2	System Architecture.....	77
6.2.1	Network Model	77
6.2.2	Dynamic Map Data	78
6.3	Dissemination and Caching Mechanisms	79
6.3.1	Dynamic Map Data Push	79
6.3.2	Dynamic Map Data Pull and Caching	80
6.3.2.1	Demand Model.....	81
6.3.2.2	Data Search Model.....	81
6.3.2.3	Cache Replacement Policy	82

6.4	Performance Evaluation.....	84
6.4.1	Mobility Characteristics.....	85
6.4.2	Impacts of In-vehicle Caching on Bandwidth Usage (BU)	86
6.4.3	Caching Dynamics	87
6.4.4	Impact of in-RSU Caching.....	89
6.4.5	Larger RSU Coverage.....	90
6.4.6	Map Fetch Latency	92
6.5	Summary.....	93
Chapter 7: Content Dissemination Through Mobile Edge Cache Servers in Vehicular Networks.....		94
7.1	Introduction.....	94
7.2	Content Search and CECS Operation	96
7.3	Caching Mechanism.....	97
7.3.1	Content Segment Distribution.....	97
7.3.2	Cache Metadata Tables (CMTs).....	102
7.3.3	Cache Replacement Policy	106
7.4	Performance Evaluation.....	107
7.4.1	Impacts of Cache Space in the CECSs (α)	110
7.4.2	Impacts of Available In-vehicle Cache Storage Space (β).....	113
7.4.3	Impacts of Tolerable Access Delay (TAD)	117
7.4.4	Comparison with Sparse-CECSs Network	118
7.4.5	Caching Performance in a Synthetic Network.....	120
7.5	Summary.....	122
Chapter 8: Future Work		123
8.1	Introduction.....	123
8.2	Machine Learning Models for Content Caching and Dissemination.....	123
8.3	Placement of Edge Cache Servers	124
8.4	Handling Selfishness.....	124
BIBLIOGRAPHY.....		125

LIST OF TABLES

Table 3-1: Simulation's baseline parameter	27
Table 4-1: Simulation's baseline parameter	38
Table 5-1: List of all notations used in the caching algorithm	56
Table 5-2: Baseline parameters used in the simulation experiments	61
Table 5-3: Average no. of neighbor ECs encountered per node for different terrain sizes (number of nodes is set to 40)	63
Table 6-1: Baseline parameters used in the simulations	85
Table 7-1: Baseline parameters used in the experiments	109

LIST OF FIGURES

Figure 1-1: Network architecture of digital content access	1
Figure 1-2: Content access methods for D2D networks	4
Figure 1-3: Example of content dissemination through a CECS	5
Figure 1-4: Summary of investigated topics in the thesis	8
Figure 3-1: Content and cost flow	22
Figure 3-2: Cache Partitioning in HSC	23
Figure 3-3: Impact of α on cost	28
Figure 3-4: Hit rates for baseline HSC	29
Figure 3-5: Impact of β on cost	30
Figure 3-6: Comparison of minimum cost	31
Figure 3-7: Object density of various algorithms	32
Figure 3-8: Convergence of cost for HSC baseline to the pre-filling scenario	33
Figure 4-1: Cost V.S. λ for baseline HSC and SFN	38
Figure 4-2: Hit rates for baseline HSC and SFN	39
Figure 4-3: Minimum cost V.S. terrain size for baseline HSC and SFN	40
Figure 4-4: Cost V.S. λ for baseline HSC with various β and terrain sizes	41
Figure 4-5: Cost V.S. λ for HSC-baseline and prefilling	42
Figure 4-6: Comparison with traditional algorithms in monolithic scenario	43
Figure 4-7: Cost V.S. λ for HSC baseline	45
Figure 4-8: Comparison with traditional algorithms in community-based scenario	45
Figure 4-9: Comparison with traditional algorithms in Helsinki	46
Figure 5-1: Timeline of video play model	49
Figure 5-2: State machine of streaming video play model	51

Figure 5-3: Example of how providing higher-than-requested quality content can serve future high-quality user demands	57
Figure 5-4: Default video quality preference distribution	62
Figure 5-5: Provisioning cost with different terrain sizes and per-node cache availability.....	63
Figure 5-6: Local and remote hit rates for different network terrain sizes	64
Figure 5-7: Total value of cached objects in the entire network.....	65
Figure 5-8: Bandwidth overhead for different network terrain sizes.....	66
Figure 5-9: Provisioning cost, and local and remote hit rates over time	67
Figure 5-10: Cached video quality density for AQ and RQ	68
Figure 5-11: Content Delivery Latency (CDL) for the AQ and RQ policies	69
Figure 5-12: Impacts of different video play sequences on provisioning cost	70
Figure 5-13: Provisioning cost and hit rates of R-UPP, RQ, and AQ	72
Figure 5-14: Provisioning costs for varying rebate-to-download-cost ratio	73
Figure 5-15: Provisioning cost with different distributions of video quality preferences	74
Figure 6-1: Layered abstraction of dynamic maps	76
Figure 6-2: Generalized model with different networking components	78
Figure 6-3: Push mechanism of dynamic data dissemination.....	80
Figure 6-4: Mean vehicle encounter rates for the vehicles and RSUs.....	86
Figure 6-5: Bandwidth usages with different in-vehicle cache sizes.....	86
Figure 6-6: Temporal bandwidth usages of V2V and cellular links.....	88
Figure 6-7: Impacts of allocated cache sizes in the RSUs	90
Figure 6-8: Mean vehicle encounter rate of RSUs.....	90
Figure 6-9: Impacts of allocated cache sizes in the RSUs with long-range V2I	91
Figure 6-10: Bandwidth usage with varying number of G-RSUs with long-range V2I.....	92
Figure 6-11: Map fetch latency with various in-vehicle cache sizes	92

Figure 7-1: Content dissemination and caching using edge infrastructures	96
Figure 7-2: Example of the accessible CECSs to a vehicle during the TAD	99
Figure 7-3: Example updates of the CMTs within CECSs	101
Figure 7-4: Example of CMT exchange between a vehicle and a CECS	103
Figure 7-5: Map of the simulation scenario in the East Lansing area	107
Figure 7-6: Cellular retrieval rate and hit rate at CECSs with various α	111
Figure 7-7: Impacts of CECS cache space α on Completeness Index μr	111
Figure 7-8: Content Delivery Latency for different caching mechanisms	112
Figure 7-9: CMT exchange overhead with varying storage in the CECSs	113
Figure 7-10: Impacts of varying in-vehicle cache storage space β	114
Figure 7-11: Impacts of in-vehicle cache storage space on average CI (μr)	115
Figure 7-12: CDL with different in-vehicle cache storage space β	115
Figure 7-13: CMT exchange overhead with varying vehicle cache space	116
Figure 7-14: Evolution of cellular retrieval rate and the completeness index μr	117
Figure 7-15: Impact of TAD on cellular retrieval rate and hit rate at CECSs	117
Figure 7-16: East Lansing scenario with reduced number of CECSs	118
Figure 7-17: Cellular usage with various α with different CECS-counts	119
Figure 7-18: Cellular usage with β with different CECS-counts	120
Figure 7-19: CECS placement in a synthetic network scenario	121
Figure 7-20: Impacts of CECS cache space in the synthetic scenario	121
Figure 7-21: Impacts of vehicle cache space in the synthetic scenario	122

LIST OF ALGORITHMS

Algorithm 3-1: Caching algorithm and replacement policy in HSC	26
Algorithm 5-1: Value-based caching and replacement policy.....	56
Algorithm 6-1: In-vehicle and in-G-RSU caching/replacement policy	83
Algorithm 7-1: CMT exchange algorithm between a vehicle and a CECS.....	105

Chapter 1: Introduction

Wide popularity of wireless devices such as Android, iPhone, Windows Phone, and their data-enabled applications such as Kindle book reader, Netflix, and various content stores have created many digital content ecosystems. As shown in Figure 1-1, users with mobile devices (e.g. smart phones and smart pads, etc.) can obtain content such as e-books, music and movies through cellular networks (e.g. 5G, etc.) from these ecosystems. In addition to user-carried mobile devices, such ecosystems can also cater to various smart devices installed on machines. A prominent example of the latter is in-vehicle wireless interfaces. With more and more vehicles being wirelessly connected through Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I) [1], and cellular links, many data-driven applications are emerging for in-vehicle usage. Such applications typically also involve digital content including music, books, and streaming audios and videos.

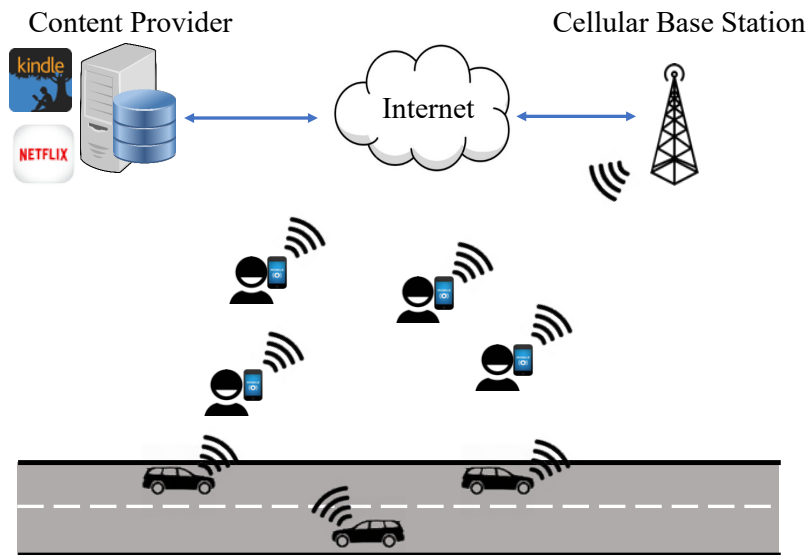


Figure 1-1: Network architecture of digital content access

A common requirement in those ecosystems is to be able to disseminate content (e.g., books, magazines, music, etc.) in a manner that reduces the cost of cellular link usage. Traditionally, an End Consumer (EC) downloads content directly from a Content Provider's (CP) server in the core

network via a Communication Service Provider (CSP) such as AT&T, Verizon and Sprint, etc.. This approach involves a cost which must be paid either by ECs or by the CP. For example, in the business model of Amazon Kindle electronic book delivery [2, 3], the CP (Amazon) pays to the CSP (Sprint) for the cost of network usage due to content downloaded by users.

1.1 Caching for Cellular Bandwidth Cost Reduction

One possible solution for reducing such cost is to cache the most popular content in local infrastructures or mobile devices with storage. In this way, the ECs can often find their requested content within those caches, and avoid the communication cost of always downloading them from the CP's server via the CSP's network.

1.1.1 Traditional Infrastructure-based Caching

Traditional infrastructure-based models of caching have been proposed for communication cost reduction by pushing popular content to local infrastructures near the targeted EC population. One typical implementation of such models is edge servers in Content Delivery Networks (CDN) [4]. These edge servers are usually deployed proximal to the ECs. For example, the Wi-Fi Access Points (APs) [5] installed in a building can be directly accessed by the ECs in the building. The APs also connect to the CP's servers through the Internet without going through the CSP's cellular networks. The popular content can be cached in such local edge servers so that the ECs can fetch the content directly from these infrastructures. Another kind of edge device is roadside units (RSUs) [6] deployed beside the roads. These devices connect to the Internet through fiber links, thus they can cache and relay the popular content to the nearby vehicles via Dedicated Short-Range Communications (DSRC) [7] links.

Such infrastructure-based caching avoids downloading content repeatedly from the CP's (e.g., Amazon's) server via the CSP's network, thus it reduces the cellular network usage. However,

for each single local infrastructure, its coverage (i.e. transmission range) is limited. For example, a typical transmission range of 802.11 (e.g., Wi-Fi) is only between 100 and 300 meters [8]. Thus, the ECs out of the range from a Wi-Fi AP are not able to access any content cached in the infrastructure. Additionally, it may be expensive to deploy too many caching infrastructures (i.e., edge servers) for improving the coverage.

1.1.2 Device-to-Device Cooperative Caching

An alternative model would be to cache content in the ECs' devices following the localities in human interactions and content interests. One typical example is when users physically gather in settings such as university campuses, malls, airports, and other public places, Social Wireless Networks (SWNETs) [9] can be formed over ad hoc wireless connections among their mobile devices [10]. Examples of SWNETs would include students in a university campus, a group of colleagues in a workplace, and people in a shopping district. In this model, each device in a SWNET is able to cache content following some policy that leverages the above localities. For downloading content such as a Netflix movie, a user device can first search within its SWNET for the requested content before downloading it from the CP's server. The expected CSP's communication cost in this approach can be lower since the download cost paid to the CSP would be avoided when the content is found within the local SWNET of the requesting user. This is termed as *cooperative caching*. Figure 1-2 (a) shows an example of an SWNET formed by several users with mobile devices. Under this setting, the ECs, carrying mobile devices, can obtain an object either: 1) directly from the CP's server through the CSP's cellular network, or 2) locally, from other devices they interact with through high-speed wireless links (e.g. Bluetooth Low Energy, Wi-Fi, etc.).

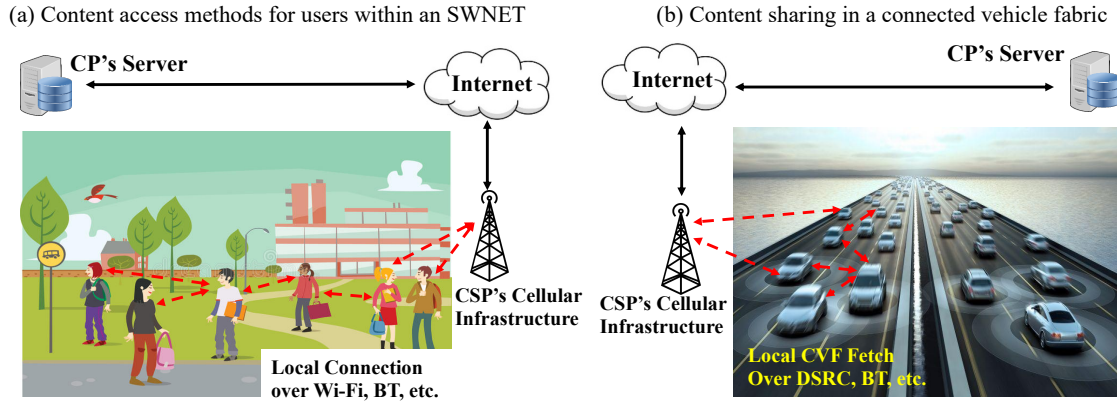


Figure 1-2: Content access methods for D2D networks

Similarly, in the vehicular context shown in Figure 1-2 (b), cellular bandwidth usage cost reduction is accomplished by letting a vehicle first search content in nearby Connected Vehicle Fabric (CVF) via DSRC or Bluetooth, etc.. A download from a CP's server via 5G link is triggered only if the local CVF search fails.

Compared with traditional infrastructure-based caching models (e.g., at the edge servers, etc.), there are at least two notable advantages of the proposed D2D cooperative caching. First, with D2D caching, there is no need to add any new hardware infrastructure. Second, unlike the infrastructure-based models, the coverage and shared storage capacity of a D2D network can be organically extended. This happens as more ECs join the network and contribute to the growth of overall cache storage in the network. Another example is when some ECs fetch content from one D2D network, and are able to disseminate them in another network as they move across networks.

1.1.3 Connectionless Edge Cache Servers

To address the issues of traditional edge cache servers in Section 1.1.1, the concept of Connectionless Edge Cache Servers (CECSs) is introduced in this thesis. The idea is to use such servers without incurring the cost of backhaul connectivity, while gaining the ability to make the cache server mobile. Such mobility can provide a great deal of flexibility in temporarily placing

them in areas with low vehicle and RSU densities and high content demands. Putting the CECS on vehicles and placing them on-demand can cater to events such as games, accidents, weather conditions, etc. As shown in Figure 1-3, Without backhaul connectivity to a CP, the CECSs can cache content collected via DSRC links from the current-passing vehicles and provide them to future-passing vehicles over DSRC, and reduce the cellular bandwidth usage of the vehicles in that process.

Compared with caching at vehicles or traditional infrastructures (e.g., RSUs, etc.), the advantages of caching at a CECS are as follows: 1) a CECS can be used as an intermediate device for data dissemination among vehicles, especially in the scenarios where the V2V connectivity is sparse so that it is difficult for vehicles to disseminate the data directly through V2V links, and 2) the cost for deploying a CECS is less than a traditional cache server because there is no the requirement of backhaul connectivity for a CECS.

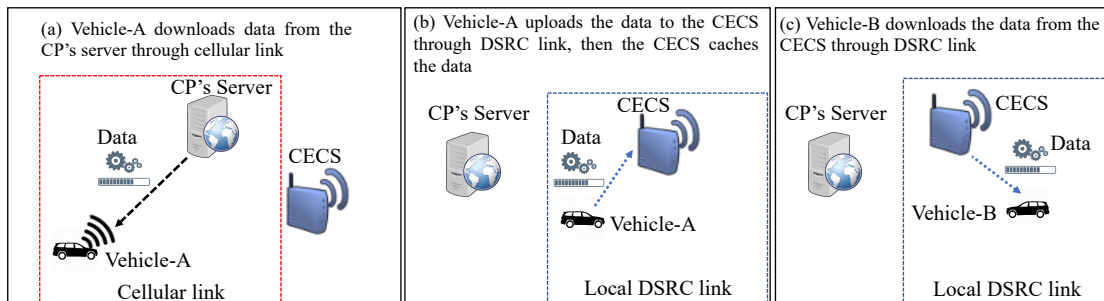


Figure 1-3: Example of content dissemination through a CECS

1.2 Content Search

Upon originating a request, a node (i.e. an EC) first performs a local search for the requested content in its own cache. If that fails, the node performs a remote search in its local network. If the node cannot obtain the content from the network within a pre-defined Tolerable Access Delay (TAD), it sends a request to the CP's server. TAD, which is part of a request, represents the duration that a user application is willing to wait before a successful content request is served. The remote

search can take place using devices' Wi-Fi, Bluetooth, and other short-range RF interfaces (e.g. DSRC, etc.). The content from the CP's server is downloaded using a CSP's cellular network.

The content search and retrieval within the local networks can be performed using traditional IP routing protocols or the emerging Information-Centric Networking (ICN) based protocols such as Named Data Networking (NDN) [11,12,13]. In other words, this content retrieval can work seamlessly with ICN/NDN mechanisms that may be already in place within the local networks.

1.3 Cache Replacement

Due to limited storage, mobile devices are not expected to store all downloaded content for long. This means after downloading and using a purchased electronic content, a device may remove it from the storage following a replacement policy for keeping the most popular content in the storage. Possible cache replacement policies include Least Frequently Used (LFU) [14], Least Recently Used (LRU) [15], Random replacement [16] and popularity-driven policy [17], etc..

1.4 Incentives, Security Issues and User Selfishness in D2D Caching

In order to encourage an EC to cache previously downloaded content and to share it with other ECs, a micro-rebate mechanism such as the ones proposed in [18, 19, 20] can be used. These micro-rebates that an EC can redeem with the content provider at a later time can serve as an incentive and compensation for the resource used during D2D caching. Such resources include both device storage and battery drainage due to caching and D2D content transfer.

A key requirement for implementing D2D cooperative caching would be to implement a digitally signed rebate framework in which the rebate recipients can electronically validate and redeem rebates with the CP. Also, a digital usage right mechanism [21] is needed so that an EC who is caching a content should not necessarily be able to use it unless she had explicitly purchased

the content from the CP. Such digital usage right mechanisms are already available in the literature [22], and are relied on for the proposed caching architecture.

The potential for earning peer-to-peer rebate may promote selfish behavior [23]. A selfish user is one that deviates from the network-wide optimal caching policy in order to earn more rebates. Any deviation from the optimal policy is expected to incur higher network-wide provisioning cost. In this thesis, it is assumed that the users are not selfish in that they fully comply with the network-wide optimal policy.

1.5 Dissertation Objectives

The objective is to design optimal mechanisms for caching at mobile devices such that under different network topologies the network-wide provisioning cost is minimized. A key question for content caching is: how to store content in mobile nodes so that the overall content provisioning cost in the network is minimized. This question needs to be addressed under heterogeneous user demands. All results presented in our previous work [24] were for homogeneous request conditions in which it was assumed that all consumers' requests follow a common global Zipf [25] distribution for content popularity. While providing a reasonable model for a baseline cooperative caching architecture, a common global popularity for content does not represent real scenarios in which content preferences usually vary significantly across different ECs or local areas [26]. For example, a globally popular e-book may not be preferred in some specific areas. Another example is that each user may request different qualities for the same video resulted by the limit of device or network resource, or personal situation (e.g. membership of Amazon Prime, etc.). Thus, the heterogeneity of quality demands for the same video content needs to be involved for designing a cooperative caching mechanism.

In this thesis, we attempt to develop caching structures that cater to such content preference heterogeneity. In [27] a Distributed Benefit based heuristics was used for optimal caching in the presence of request heterogeneity. The primary limitation of this approach is in its assumption that a centralized content server requires to store the request preference of each consumer, thus rendering it non-scalable. Especially so, when there are potentially millions of consumers in the system. The main goal of this thesis is to present scalable yet optimal approaches to implement content caching with a fully heterogeneous request model.

1.6 Scope of Thesis

The main objective of the thesis is to provide content caching mechanisms that minimizes the overall provisioning cost in different network topologies. This is implemented by caching right objects in data-enabled mobile devices such as smartphones, smart pads, vehicles and novel edge devices. When people with smart devices physically gather in a place such as university or downtown area, they may be interested in the same content that is statistically popular in this place. The summary of investigated topics in the thesis is shown in Figure 1-4.

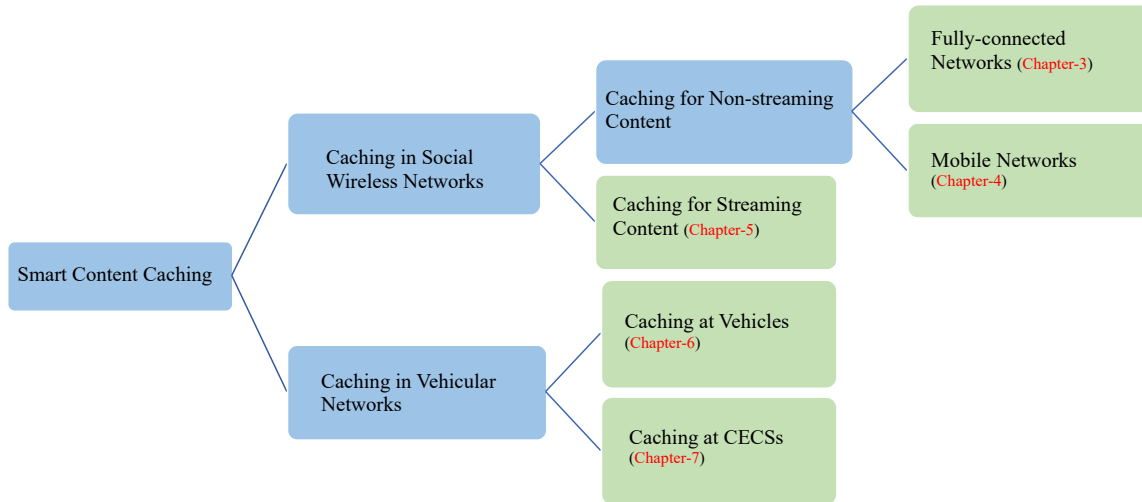


Figure 1-4: Summary of investigated topics in the thesis

Chapter-2 is a survey about existing caching strategies. First, content caching mechanisms in traditional stationary wired networks are reviewed, even though most of the ideas cannot be directly used in wireless networks. Then some most recent caching strategies, including infrastructure-based and D2D cooperative caching policies, are introduced. Furthermore, the differences and advantages of the proposed caching mechanisms in this thesis are highlighted and compared to these existing caching strategies.

In Chapter-3, an incentive based cooperative content caching framework is developed for Social Wireless Networks (SWNETs) in which content demands are hierarchically heterogeneous. The heterogeneous request model incorporates user preference for different categories/genres and contents under each category. The experiment results show that the proposed mechanism is able to reduce content provisioning cost compared to traditional caching mechanisms in fully-connected SWNET.

In Chapter-4, the caching mechanism proposed in Chapter-3 is applied on the scenario of mobile wireless networks. Unlike Chapter-3, in this scenario the connection between each pair of nodes is not stable anymore because a node may dynamically join or leave a network. However, the experiments show that the proposed mechanism is still able to reduce bandwidth usage and the resulting content provisioning cost compared to traditional caching mechanisms in both monolithic and community-based mobility scenarios.

In Chapter-5, a D2D cooperative caching framework is proposed for streaming video with heterogeneous quality demands in SWNETs. This caching framework is formed of two components: a value-based caching strategy in which the value of caching a streaming video segment is defined for given pricing and video sharing models, and an Adaptive Quality (AQ) provisioning algorithm that minimizes the overall video content provisioning cost within an

SWNET. The simulation results indicate that the proposed mechanism is able to appreciably reduce the overall video provisioning cost in the presence of end-user mobility.

In Chapter-6, a vehicular content caching mechanism is presented for disseminating navigational maps while minimizing cellular network bandwidth usage. The key concept is to collaboratively cache the dynamic components of navigational maps in roadside units (RSUs) and vehicles such that the majority of dissemination can be accomplished using V2V and V2I communication links. The simulation results indicate that compared to infrastructure-only caching strategies, the proposed vehicle-involved collaborative caching mechanism is able to reduce the bandwidth usage of cellular networks and the delivery delay for obtaining dynamic map data.

Chapter-7 presents a caching mechanism based on a novel edge infrastructure CECS for Software Update Package (SUP) dissemination in the context of vehicular networks. The research goal is to intelligently cache content at CECSs and vehicles such that the cellular bandwidth usage is reduced. Using the DTN simulator ONE, we run detailed simulations in context of vehicular networks in the East Lansing area and a synthetic scenario. The results indicate that the proposed caching mechanism is able to reduce the cellular bandwidth usage and SUP fetching delay compared to some other caching strategies.

Finally, in Chapter-8 this thesis is summarized, and a list of future work is compiled.

Chapter 2: Related Work

2.1 Caching in Stationary Networks

The nodes and topologies in a stationary wired network may not often change. Moreover, the connection between each pair of different networks is also relatively stable. These characteristics result that each node can always access the other nodes in the networks.

Approaches of web caching have been widely used in stationary wired networks [28, 29]. However, the emerging information centric networking (ICN) [30, 31, 32], which is also known as content centric networking (CCN), involves caching mechanisms to improve the performance of content retrieval including e-books, music and videos, etc.. For example, a Dynamic Adaptive Streaming over HTTP (DASH) strategy is adopted in [33] for Dynamic Adaptive Streaming over Content centric networking (DASC). With DASH, a content centric networking (CCN) node is used instead of HTTP for caching popular video content. Still based on CCN, a new caching strategy, namely, Most Popular Content (MPC) is presented in [34]. By caching only popular content, this method is able to cache less content while still achieving a higher cache hit.

Even without CCN, a two-tier caching strategy is presented in [35] for streaming video over the IPTV. In this thesis, the video server connects with many local area networks, each of which maintains a separate cache server that serves all clients within its network. In [36] a content caching scheme, WAVE, is presented in which the number of chunks to be cached is adjusted based on the popularity of the content. This caching mechanism is for lowering the hop count of content delivery while increasing the cache hit ratio. Finally, in [37] an architectural framework named CachePortal system is proposed for enabling dynamic content caching for database-driven e-commerce sites that most of the traditional caching strategies cannot handle.

However, in a wireless mobile network the nodes may dynamically join or leave the network over time. This means a node is not always reachable, and results that most of the caching mechanisms in stationary wired networks are not available in wireless mobile networks.

2.2 Caching in Mobile Networks

Wireless communication is a fast-growing part in communication area. Compared with stationary wired networks, the topology of a wireless mobile network is more dynamic. Thus, the content caching mechanisms in wireless mobile networks [38, 39] should be different from those in wired networks.

2.2.1 Traditional Infrastructure-based Caching

Traditional infrastructure-based models of caching have been proposed for communication cost reduction by pushing popular content through the core networks to local infrastructures near the targeted user population. A caching approach based on Small Base Stations (SBSs) is presented in [40]. In this solution, the base station serves only few influential nodes, and lets them help in disseminating the content to other nodes. That simplifies the content dissemination process, and increases offloading gains in mobile wireless networks. In [26], a method called Reactive User Preference Profile (R-UPP) is introduced for caching streaming videos in base-stations of Radio Access Networks as a way to reduce the need for downloading requested videos from Content Distribution Networks. R-UPP is based only on the popularity of objects without considering the underlying quality preference distributions. Similarly, the caching strategy proposed in [41] is based on the base-stations. In this case, the cache server is deployed in Mobile Switching Centers (MSC), which can be connected to multiple base stations. Another example of infrastructure-based streaming video caching was proposed in [42, 43]. The authors propose a proactive caching approach named “smart scheduler” for railway systems. Popular content is proactively cached in

each railway station before the next train arrives. The objective is to improve the content distribution throughput in transportation network system. Using a Markov process-based modeling of content dynamics, the authors in [44] propose a proactive caching mechanism for reducing retrieval delay of video content in the presence of disconnections in 5G cellular networks.

In the context of vehicular networks, the authors in [45] present a novel vehicular network architecture in which the RSUs are allocated with large storage capacity and the needed V2I and backhaul links for effective content caching. The main objective here is to minimize the average delay for obtaining content by caching the popular content in the RSUs. An integer linear programming (ILP) based optimal content prefetching algorithm within Access Points (Aps) is presented in [46] for vehicular networks. The proposed method is for maximizing the probability of access of requested content that is cached in the APs. Another caching method in [47] attempts to improve the accessibility of requested content by caching popular contents in cache stores (CS) in information-centric networks (ICNs). While being able to provide effective cache performance, all these mechanisms extensively rely on infrastructure such as RSUs, cache stores, and other dedicated caching objects. This limits the effectiveness of such mechanisms in vehicular networks with insufficient infrastructure availability. The following papers report research on in-vehicle caching.

The main disadvantage of traditional infrastructure-based caching is that the coverage of each single local infrastructure is limited. Thus, the users out of the range from a infrastructure are not able to access any content cached in the infrastructure. Additionally, it may be expensive to deploy too many such caching infrastructures with backhaul connectivity for improving the coverage.

2.2.2 Device-to-Device Cooperative Caching

Unlike infrastructure-based caching, in D2D cooperative caching the content is cached on the mobile devices of each user in the networks. A distributed heuristic solution for effective replica placement in Wireless Mesh Networks (WMNs) is presented in [48]. The local popularity of an object, which is defined as the relative demand for it within a network partition compared to that in the whole network, is used for improving the hit rate of cache. The Give-and-Take (GT) criterion is proposed in [49] for addressing the issue of free riders in network caching. Free riders are selfish peers who only obtain objects and leave the network without uploading anything in return. The mechanism in [50] proposes a distributed mobile caching system to cache data temporarily in a designated local area. The caching system is realized using collaborative consumer devices. The mechanism in [51] proposes a cooperative caching solution for vehicular network that formulates and solves an optimization problem to maximize content dissemination among vehicles within a predetermined deadline. The proposed approach minimizes the cost associated with communicating over the cellular connection. In another work, [52] presents a cooperative caching strategy in which the proposed protocol uses a class of reputation-based data forwarding and caching heuristics where the forwarding and caching decisions maximize the performance of the global system. It also proposes a Heterogeneous Community-based Random Way Point (HC-RWP) mobility model, which captures the properties of real human mobility. In [53], a cooperative caching solution for sensor networks is proposed in which some of the chosen sensor nodes are selected to maintain special roles in taking the caching and request forwarding decisions. The scheme in [54] presents a scheme that selects appropriate nodes as Network Central Locations (NCLs) to coordinate multiple caching nodes to optimize the tradeoff between data accessibility and caching overhead. In [55] a strategy based on a P2P network, named Small World Network (SWN), was proposed. In SWN,

the correlation of demands across different video segments are used to guide nodes to form a cooperative caching overlay so that each node maintains neighborhood relationship with related video segments. In [56], a quality of experience (QoE) centric distributed caching approach was proposed to improve the cache hit ratio for requests with specific quality. The work in [57] proposed a new QoS-aware hierarchical web caching (QHWC) scheme for Internet-based vehicular ad hoc networks (IVANET). This scheme leverages any locality in vehicle dis-connectivity and mobility in order to increase the cache hit ratio and to reduce query delays. In [58], a caching strategy named Chunk Select method Adaptive to Neighbor Reception (CSANR) was proposed for avoiding nodes to download the same segments of streaming videos in order to save the cache space. A video dissemination solution based on hybrid P2P/Multi-server Quality-Adaptive Live-Streaming is proposed in [59]. This approach combines P2P caching and multiple edge servers in order to reduce the traffic load of the Internet as well as improving users' QoE.

In vehicular networks, the authors [60] presents an in-vehicle caching method which is termed as Collaborative Caching Based on Socialized Relations (CCBSR). The objective is to mitigate the impacts of mobility-triggered V2I dis-connectivity towards minimizing content access delay. It also attempts to minimize the number of vehicles in which the content needs to be cached. The P2P Cooperative Caching (P2PCC) in [61] also attempts to mitigate the impacts of dis-connectivity using a Markov model-based caching approach. It does so by in-vehicle caching and sharing content across the V2V network. The researchers in [62] proposes a V2V caching strategy based on Evolutionary Game. The objective of the method is to avoid the free riders, which only obtain content from other nodes without contributing to caching. An innovative V2V caching mechanism in [63] attempts to mitigate the impacts of signal quality and connectivity impairments due to large buildings in urban settings. The paper proposes a Leave Copy Everywhere (LCE) strategy to

improve content accessibility in urban environments. A Community Similarity and Population-based Cache Policy (CSPC) is proposed in [64] for ICN vehicle-to vehicle (V2V) scenario. This method evaluates the community similarity and privacy rating of vehicles, and selects the caching vehicle based on content popularity to reduce the cache redundancy. Finally, [65] presents a cooperative content caching framework, and proposes a hierarchical mobility-aware edge caching scheme that harnesses the synergies between mobile edge computing (MEC), multi-BS caching, and vehicular caching.

2.3 Summary

Almost all the mentioned approaches focus on maximizing the cache hit rate and reducing the access latency, without considering its effects on the overall cost which depends heavily on the content service and pricing models. Specifically, for caching of streaming content most of the mentioned approaches do not consider caching videos with heterogeneous qualities in users' devices. In this thesis, various Device-to-Device cooperative caching strategies are proposed to address these gaps. These caching strategies are inspired by the notations of: 1) minimizing the overall content provisioning cost, and 2) heterogeneous user preferences.

Moreover, in the context of vehicular networks most of the existing vehicular caching work attempts to improve content accessibility and reduce the access latency, without considering minimizing the bandwidth usage of cellular networks. In this thesis, the primary objective is to minimize the access cost by the way of reducing the usage of in-vehicle cellular links. While accomplishing that, the accessibility and access delay are also improved.

Finally, the caching mechanisms proposed in the above literatures are based on the traditional edge infrastructures (i.e. RSU, etc.) with backhaul connectivity, or in-vehicle caching. However, the deployment cost of such infrastructure is expensive. On the other hand, the in-vehicle caching

does not work when the vehicle density is low. To deal with these challenges, this thesis proposes a novel edge infrastructure CECS introduced in Section 1.1.3 in Chapter 1 that has no backhaul connectivity.

Chapter 3: Cooperative Caching in Social Wireless Networks

3.1 Introduction

Wide popularity of wireless devices and their data-enabled applications have created an evolving marketplace for digital content ecosystems. A common operation in those ecosystems is to disseminate content (e.g., books, magazines, music, etc.) in a cost-optimal manner. With the conventional download model, a user downloads content directly from a Content Provider's (CP) server via a Communication Service Provider's (CSP) network. Downloading content through CSP's network involves a cost, which must be paid either by End Consumers (EC) or the CP.

In this chapter, an incentive based cooperative content caching framework is developed for fully-connected Social Wireless Networks (SWNETs) in which content demands are hierarchically heterogeneous. The heterogeneous request model incorporates user preference for different categories/genres, and contents under each category, both following power law distributions at local as well as global levels. Based upon such request generation model, an optimal incentive based Heterogeneous Split Caching algorithm is proposed which can minimize electronic content provisioning cost using cooperative caching policies.

3.2 Hierarchically Heterogeneous Requests

Each object (i.e. content) is assumed to be tagged with its global-popularity by a centralized content server such as Amazon or iTunes. The global popularity rank order is determined based on the network-wide request rates for all the content stored in the server. The larger the global popularity of an object is, the more likely it is to be requested across the entire network.

Each object is labeled with a category or genre at its creation time. One practical example is the list of categories for Amazon audio books, or the list of categories for songs provided by Spotify. Typically, a user has her/his own preference of such categories, which is maintained in

terms of a rank ordered list of categories. This user-specific list is referred to as content local popularity for that user. The content request model considers both local and global popularity models which are assumed to follow Zipf distributions. For global popularity, Zipf is applied at the per-content level, and for local popularity, it is applied at per-category level.

3.2.1 Global Popularity of Object

Global popularity of an object- i (i.e., O_i) can be expressed as the probability that any random request from the network is for O_i . According to the Zipf law, it is expressed as follows:

$$p_G(O_i) = \frac{\left(\frac{1}{i}\right)^\alpha}{\sum_{l=1}^L \left(\frac{1}{l}\right)^\alpha} \quad (3-1)$$

The parameter L represents the total number of objects in the network. The Zipf parameter, α , determines the steepness of the distribution curve. The higher the Zipf parameter α , the larger the difference of global popularity value for two objects with consecutive ranks.

3.2.2 Local Popularity of Category

A user's local object preference is determined by the local popularity of the object categories. In this chapter, it is assumed that the global popularity of these categories follow uniform distribution. However, for each node, the local popularity of the categories can be different. A user's local object preference is represented as a rank-ordered list of M object categories. That is one of $M!$ possible lists. The rank of a specific category for a user- n can be different from the rank of the same category for another node in the network. The local popularity of a category- j at node- n can be expressed as the probability that category- j has a popularity rank of k out of all M categories. According to the Zipf law, it is expressed as follows:

$$p_n(C_k^j) = \frac{\left(\frac{1}{k}\right)^\alpha}{\sum_{m=1}^M \left(\frac{1}{m}\right)^\alpha} \quad (3-2)$$

The α in Zipf can be different for the global and local popularity distributions. While global popularities are defined for each object, the local popularities are for each category.

3.2.3 Local Popularity of Object

An object- i that belongs to category- j is represented as $O_i^{C_k^j}$, and its global popularity $p_G(O_i^{C_k^j})$ is determined based on Eqn. 3-1. The quantity $p_G(O_i^{C_k^j})$ indicates the global popularity rank of the object across all other objects. The rank of $O_i^{C_k^j}$ among all other objects belonging to C_k^j can be computed as:

$$r(O_i^{C_k^j}) = \frac{p_G(O_i^{C_k^j})}{\sum_{m=1}^{|C_k^j|} p_G(O_m^{C_k^j})} \quad (3-3)$$

$|C_k^j|$ indicates the total number of objects in category- j , and $r(O_i^{C_k^j})$ represents how popular object $O_i^{C_k^j}$ is among the members of category- j . We then define the local popularity of $O_i^{C_k^j}$ at node n , in which category- j has a local popularity rank of k out of all M categories, for node n . This local popularity of object $O_i^{C_k^j}$ at node n is computed as:

$$p_n(O_i^{C_k^j}) = r(O_i^{C_k^j}) \times p_n(C_k^j) \quad (3-4)$$

The quantity $p_n(C_k^j)$ is the local popularity of category- j as defined in Eqn. 3-2. Note that this local popularity of a category at a node is different from the local popularity of an object at a node, as newly computed by Eqn. 3-4.

3.2.4 Request Generation Process

Using the above hierarchical model, requests are generated from a consumer using the following process. First, a category is selected from the available M categories using the Zipf based local popularity distribution for the specific consumer. After a category is selected, one object is chosen from that category based on the global popularity Zipf distribution. Objects within a category with higher global popularity are more likely to be chosen. Since the local popularity of categories for each user is different, the ranking order of popularity of categories in the Zipf distribution is different for each user. Therefore, different users in the SWNET generate different request rates for same content, thus reflecting heterogeneity.

3.3 Network Model and Problem Formulation

3.3.1 Content Search Model

Upon originating a request, a node first locally searches in its own cache. If that fails, the node performs a remote search in its SWNET partition. Note that in this chapter it is assumed that all the nodes in an SWNET partition are fully connected. If the node cannot obtain the object from the SWNET, a request is originated to the CP's server to obtain the content. The local search can take place using devices' Wi-Fi, Bluetooth, and other near-field RF interfaces.

3.3.2 Pricing Model

The pricing model in this chapter is similar to Amazon Kindle's business model described in Chapter-1. Figure 3-1 depicts the adopted content and cost flow model. When the content is downloaded from CP (e.g. Amazon), a cost C_d must be paid to the CSP by the CP for using its services, which accommodates for the service given from the CSP to CP and the service given to nodes from the cellular infrastructure. In order to encourage the nodes to cooperate in caching by

sharing their resources (e.g. energy, storage, etc.), CP pays a rebate C_r to the participant nodes (e.g. EC_A) when a cached object is transferred from the node to a requester (e.g., EC_B) in the SWNET. The $\frac{C_r}{C_d}$ ratio, namely, β should be in range $[0,1]$, so that the total cost can be reduced by caching the most popular content remotely in the SWNET. Since C_d and C_r are set by a CP according to CP and CSP's marketing and revenue models, the ECs do not have any control on the parameter β . Generally, it would not entice the ECs to participate in the cooperative caching when β is too low. On the other hand, a too high β does not provide significant cost savings.

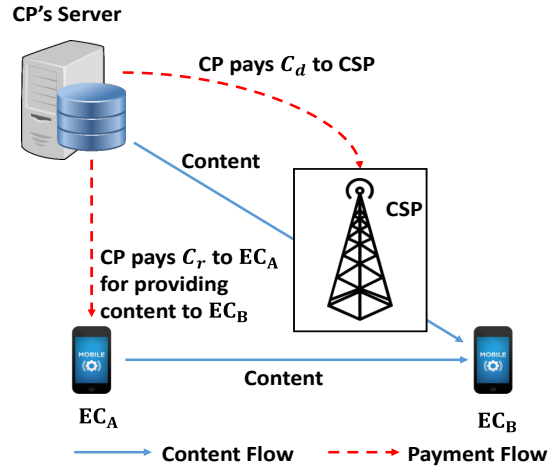


Figure 3-1: Content and cost flow

Note that the cost items C_d and C_r are independent from the price of content (e.g., an e-book) that the EC pays to the content provider, preferably using a secure payment system.

3.3.3 Cost under Heterogeneous Request Model:

When a content is found locally within the requesting node, no cost is incurred for the content provider. Otherwise, the costs are C_r or C_d depending on if the content is downloaded from another node within the SWNET or from the CP's server. The expected cost for a node n is computed as:

$$cost_n = 0 \times \sum_{O_i^{C^j} \in L} p_n(O_i^{C^j}) + \beta \times C_d * \sum_{O_i^{C^j} \in R} p_n(O_i^{C^j}) + C_d \times$$

$$\sum_{O_i^{C_k^j} \in \neg(L+R)} p_n \left(O_i^{C_k^j} \right) \quad (3-5)$$

Where L is the set of contents stored locally at a node; R is the set of contents stored in cache of other nodes in the SWNET ($L \cap R = \emptyset$). The network-wide total cost of provisioning in a network with size N can then be computed as follows:

$$cost_{total} = \sum_{n=1}^N cost_n \quad (3-6)$$

3.3.4 Problem Definition

For a given combination of C_r, C_d , and a heterogeneous request model as presented in Section 3.2, in a network of size N , the objective is to design a cooperative caching mechanism which minimizes the total cost of provisioning as stated by $cost_{total}$.

3.4 Heterogeneous Split Caching Algorithm

The proposed Heterogeneous Split Caching (HSC) algorithm is designed for minimizing the provisioning cost under the proposed heterogeneous request model. Each node's cache is divided into two parts shown as Figure 3-2: 1) local segment, which is λ ($0 \leq \lambda \leq 1$) fraction of the available storage, and 2) global segment, which is the rest of the storage. In the local segment, nodes store objects that are locally most popular for the node, while in the global segment nodes cache objects that are globally most popular.

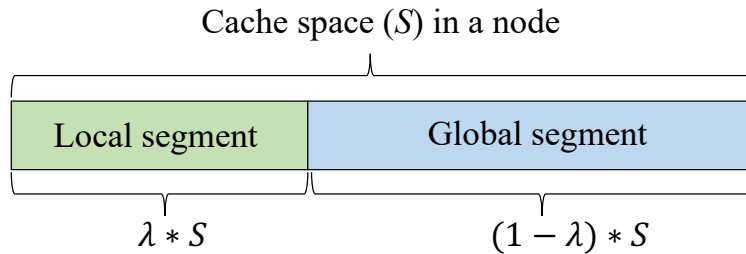


Figure 3-2: Cache Partitioning in HSC

Information Availability: CP maintains information regarding number of requests made for any object, with which it can compute the global popularity ($p_G(O_i^{c_k^j})$) using Eqn. 3-1, and the popularity of the object among all the members of its category ($r(O_i^{c_k^j})$) using Eqn. 3-3. Each node only needs to maintain its personal preference for categories. Once an object labeled with $p_G(O_i^{c_k^j})$ and $r(O_i^{c_k^j})$ is downloaded, the node-specific local popularity of the object is computed based on Eqn. 3-4 locally by the node. In the global segment, globally popular unique objects are stored. An object's global popularity ($p_G(O_i^{c_k^j})$) follows the assigned Zipf distribution from Eqn. 3-1, and is known by nodes when the object is downloaded.

Uniqueness of objects stored in the global segment is required to guarantee maximum content diversity, and to avoid network-wide duplications of globally popular objects. It means that the objects in the local segment can be network-wide duplicated. If S is the per-node cache/storage size, local segment occupies $\lambda * S$ portion of the cache, which leaves $(1 - \lambda) * S$ for the global segment. The objective of the HSC algorithm is to find the optimal λ such that the total provisioning cost as defined in Eqn. 3-6 is minimized.

Upon receiving a content (O_{new}) either remotely from its Social Wireless Network (SWN) or from the CP, the requester node- i stores it in its local segment of its cache. If the local segment is full, it follows a replacement policy as follows.

The node first compares the local popularity of the obtained object O_{new} with the popularity of the least locally popular content $O_{l_{min}}$ in its local segment. If $p_i(O_{new}) > p_i(O_{l_{min}})$, $O_{l_{min}}$ is replaced with O_{new} . Otherwise, following happens.

1) If O_{new} is obtained from the SWNET, the content is dropped. This is since the object is less popular locally compared to other objects existing in the local segment, it is not going to be requested as frequently. Therefore, it is reasonable to obtain it remotely from the SWNET, anytime it is requested again.

2) If O_{new} is downloaded from the CP, the node stores it in the global segment. If that segment is full, the node compares the global popularity of O_{new} with that of the least globally popular object O_{g_min} in its global segment. If $p_G(O_{new}) > p_G(O_{g_min})$, O_{g_min} is replaced by O_{new} . Otherwise, the object is dropped. Storing a downloaded content in the global segment improves the network-wide content diversity. By storing the content in at most one node, the object becomes accessible to other SWNET nodes, thus eliminating cost of future downloads.

The full logic of caching and replacement is summarized in Algorithm 3-1. The optimal solution which results in the minimum provisioning cost can be achieved when lambda is set at the optimal lambda, which is found experimentally.

Since the values of $p_G(O_i^{c_k^j})$ and $r(O_i^{c_k^j})$ are independently computed by the CP, and delivered as a part of the host object $O_i^{c_k^j}$, a node can compute its specific local popularity for an object within the time complexity of $O(1)$. The main computation cost of the algorithm is for running the replacement policy, which could be $O(n)$ in the worst situation, where n is the local cache size.

```

1: Input: New Coming Content  $O_{new}$ 
2: if( $i.cache_{local}$  is not full) then
3:     store  $O_{new}$  in  $cache_{local}$ 
4: else
5:      $O_{l\_min}$  = least locally popular content in  $cache_{local}$ 
6:     if( $p_i(O_{new}) > p_i(O_{l\_min})$ ) then
7:         replace( $O_{new}, O_{l\_min}$ )
8:     else

```

```

9:         if( $O_{new}$  is obtained from other nodes) then
10:             drop( $O_{new}$ )
11:         else
12:             if( $i.cache_{global}$  is not full) then
13:                 store  $O_{new}$  in  $cache_{global}$ 
14:             else
15:                  $O_{g\_min}$  = least globally popular content in  $cache_{global}$ 
16:                 if( $p_G(O_{new}) > p_G(O_{g\_min})$ ) then
17:                     replace( $O_{new}, O_{g\_min}$ )
18:                 else
19:                     drop( $O_{new}$ )
20:                 end
21:             end
22:         end
23:     end
24: end

```

Algorithm 3-1: Caching algorithm and replacement policy in HSC

3.5 Performance Evaluation

Using a Java based simulator ONE [66], performance of the following two caching algorithms is evaluated in a static fully-connected network of 1000 nodes requesting similar-size objects. The optimal λ that minimizes the provisioning cost is chosen experimentally in various scenarios since there is no closed-form expression of it.

Baseline HSC: Each node in this case starts with an empty cache and then the cache replacement process follows the algorithm presented in Section 3.4.

HSC with Cache pre-filling: This represents a network-wide steady state of cache distribution, when the baseline HSC is applied for an infinite number of requests. Such steady state is emulated by prefilling each node's cache space (by the content server) with $\lambda * S$ number of locally popular objects for the node. Node's global cache segment is filled with $(1 - \lambda) * S$ unique and globally popular objects. Note that the objects stored in local segments are duplicated across multiple nodes. However, objects in a node's global segment needs to be network wide unique. No replacement is executed since pre-filling represents a desirable network wide steady state.

Pre-filling is not scalable and realistic. It, however, provides the benchmark results, that is, the best-case performance corresponding to when the baseline HSC is executed over long duration.

Parameter	Default Value
Number of Nodes	1000
Number of Categories	5
Zipf Parameter α	0.8
Object Population	100000
Total Simulation Duration	500000 Requests
Download Cost	10
Ratio of Rebate to Download Cost β	0.6
Node's Cache Size	50 Objects

Table 3-1: Simulation's baseline parameter

Unless stated otherwise, all parameters are set to baseline values as shown in Table 3-1. The default α is set as 0.8 [25], and number of objects is set to 100 thousand which is sufficiently larger than the total number of objects which can be stored throughout the network (i.e. 50000). Simulation is stopped when 500 thousand requests are generated. Since the popularity (i.e., probability of being requested) of the lowest popular object is 2×10^{-6} , such number of requests ensures that every object be highly likely requested at least once in the network. Each node's cache size is set such that it can accommodate 50 objects.

3.5.1 Impacts of Zipf parameter (α) and Rebate cost ratio (β)

The U-shape trend for the cost in both graphs of Figure 3-3 indicates that there is an optimal cache split factor λ that minimizes the provisioning cost. The figure shows the total cost of provisioning for two prefilling and baseline scenarios as well as the theoretical cost of provisioning all objects for all nodes computed based on Eqn. 3-5 and 3-6. When λ is zero, the entire cache in a node contains globally popular objects. $\lambda=1$ corresponds to when every node's cache contains only locally popular objects for that node. The cost in pre-filling case is very similar to the cost computed based on Eqn. 3-5 and 3-6, since the pre-filling scenario represents the network-wide steady state

of caching, and also the best performance under each set of parameters. On the other hand, the cost of pre-filling case is always less than the baseline HSC except for when $\lambda=1$. The reason for this exception is as follows. Before convergence, caches still hold objects, which are globally popular. When $\lambda=1$, the whole cache is supposed to maintain locally popular objects. The existence of globally popular objects in the cache results in an increase of remote hit rate, which causes a lower cost for HSC compared to pre-filling case.

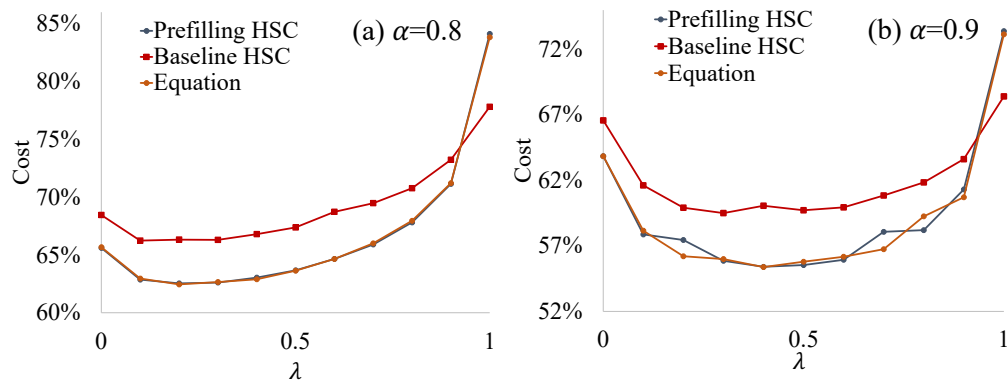


Figure 3-3: Impact of α on cost

Compared with $\alpha = 0.8$, the costs for both caching methods decrease when $\alpha = 0.9$. This is because with increase in α , the difference of popularity for every pair of objects with consecutive ranks increases in Zipf for the global popularities. That is, the popularity gap between the few top objects and the lower rank objects increases. That results in more frequent requests be targeted at top popular objects. Thus, with larger α , caching such objects is more beneficial and it reduces cost.

The local and remote hit rates for the baseline HSC is shown in Figure 3-4 for various α . With a fixed α , a larger λ results in higher local and lower remote hit rates. That is because with large λ , the local segment's size increases, leaving less space for globally popular objects, which would have contributed to higher remote hit rates.

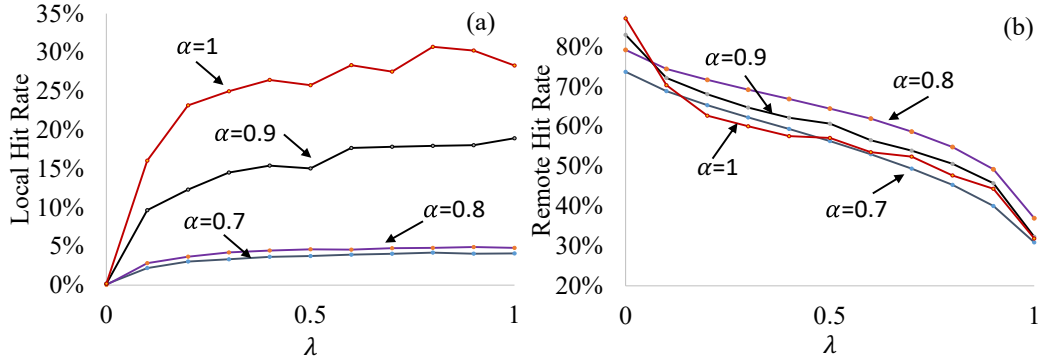


Figure 3-4: Hit rates for baseline HSC

With large α , local hit rate increases more drastically, since the generated requests are more targeted at top globally popular objects. The optimal strategy is then to increase the size of the local segment (i.e. larger λ) to store such objects locally.

Figure 3-5 depicts the impacts of β on the provisioning cost and the optimal λ . Higher β represents higher cost of getting content from in-SWNET nodes. This explains why higher β increases the overall cost. On the other hand, it can be seen that the optimal λ is 0 when $\beta = 0$ (rebate cost $C_r = 0$). That is because the best caching strategy is to cache objects as diverse as possible throughout the network when the cost for remote hit is same as the cost for a local hit. The provisioning cost is minimized by caching the locally popular objects (i.e. the optimal $\lambda = 1$) when $\beta = 1$. That is because it is not beneficial for nodes to store globally popular objects when the rebate C_r is same as the download cost C_d . For $0 < \beta < 1$, larger β results in larger optimal λ , which means caching globally popular objects is less beneficial when C_r increases.

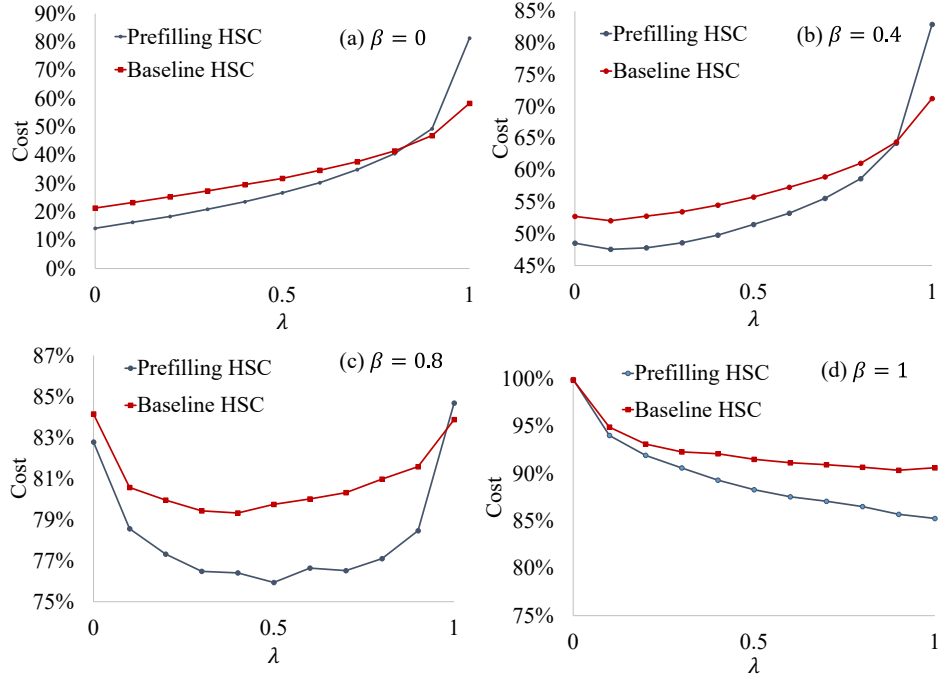


Figure 3-5: Impact of β on cost

3.5.2 Comparison with Traditional Caching Strategies

Provisioning costs corresponding to traditional caching algorithms, including Least Frequently Used (LFU), Least Recently Used (LRU), and Random replacement are evaluated.

As observed in Figure 3-6, both versions of HSC incur the lowest cost, while the Random method holds the highest cost, and the costs of LFU and LRU are very similar. As expected, with increasing β , the provisioning cost for all the protocols increase. Also, the cost difference between the HSC algorithms and the traditional ones increase. This is because with larger rebates, it is more beneficial to cache locally popular objects for which HSC is more efficient than the traditional algorithms.

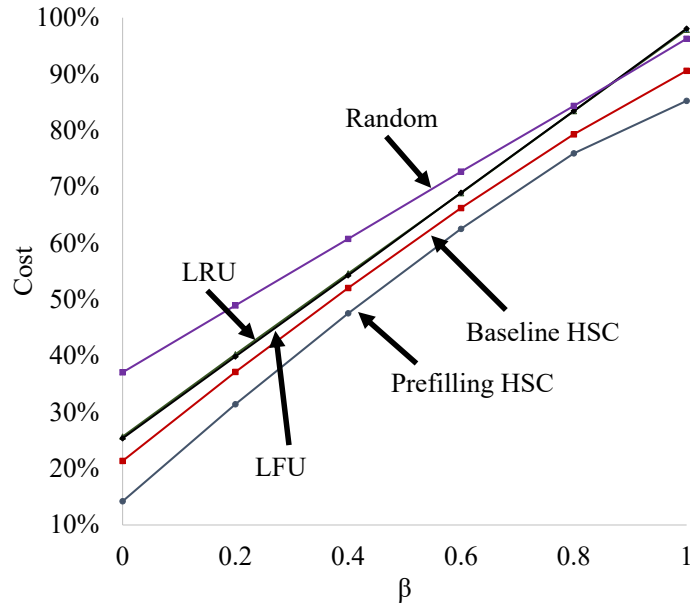


Figure 3-6: Comparison of minimum cost

3.5.3 Object Density

Object density is defined as the number of copies of an object maintained within the network. Figure 3-7 shows the density of the top 50 globally popular objects. The ID of each object represents its global rank (i.e., popularity). For both pre-filling and baseline versions of HSC, for $\lambda = 0$, the object density for all objects is 1. That is because for all nodes, the whole cache is assigned to the global segment. Since the global segment of the cache in a node stores only unique objects, only one copy of such an object can be stored in the entire SWNET. When $\lambda > 0$, with pre-filling (Figure 3-7 (a)), the density of most popular objects equals the total number of nodes, which is 1000. That means, there is a copy of each object in each node's local cache segment. Note that the local popularity of an object is a function of its global popularity and the local popularity of the category to which that object belongs to. When an object's global popularity is very large, irrespective of the local popularity of its category for a node, its local popularity for the node is also generally high.

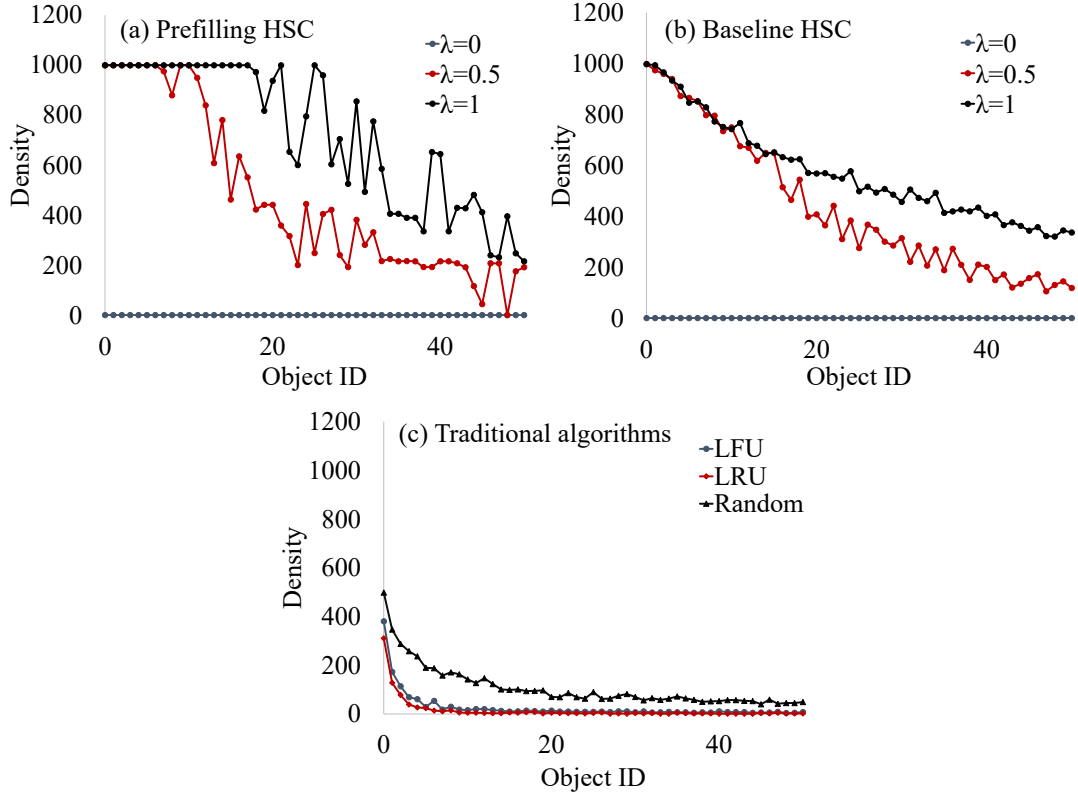


Figure 3-7: Object density of various algorithms

However, for some nodes, the local popularities of some objects with low global popularity is high because of the heterogeneity, which is also the reason for the fluctuation happening when the density starts to reduce. Since $\lambda = 1$ assigns the whole cache to the local segment, it can store more globally popular contents across nodes in the network, compared to for example when $\lambda = 0.5$. The decreasing trend of density shows that generally objects with higher rank (global popularity) are requested more often, thus, stored in the SWNET.

Density curves for the baseline non-prefilling HSC have similar trend. The differences in density, especially for top globally popular objects, are caused by fewer requests in the baseline HSC compared to the infinite request steady state for the pre-filling HSC case.

For traditional caching algorithms, the density trend looks more long-tailed and similar to the Zipf distribution for the global popularity. That means, each object is cached according to its global popularity. Since there is no distinction between the global and the local segments, the density

3.5.4 Convergence of Baseline HSC to Prefilling HSC

To demonstrate how baseline HSC results convergence to that for the prefilling version for infinite number of requests, simulations with different number of requests over various λ values are run. Figure 3-8 shows the cost for the two protocols. As mentioned, baseline HSC achieves a similar cost to pre-filling HSC as the number of requests is large enough (e.g., larger than 4 million). Note that, for each number of request scenario, the dip in the curves indicate the optimal λ at which the cost is minimized.

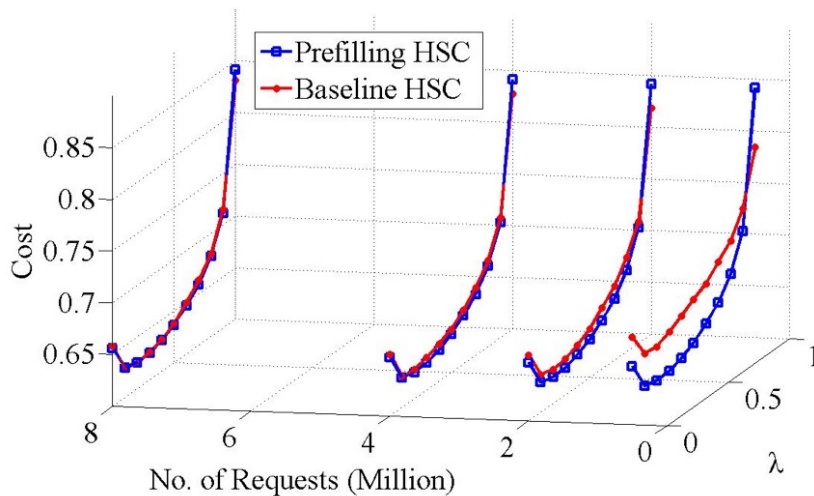


Figure 3-8: Convergence of cost for HSC baseline to the pre-filling scenario

3.6 Summary

A heterogeneous request generation model is proposed which is inspired by hierarchical user preferences for electronic content. The model is heterogeneous in that each user's preferences for genre/category and contents under those preferred categories are different. Based on this request

generation model, a Heterogeneous Split Caching (HSC) which incentivizes cooperative users in caching and aims at minimizing the network-wide provisioning cost is proposed. Simulation results indicate that the proposed mechanism is able to reduce content provisioning cost compared to traditional caching mechanisms. It also shows that after large number of content requests when the network caches reach a steady state, the mechanism can achieve cost minimization bounds at the same level offered by a benchmark strategy.

Chapter 4: Distributed Caching in Mobile Wireless Networks

4.1 Introduction

In this chapter, the caching mechanism proposed in Chapter-3 is applied on the scenario of mobility wireless networks. A typical example of such scenario is the vehicular networks formed by moving vehicles that connect to each other through V2V links. Unlike fully-connected networks in Chapter-3, in a mobility network the frequent disconnections caused by nodes' (e.g. vehicles, etc.) mobility influence the level of nodes' cooperation in caching. However, the experiments show that the proposed mechanism is still able to reduce bandwidth usage and the resulting content provisioning cost compared to traditional caching mechanisms in both monolithic and community-based mobility scenarios.

4.2 Content Search with Tolerable Access Delay (TAD)

Even though the network and pricing models applied in this chapter are the same as in Chapter-3, the content search model is extended for mobile networks. Upon originating a request, a node first performs a local search for the requested content in its own cache. If that fails, the node performs a remote search in the mobile network (e.g., a connected vehicle fabric). If the node cannot obtain the content from the network within a pre-defined Tolerable Access Delay (TAD), it sends a request to the CP's server. TAD, which is part of a request, represents the duration that a user application is willing to wait before a successful content request is served. The remote search can take place using devices' Wi-Fi, Bluetooth, and other short-range RF interfaces (e.g. DSRC, etc.). The content from the CP's server is downloaded using a CSP's cellular network.

4.3 Heterogeneous Split Caching (HSC) for Mobile Networks

In this section, the HSC proposed in Chapter-3 is specifically extended to be used in a mobile network. Frequent disconnections caused by nodes' mobility influence the level of nodes'

cooperation in caching. The objective of the HSC algorithm is to find the optimal caching strategy for mobile networks with various connection density.

In a fully-connected network described in Chapter-3, ideally, the global segment in HSC of the nodes should contain objects as diverse as possible. This is to prevent network-wide duplications of globally popular objects. Though, this approach is neither optimal nor feasible in a mobile network due to frequent disconnections, where a node has limited number of encounters during TAD. Under such conditions, replicating globally popular objects into different caches may be a more efficient solution. Also, downloading an object from CP's server does not necessarily indicate that no other node in the network holds the object in its global segment. It implies that the requester has not succeeded in meeting the specific node with the object during the TAD. Therefore, in a mobile network, it is not guaranteed that objects in global segments of nodes are unique. However, it is guaranteed that at each point of time, an object is downloaded and stored in global segment of a requester, only if, its neighbor nodes do not carry it. Objects in the local segment can be network-wide duplicated. If S is the per-node cache/storage size, local segment occupies $\lambda * S$ portion of the cache, which leaves $(1 - \lambda) * S$ for the global segment. The objective of the HSC algorithm is to find the optimal λ such that the total provisioning cost as defined in Eqn. 3-6 in Chapter-3 is minimized.

4.4 Performance evaluation

Using the simulator ONE, performance of the following two caching algorithms is evaluated in a 100-node mobile network under scenarios with or without community-based movement, and also a scenario of Static Fully-Connected Network (SFN) for comparison.

Baseline HSC: Each node in this case starts with an empty cache and then the cache replacement process follows the algorithm presented in Section 3.4 in Chapter-3.

HSC with Cache pre-filling: This represents an ideal cache structure when it is assumed that the network is so sparse that the nodes' remote hit rate is near zero. With pre-filling, each node's local cache segment is filled with $\lambda * S$ locally popular objects. The global segment is filled with duplicated $(1 - \lambda) * S$ top globally popular objects. The logic behind this is that in a sparse network, without server's involvement, it cannot be ensured that unique objects are stored in global cache segments. Due to disconnections, multiple nodes may download an object from the CP's server without knowing that another node in the network carries it. No replacement is executed since pre-filling represents the ideal cache state. Pre-filling is not scalable, though it is supposed to show the best-case performance, corresponding to the extreme situation in which the baseline HSC is executed over long time, in presence of frequent disconnections.

Unless stated otherwise, all parameters are set to baseline values as shown in Table 4-1. The default α is set to 0.8 , and number of objects is set to 10 thousand which is sufficiently larger than the total number of objects which can be stored throughout the network (i.e. 5000). Simulation is stopped when 50 thousand requests are generated. Since the global popularity (i.e., probability of being requested) of the lowest popular object is 2×10^{-5} , such number of requests ensures that every object be highly likely requested at least once throughout the network. 5 number of categories is chosen for the network size of 100 nodes. Parameters TAD, β and terrain size vary.

Parameter	Default value
Number of Nodes	100
Number of Object Categories	5
Zipf Parameter α	0.8
Object Population	10000
Simulation Duration	50000 Requests
Download Cost	10
Ratio of rebate to download cost β	0.8
Cache Size in Each Node	50 Objects
Transmission Range (e.g., Bluetooth)	10m

Terrain Size (e.g., University Campus)	800m×800m
Tolerable Access Delay (TAD)	300s

Table 4-1: Simulation’s baseline parameter

4.4.1 Monolithic Mobility

In this simulation, 100 nodes move randomly among 10 waypoints (points of interest). Figure 4-1 depicts how λ affects total provisioning cost under different terrain sizes (TS) for baseline HSC. The HSC algorithm is also run in a Static Fully Connected Network (SFN) to show protocol’s performance in absence of any disconnections. This is to investigate if and how mobility and resulting disconnections affect HSC’s behavior.

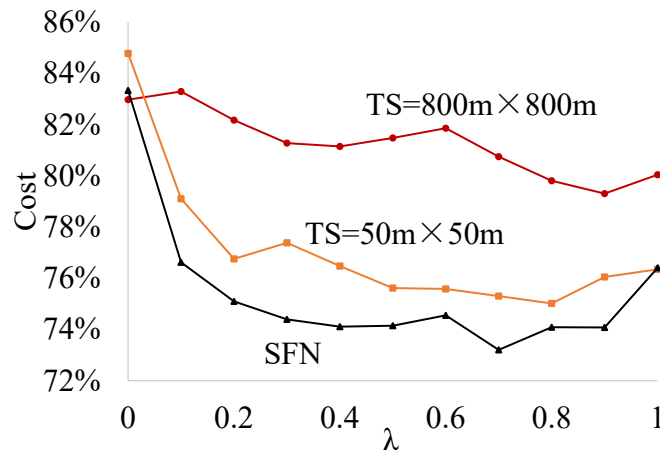


Figure 4-1: Cost V.S. λ for baseline HSC and SFN

The U-shape trend for the cost in both graphs of SFN and HSC (TS=50m×50m) indicates that there is an optimal λ that minimizes the provisioning cost. When λ is zero, the entire cache for a node contains globally popular objects. $\lambda = 1$ corresponds to when every node’s cache contains only locally popular objects for that node. When terrain size is as large as 800m × 800m, disconnections happen more frequently. Therefore, a requester node visits fewer nodes during the tolerable access delay (TAD) compared to when network is denser (for scenario TS=50m×50m). Under such conditions, the best cost-reducing strategy is to store as many locally popular objects

as possible, which happens when λ approaches 1. SFN maintains the lowest cost since it corresponds to a fully-connected network where the remote hit rate is supposed to be much higher compared to a mobile network with disconnections. For a similar reason (fewer disconnections and higher remote hit rate), the cost of small terrain size (i.e. $50m \times 50m$) is lower than that for the larger size network (i.e. $800m \times 800m$), except when $\lambda=0$. Corresponding average local and remote hit rates over all nodes are shown in Figure 4-2. As mentioned, remote hit rate for SFC and the smaller terrain size ($50m \times 50m$) is lower than the remote hit rate for larger size networks (Figure 4-2 (b)).

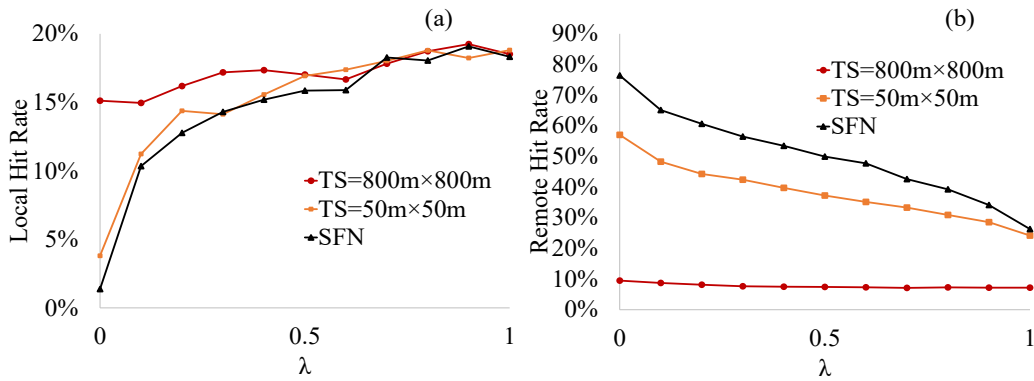


Figure 4-2: Hit rates for baseline HSC and SFN

The reason for a higher cost of provisioning in the $TS=50m \times 50m$ case, compared to the larger size scenario, when $\lambda=0$, is as follows. In a smaller size network, nodes have higher chance of obtaining objects remotely. With a rebate to download cost ratio 0.8, the involved rebate cost becomes so large, it results in a higher provisioning cost in general.

Lower local and higher remote hit rates can be observed for SFN and $TS=50m \times 50m$ scenarios in Figure 4-2 (a) and Figure 4-2 (b) respectively. That is, for these scenarios, the optimal λ is smaller compared to optimal $\lambda=1$ for $TS=800m \times 800m$ case. This represents a cache space, which can accommodate more globally popular objects, leading to a higher remote hit rate. Impact of λ on remote hit rate is not as clear when the terrain size is very large (i.e. $800m \times 800m$). That is because the network is so sparse that nodes can rarely meet each other during the TAD for any

values of λ . For $TS=800m \times 800m$ scenario, the local hit rate does not change significantly either. The reason is that nodes' global segments end up containing many duplicate objects, since each node individually downloads the content. Eventually, with any λ , each node carries most locally popular objects (in local segment), as well as most globally popular objects. This leads to a high local hit rate at most λ values.

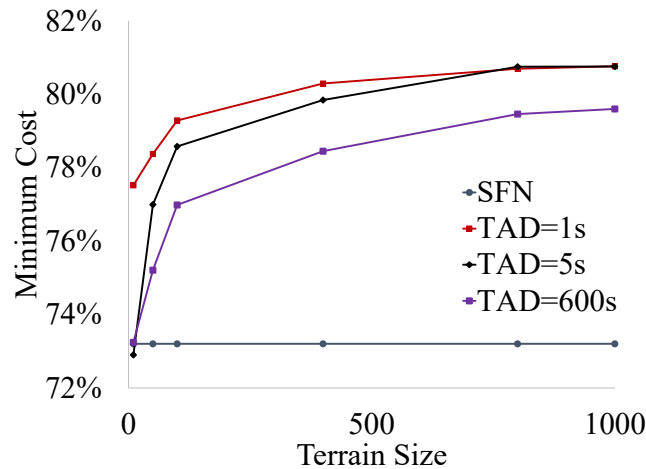


Figure 4-3: Minimum cost V.S. terrain size for baseline HSC and SFN

Figure 4-3 shows the impact of TAD on minimum provisioning cost for various terrain sizes. With an increase in terrain size, the minimum cost with different TAD increases, since the remote hit rate decreases in sparser networks. With longer TAD, the minimum cost decreases because nodes obtain objects remotely rather than through downloading. In the smallest terrain size (i.e. $10m \times 10m$) the minimum cost using baseline HSC is very close to SFN's cost when TAD is larger than 1s. This is because high node density in a small terrain size provides near full connectivity, as in the SFN case.

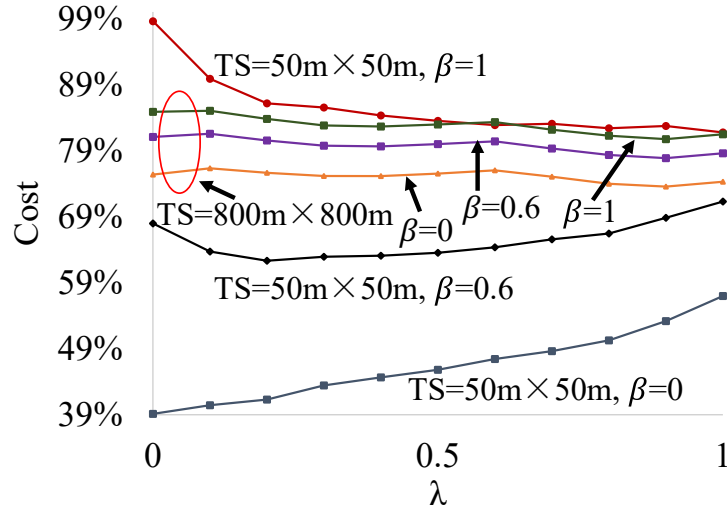


Figure 4-4: Cost V.S. λ for baseline HSC with various β and terrain sizes

Figure 4-4 depicts the effects of both β and terrain size on the optimal λ . For both large and small terrain sizes, the provisioning cost increases with an increase in β . This is because the cost of rebate grows larger with β . It can also be observed that generally smaller terrain sizes result in lower provisioning cost. Only for $\beta = 1$, the provisioning cost in $TS=50m \times 50m$ scenario is larger than that for $TS=800m \times 800m$, especially for smaller λ 's. The reason is the larger gap between the corresponding remote hit rates, as shown in Figure 4-2 (b). For an expensive rebate cost ($\beta=1$), higher level of nodes' cooperation in the smaller terrain size ($TS=50m \times 50m$) results in large rebate and subsequently higher provisioning cost. In a smaller terrain size, ($TS=50m \times 50m$), optimal λ is clearly affected by β . For example, the optimal λ equals 0 when $\beta = 0$, because the cost of rebate is 0, which makes remote hit rate more beneficial. On the contrary, when $\beta = 1$, optimal λ is 1 because providing rebate is as expensive as downloading content directly from the CP's server. It should be noted that β does not significantly affect the optimal λ when terrain size is larger (i.e. $800m \times 800m$). With low remote hit rate, the best strategy for minimizing the cost is to store as many locally popular objects as possible, irrespective of the rebate cost.

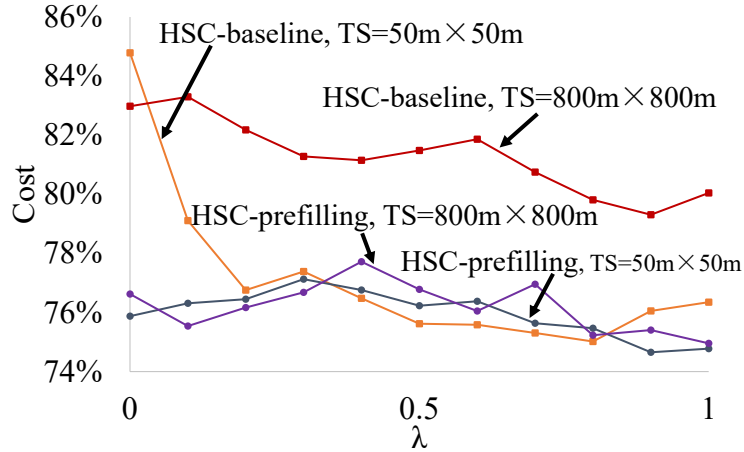


Figure 4-5: Cost V.S. λ for HSC-baseline and prefilling

Figure 4-5 shows the provisioning costs for both baseline HSC and prefilling for different network size. Cost for HSC-prefilling does not change significantly across terrain size. The global segment of all caches contains similar (and most globally popular) objects. Therefore, even for smaller terrain size, when nodes can contribute to cooperative caching, they cannot provide objects newer than what their peers already maintain.

The provisioning cost of prefilling scenarios is minimized around $\lambda = 1$. This is because the global section of cache in all nodes contains duplicate objects. Therefore, the remote hit rate is 0 and the most efficient strategy is to store as many locally popular objects as possible. For a small terrain size (i.e. $50m \times 50m$), the provisioning cost for HSC-baseline and HSC-prefilling are different only for smaller λ 's. Consider the $\lambda = 0$ case with entire cache consisting of only the global segments. For the prefilling scenario, remote hit rate is 0, since all nodes contain similar top globally popular objects. Though, for the HSC-baseline, because of high node density and distribution of globally popular objects across nodes, the remote hit rate, and its subsequent rebate cost leads to a higher provisioning cost. For larger λ 's, for which part of the cache consists of locally popular objects, the HSC-prefilling and baseline have similar cost because the local hit rate increases in both, and that reduces the effect of high rebate cost. This is the reason behind the larger

gap for smaller λ 's. When terrain size is large (i.e. $800m \times 800m$), the provisioning cost of baseline is significantly larger than that for prefilling for all λ values, because with sparser connectivity, prefilling provides a more efficient solution.

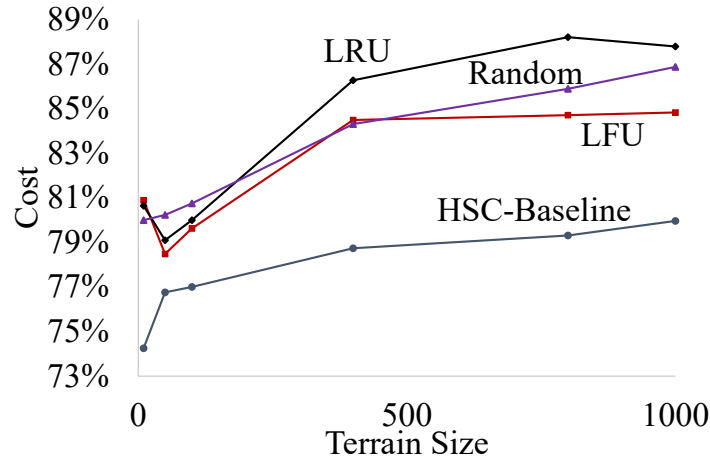


Figure 4-6: Comparison with traditional algorithms in monolithic scenario

Provisioning cost corresponding to traditional caching algorithms, including Least Frequently Used (LFU), Least Recently Used (LRU), and Random replacement are evaluated, and compared with optimal HSC. As observed in Figure 4-6, with larger terrain size, provisioning cost for all the protocols increase. HSC incurs the lowest cost for all terrain sizes.

4.4.2 Community-based Mobility

Here it is investigated if and how community-based movements along with community-based request model affects the cost results for the proposed caching algorithm. The population of 100 nodes are divided into 4 communities. Members of each community are similar in terms of the local popularity for object categories; thus, for the request generation model. They also follow a similar movement pattern as described below. Communities are formed based on nodes' local popularities for categories. Each node maintains a category preference vector, which defines the popularity and rank of each of the 5 categories for that node. For example, a node- i maintains a category preference

vector such as $\langle p_i(C_1^a), p_i(C_2^b), p_i(C_3^c), p_i(C_4^d), p_i(C_5^e) \rangle$. As an example, this vector indicates that category- a is the top most popular category for node- i with popularity $p_i(C_1^a)$. Based on the concept above, the preference similarity of each pair of nodes can be measured by Euclidian distance between their category preference vectors. Then the 100 nodes are divided into 4 groups of 25 nodes such that in each group the pairwise Euclidian distance between the category preference vectors are minimized. That is, nodes from one community are more similar in terms of category preference compared to nodes from two different communities. On the other hand, members of each community are imposed to have similar transition probabilities to few specific waypoints across 10 waypoints in a synthetic map. Therefore, intra-community contact frequency for each pair of nodes from the same community is higher than the contact frequency of each of the nodes with members of other communities.

Figure 4-7 depicts the provisioning cost of HSC-baseline under both community-based and monolithic mobility scenarios with different λ . Generally, the algorithm incurs smaller cost of provisioning in a community-based scenario. This is because, nodes from the same community have similar preferences and can lessen the total cost by participating in caching and providing content to members of the same community.

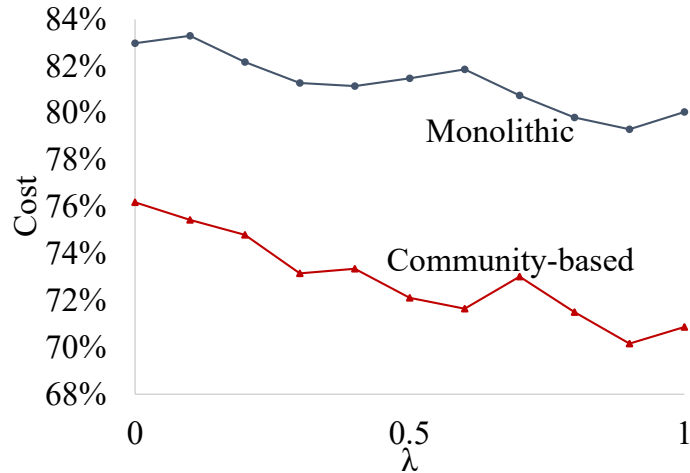


Figure 4-7: Cost V.S. λ for HSC baseline

The optimal λ for the community-based model is around 1 (similar to that for the monolithic mobility scenario in Section 4.4.1). This is because, it is more beneficial if nodes store locally popular objects which is likely to be requested by other nodes in the same community.

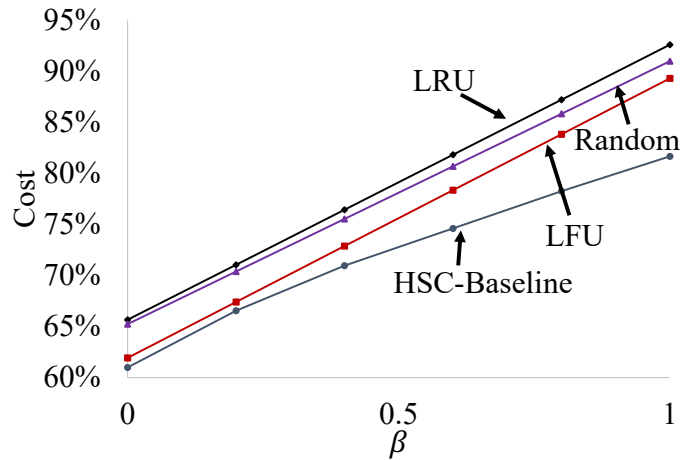


Figure 4-8: Comparison with traditional algorithms in community-based scenario

The provisioning costs for traditional caching algorithms under the community-based scenario are shown in Figure 4-8. With increasing β , costs for all protocols increase because of higher rebate. However, HSC incurs the lowest cost for all β . The cost differences between the HSC and traditional algorithms increase with an increase in β . This is because with larger rebates, it is

more beneficial to cache locally popular objects and perform cooperating caching, which is what HSC accomplishes.

4.4.3 Helsinki-map Mobility

Finally, we evaluate HSC and the traditional algorithms when nodes follow a random waypoint mobility model within the map of Helsinki as embedded in the DTN simulator ONE. This mobility scenario is different than the monolithic mobility presented in Section 4.4.1 that it simulates mobility of nodes in a larger city-scale terrain size ($4500m \times 3400m$) with transmission range set to $200m$. This complies with the transmission range in conventional Wi-Fi networks. Other parameters follow the baseline values shown in Table 4-1.

Figure 4-9 shows that similar to the other mobility scenarios, HSC incurs lower provisioning cost compared to traditional caching algorithms. In summary, HSC-prefilling provides lower cost than HSC-baseline for large terrain sizes, though, it is not as scalable since every node requires to pull content from CP's server upon joining the network.

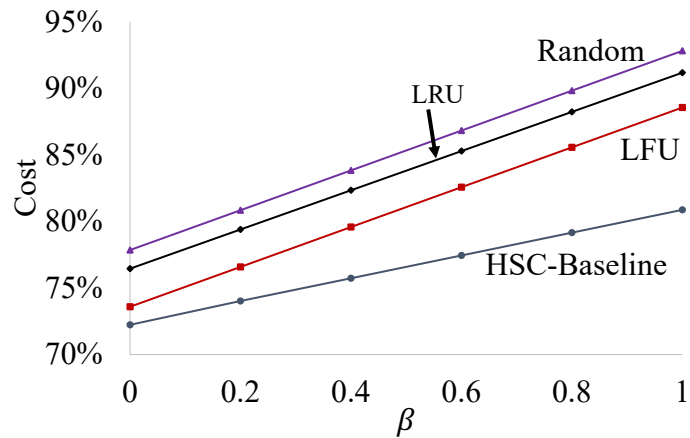


Figure 4-9: Comparison with traditional algorithms in Helsinki

4.5 Summary

The Heterogeneous Split Caching (HSC) mechanism proposed in Chapter-3 that aims at minimizing the network-wide provisioning cost is applied on the scenario of mobility networks.

Simulation results indicate that the proposed mechanism is able to reduce content provisioning cost compared to traditional caching mechanisms in both monolithic and community-based mobility scenarios.

Chapter 5: Caching for Streaming Video in Social Wireless

Networks

5.1 Introduction

In this chapter, a Device-to-Device cooperative caching framework is proposed for streaming video consumption by mobile users. For downloading video with cooperative caching, a mobile device first searches within its neighboring Bluetooth-connected peers for whole or parts of the requested video before downloading them from a Content Provider's (CP's) server. A specific cooperative caching framework, namely, *value-based* caching is proposed in which the value of caching a hierarchically coded streaming video segment is defined for given pricing and video sharing models. Within this framework, we develop an Adaptive Quality (AQ) provisioning algorithm that minimizes the overall video content provisioning cost incurred due to the bandwidth usage of a cellular network.

5.2 Content Search and Pricing Model

The similar content search and pricing models in Chapter-3 and Chapter-4 are still adopted in this chapter. However, for streaming video, the C_d and C_r in the pricing model are defined as the cost and rebate of unit data size such as per MB. Thus, the larger a video is, the larger the rebate will be for obtaining the video.

5.3 Streaming Video Play Model

5.3.1 Play Buffering and Caching

A generalized video play model similar to [67] is adopted in which a streaming video consists of multiple fixed duration segments. When an EC's device starts playing a video, it attempts to pre-fetch several segments to its play buffer from its local cache, or nearby ECs devices, or the CP

server via the CSP's network – in that order. When the play buffer gets full, the old buffered segments are replaced with new segments using a First-In-First-Out (FIFO) policy.

The play buffers are different from the cache. While the play buffer is used to smoothly play a currently available video by pre-fetching some of its segments, the cache is used to reduce the overall provisioning cost of that video by proactively storing its segments. Upon fetching a video-segment, a device puts it in its play buffer based on a FIFO policy, but it may or may not cache the segment depending on a separate caching policy as detailed in Section 5.4.

5.3.2 Video Play Model

When an EC's device starts playing a video, it first attempts to pre-fetch several of the video segments from the local cache to the play buffer. If any of those segments is not currently in the local cache, the device broadcasts a request for the absent segments to all other devices that it encounters during a Tolerable Access Delay (TAD). As defined in Chapter 4, TAD represents the duration a user is willing to wait for a successful segment retrieval from the network of peer user devices. If the segment is not found within the TAD period, the segment is downloaded from the centralized CP's server. The TAD for each video segment should be dimensioned based on the number and length (i.e. play duration) of its previous segments.

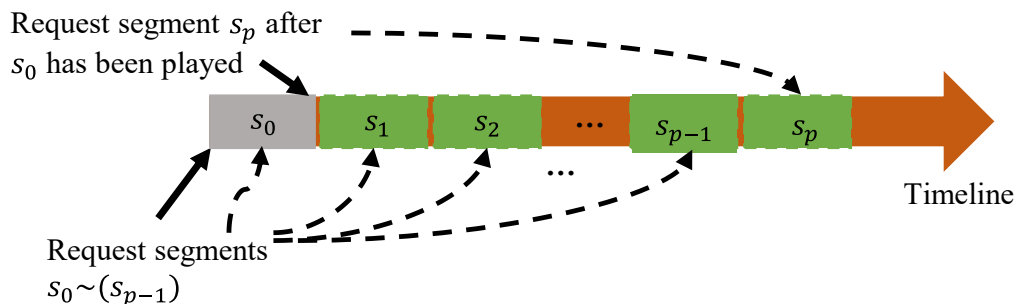


Figure 5-1: Timeline of video play model

Figure 5-1 depicts an example of a video play scenario from [67] in which a device first requests segments s_0 through s_{p-1} simultaneously to pre-fetch the next p segments. The TAD for each segment s_i can be computed as follows:

$$TAD_{s_i} = l_s(i - c) \quad (5-1)$$

where l_s is the length of each segment, and c is the sequence number of the segment that is currently being played. After *segment-0* is played out, *segment-p* is requested. This process continues until the video is fully sequentially played out, even when the user temporarily pauses the play. This is to ensure that the user should experience a smooth play when she/he chooses to resume from the pause. Following this strategy, the average TAD through all the pre-fetched segments in the buffer is computed as follows:

$$TAD_{ave} = \frac{\sum_{i=c}^{c+m} TAD_{s_i}}{m+1} = \frac{ml_s}{2} \quad (5-2)$$

A node can pre-fetch and store maximally $m+1$ next segments in the buffer. A larger m results in a higher TAD_{ave} .

In addition to sequential playing, a node can fast-forward or rewind a video on user requests. These actions are modeled with probabilities p_{ff} and p_{rew} for fast-forwarding and rewinding respectively. A node may also watch a video partially and switch to another one with probability p_{swi} .

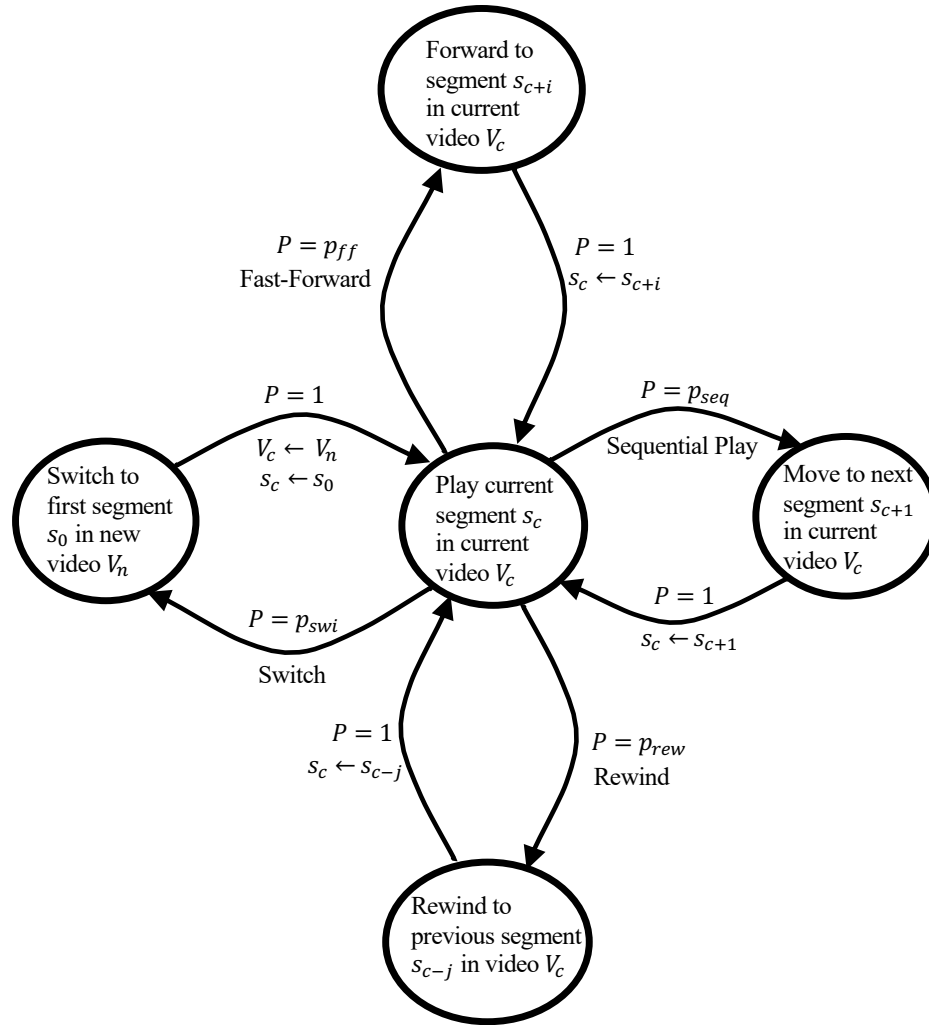


Figure 5-2: State machine of streaming video play model

The state machine [68] corresponding to these user actions is shown in Figure 5-2, in which the segment s_c in video V_c is the segment that is currently being played. After segment s_c is finished playing, the user may continue to view the video V_c (i.e., play segment s_{c+1}) with probability p_{seq} , or fast-forward to a later segment s_{c+i} ($c + i < Seg(V_c)$, where $Seg(V_c)$ represents the total number of segments in the video V_c), or rewind to a previous segment s_{c-j} ($c \geq j$). If the user switches to another video V_n , she starts the new video from its first segment s_0 . After completing each of these actions, the user starts playing the new segment (i.e. segment s_{c+1} , s_{c+i} or s_{c-j} in the current video V_c , or segment s_0 in another video V_n). In any of these cases except in sequential

playing, the requests for pre-fetching and their TADs are re-generated and re-computed. Note that by definition, $p_{ff} + p_{rew} + p_{swi} + p_{seq} = 1$. This video play model is used for the subsequent analysis.

5.4 Value-based D2D Caching

Building on the above video search, pricing, and play models, we propose a D2D cooperative streaming video caching mechanism in this section.

5.4.1 Value of Caching a Video Segment

Caching is performed at the level of video segments, which are also referred to as objects. The value of caching an object is defined as the provisioning cost that can be saved for future requests by caching the object. For example, if a node requests an object that has been stored in its local cache, the provisioning cost is zero. The value in this case would be the total savings due to caching, which is the cost of fetching the object from the CP using the CSP's network.

Value of Caching an Object Received from the CP's Server: According to the content search model, a requested object is downloaded from the CP's server only when it is not found in: a) the local cache of the requester node, or b) any other peer nodes the requester encounters within a pre-computed TAD duration. The latter means, either there is no copy of the requested video segment found in the encountered nodes' caches, or the available copies of the segment in the encountered nodes are encoded at a quality that is less than the requested quality. Let $Q = \{q_0, q_1, q_2, \dots, q_k\}$ be the set of possible video qualities that are provided by the CP, in which quality q_i is higher than q_j for any $i > j$. Hierarchical video coding [69] allows a video segment to be encoded at quality q_i to satisfy a request for the same segment at quality q_j , but not vice versa.

Let us consider the scenario in which node n downloads the object o_i with requested quality r_n ($r_n \in Q$) from the CP's server, and stores the object in its local cache. Also consider that this is

only one such copy among the nodes that node n has encountered so far (otherwise, instead of downloading it from the CP's server, the object can be fetched from a peer of node n which has such copy). Node n can receive the same object with no cost in the future since the object will be cached in its local cache. Therefore, the saved provisioning cost for node n because of caching object o_i with quality r_n in its cache is:

$$saved_n^{CP} = p_{o_i} C_d S(r_n, l_{o_i}) \quad (5-3)$$

where p_{o_i} is the popularity of object o_i . The streaming videos' popularities are assumed to follow a power law Zipf distribution, and each segment of a video has the same popularity as the parent video it belongs to. The higher the popularity, the more likely it is for it to be requested. Additionally, $S(r_n, l_{o_i})$ is the size of the object with quality r_n and duration l_{o_i} . The size of a video segment is a function of its quality and length. It should be noted that hierarchical video coding [69] allows object o_i with quality r_n to satisfy video requests for the same content o_i with any quality that is lower than or equal to r_n . As a result, when o_i with quality r_n is cached in node n , any request with quality r_n or lower from the peers of node n can obtain the object from node n without having to download it from the CP's server. Therefore, the provisioning cost in this case is only the rebate C_r to be paid to node n . This is instead of the downloading cost C_d , thus the saved provisioning cost for the peer nodes is:

$$saved_{peers}^{CP} = p_{o_i} (C_d - C_r) N \sum_{q_j=q_0}^{r_n} S(q_j, l_{o_i}) P(q_j) - p_{o_i} (C_d - C_r) S(r_n, l_{o_i}) \quad (5-4)$$

where N is the expected number of peers that node n encounters, and $P(q_j)$ is the probability that such peers' preference quality for object o_i is q_j . Such quality preference can be resulted by the limit of device resource, or the purchase price of the quality (e.g. the purchase price of the same video in HD may be lower than 4K), etc. Note that $q_j \leq r_n$ for all the q_j in Eqn. 5-4, and N , which is the total number of encountered peers, could be different over time since peers may join or leave

the network. The first term in Eqn. 5-4, namely, $p_{o_i}(C_d - C_r)N \sum_{q_j=q_0}^{r_n} S(q_j, l_{o_i})P(q_j)$ represents the saved provisioning cost when the nodes obtain the object o_i , encoded in quality no-higher than r_n , from the local network instead of from the CP's server. The second term $p_{o_i}(C_d - C_r)S(r_n, l_{o_i})$ is the saved cost generated by node n itself, which was already included in Eqn. 5-3. Therefore, the total value of caching object o_i with quality r_n downloaded from CP at node n is the sum of results from Eqns. 5-3 and 5-4:

$$value_{CP}(o_i, r_n) = saved_n^{CP} + saved_{peers}^{CP} \quad (5-5)$$

Value of Caching an Object Received from a Peer EC: Consider a situation in which a node n requests an object o_i with quality r_n and a specified Tolerable Access Delay (TAD). If there is at least one copy of the object cached in the peer nodes encountered by node n during the TAD, and the quality of the copy is no less than r_n , then the object o_i can be obtained from that peer node. Any subsequent request for the same object by node n would result in a local hit if node n caches the object. The resulting cost saving is the rebate C_r that needs to be paid to a peer node in case of no local caching in node n . This saving can be expressed as:

$$saved_n^{peer} = p_{o_i}C_rS(r_n, l_{o_i}) \quad (5-6)$$

Note that this local caching in node n does not affect the provisioning cost for the same object at other nodes since they still have to fetch it from a peer. Therefore, the saved provisioning cost for the other nodes except node n is 0. As a result, the overall cost saving or value is as follows, which remains the same as expressed in Eqn. 5-6:

$$value_{peer}(o_i, r_n) = saved_n^{peer} + 0 = p_{o_i}C_rS(r_n, l_{o_i}) \quad (5-7)$$

5.4.2 Value-based D2D Caching Algorithm at the ECs Devices

The value of an object o_{new} , which is requested by a node n for the first time, is initially computed by the CP's server (i.e., following Eqn. 5-5), and sent to the requester node along with

the content. This requires the CP to track and maintain popularity and quality preference distributions for all video content across its subscribers based on their request patterns.

When a node receives the object from the CP's server, it keeps its value unchanged. However, when a node receives an object from a peer, it re-computes the value of the received object using Eqn. 5-7. In other words, the initial value of o_{new} computed by the CP's server may change after the object is cached and shared by the EC's devices. A requester node n caches o_{new} if there is enough empty cache space. Otherwise, it runs the following replacement heuristic.

The node n first finds s lowest valued objects in its cache such that the total size of these objects is equal to or higher than the size of a new object o_{new} . Let us call this set as S_{least_value} . Second, the sum of the values of all objects in S_{least_value} is compared with the value of o_{new} . If the value of o_{new} is larger, then all the objects in S_{least_value} are replaced with o_{new} . Otherwise, o_{new} is dropped. The full logic of caching and replacement is summarized in Algorithm 5-1. All used symbolic notations are summarized in Table 5-1.

```

1: Input: Requested Object  $o_{new}$  Received by node  $n$ 
2: if ( $o_{new}$  is not downloaded from CP) then
3:    $o_{new}.value = value_{peer}(o_{new}, r_n)$ ;
4: end
5: // Initialize the set for objects with the least values
6: Initialize ( $S_{least\_value}$ );
7: while ( $n.rest\_cache$  is not sufficient for  $o_{new}$ )
8:    $o_{min}$  = object with the least value in  $n.cache$ ;
9:    $S_{least\_value}.add(o_{min})$ ;
10:   $n.cache.remove(o_{min})$ ;
11: end
12: // Compare the value of  $o_{new}$  with the total value of objects in
13: // the set of  $S_{least\_value}$ 
14: if ( $o_{new}.value > S_{least\_value}.totalValue$ ) then
15:   store  $o_{new}$  in  $n.cache$ ;
16: else
17:   drop  $o_{new}$  from  $n.cache$ ;

```

```

18:   put objects in  $S_{least\_value}$  back to  $n.cache$ ;
19: end

```

Algorithm 5-1: Value-based caching and replacement policy

Notation	Description
Q	Set of possible video qualities
q_i	i^{th} ranking quality in Q
$P(q_i)$	Popularity of quality q_i
N	Expected number of peers that a node encounters
r_n	Requested quality by node n
p_{o_i}	Popularity of object o_i
l_{o_i}	Play duration of object o_i
$S(q_i, l_{o_i})$	Size of object o_i with play duration l_{o_i} and quality q_i
C_d	Fixed cost for downloading per unit data size of an object (e.g. per MB) from CP's server
C_r	Fixed rebate given to nodes in exchange of sharing per unit data size of an object (e.g. per MB)
$saved_n^{CP}$	Cost saved by a requester node n with caching the object downloaded from CP
$saved_{peers}^{CP}$	Cost saved by peers of the requester node with caching the object downloaded from CP
$value_{CP}(o_i, r_n)$	Value of caching object o_i with quality r_n downloaded from CP at node n
$saved_n^{peer}$	Saved cost by the requester node n with caching the object obtained from a peer
$value_{peer}(o_i, r_n)$	Value of object o_i obtained from a peer by node n
S_v	The summed value of cached objects in the entire network

Table 5-1: List of all notations used in the caching algorithm

5.5 Adaptive-Quality Content Provisioning by Content Provider

This section details an end-to-end video provisioning method that leverages the notion of value and caching mechanism presented in the previous section. When a node n asks for a video segment (i.e., an object) o_i of quality r_n , the CP's server may provide the segment at a quality level that is r_n or higher. An example is shown in Figure 5-3 in which the CP's server may provide the requested objects with another quality (i.e. 4K) that is higher than the actually requested quality (i.e. HD) to the requester node. Even though downloading and caching a higher quality segment

would be more expensive (e.g. more cache space and downloading cost as shown in Figure 5-3.), a requester node n can potentially serve future demands from its peer nodes for the same object, but at a quality higher than that of r_n . In other words, with a higher upfront cost, future potential savings can be accomplished by caching a segment of higher quality than what was initially requested. This trade-off creates the opportunity of provisioning video segments of optimal quality against each individual request and for given network and demand situations.

Adaptive Quality (AQ) Algorithm: We propose the AQ algorithm which works with the value-based caching mechanism presented in Section 5-4. Formally stated, when the CP receives a request from node n for object o_i with quality r_n , it may provide o_i with a different quality q_n to the requester node n , such that $q_n \geq r_n$. The overall provisioning cost can be minimized by maximizing the total value of the objects cached in the ECs' devices.

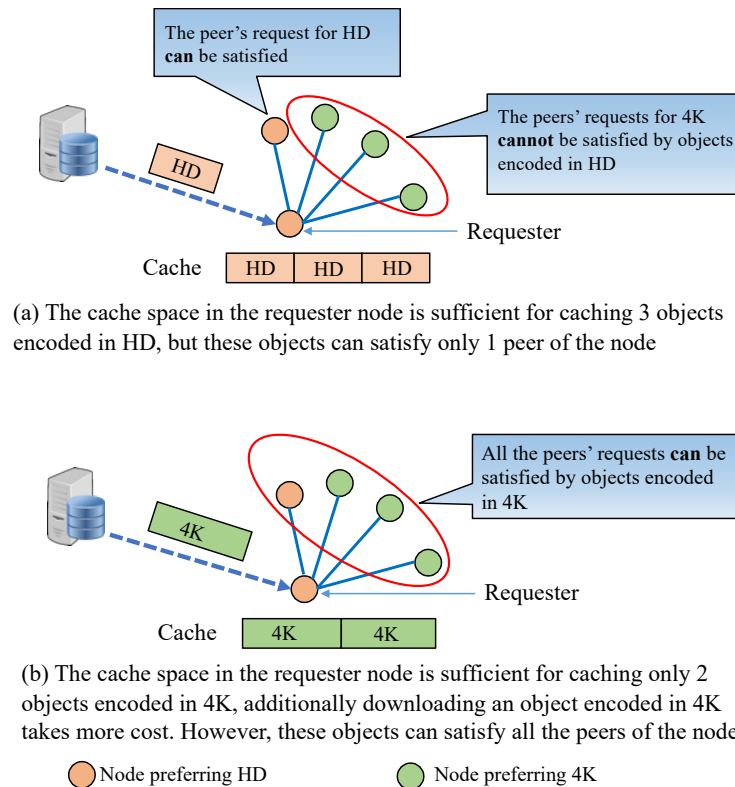


Figure 5-3: Example of how providing higher-than-requested quality content can serve future high-quality user demands

It should be noted that the AQ algorithm runs at the CP's server when it receives an object request. The value-based caching algorithm from Section 5-4, on the other hand, runs at an EC's device after it receives an object from the CP's server or from a peer node. Additionally, the quality that a node plays an object with may be different from the quality that the node actually receives and caches the object with. For example, even though the requester node n receives object o_i with quality q_n which may be higher than its requested quality r_n , the node n still plays o_i with quality r_n , instead of q_n . However, the node n may cache o_i with quality q_n for potentially serving more future demands in the local network.

With AQ provisioning, in steady state, the CP is required to maintain the demand profile for each video segment. Such profile includes the popularity and quality preference distribution for each segment, which is the same as those for its parent video. When the CP's server gets a request directly from an EC, it updates the profile information for the requested segment. Over time, the global demand profiles are tracked and tagged with the content in question by the CP's server. As a result, the popularity and quality preference of an object are known when a node receives the object from the server or from a peer. When a video segment is fetched from a peer node, the recipient is required to send information about the segment retrieval to the CP's server so that the latter can update the demand profile for that segment.

When a node sends request for an object to the CP's server, it includes information about: 1) the requested object quality, 2) size of the free cache space, and 3) local network information in the form of the number of its neighboring peers. For the goal of maximizing the total value of the objects cached in the network, the CP must provide the optimal quality of requested objects. For that, the CP must know the current distribution of the objects cached in the network. However, this introduces scalability issues. In this chapter, a heuristic-based algorithm is proposed for estimating

the sub-optimal quality of objects. This sub-optimal algorithm is based on the following assumptions. First, the total value of the objects cached in the network can be maximized by maximizing the value of objects cached in each node. Second, each node will download and cache the most popular videos with the quality provided by the CP. Based on the assumptions above, the information included in the request to the CP is sufficient for estimating the sub-optimal quality. The CP executes the AQ algorithm as follows:

When the CP receives the request from node n , it computes the total number of videos VN_n that node n is able to cache in its free space E_n using the equation:

$$VN_n = \frac{E_n}{S(q_n, l_{ave})} \quad (5-8)$$

where $S(q_n, l_{ave})$ is the average size needed to store videos with average duration of l_{ave} with quality q_n provided by the CP. The quality q_n is inversely proportional to VN_n . In other words, with fixed E_n and l_{ave} , a larger q_n makes $S(q_n, l_{ave})$ larger, thus resulting in a smaller VN_n , which indicates that lower number of objects can be cached in node n . Moreover, based on Eqn. 5-5, the $total_value_n$ for caching the most popular objects (i.e., segments in the most popular videos $\{v_1..v_{VN_n}\}$) downloaded from the CP in node n can be computed as

$$total_value_n = \sum_{m=1}^{VN_n} \sum_{k=1}^{Seg(v_m)} value_{CP}(o_k, q_n) \quad (5-9)$$

where $Seg(v_m)$ is the number of segments in video v_m . The overall provisioning cost is minimized by maximizing the $total_value_n$ for each node n in the network as follows:

$$\begin{aligned} &max: total_value_n \\ &subjected\ to: q_n \geq r_n, q_n \in Q \end{aligned} \quad (5-10)$$

Note that the quality q_n is directly proportional to $value_{CP}(o_k, q_n)$ (i.e., a larger q_n leads to larger $value_{CP}(o_k, q_n)$), and is inversely proportional to VN_n . As a result, in this optimization problem, the convexity of $total_value_n$ depends on both $value_{CP}(o_k, q_n)$ and VN_n . Specifically, the convexity of $total_value_n$ depends on the function $S(q_n, l_{ave})$ and VN_n shown in Eqn. 5-8,

and the quality preference distribution involved in $value_{CP}(o_k, q_n)$ in Eqn. 5-5. However, since the number of available qualities that can be provided by the CP is usually not large [70], we use the following brute-force approach to solve the equation: 1) the value of $total_value_n$ is computed for each possible quality q_n ($q_n \geq r_n, q_n \in Q$), and then 2) q_n is selected such that the value of $total_value_n$ is maximized. As a result, this q_n is the sub-optimal quality q_n^{sub-op} . After the sub-optimal quality q_n^{sub-op} is computed, the CP sends the requested object o_i with the quality q_n^{sub-op} to the requester node n .

Note that upon receiving a request from a node, the CP computes the quantity q_n^{sub-op} dynamically based on the requesting node's current state including its free cache space and number of neighbors. As a result, q_n^{sub-op} does vary over time.

However, when an object is found and fetched from a peer node, the provider node always sends the object with the requested quality. A higher quality object is not sent in this case because there has already been a higher quality copy cached in somewhere in the network (i.e., at the provider node itself). It is not beneficial to maintain another higher quality replica in the network.

5.6 Performance Evaluation

5.6.1 Experimental Settings

Using the DTN simulator ONE, we evaluate performance of the proposed caching strategies in a 40-node Social Wireless Network with 10 physical waypoints around which the node mobility is simulated. Wireless connections are established between each pair of nodes when they are in the transmission range (i.e. 100m, model after BT and Wi-Fi Direct) of each other. The simulations were carried out under different network topologies and terrain sizes, which determines the spatial

node density for a given number of nodes (i.e., 40). The following caching mechanisms are evaluated.

Parameter	Default Value
Number of Nodes	40
Zipf Parameter α	0.8
Video Population	1000
Average Number of Segments for Each Video	120
Simulation Duration	200000 Requests
Download Cost	10 per MB
Ratio of rebate to download cost β	0.1
Cache Size in Each Node	256 GB
Transmission Range (e.g., Wi-Fi)	100m
Terrain Size (e.g., University Campus)	400m×400m
Length of Each Segment	10s
Maximum Number of Pre-fetching Segments	60

Table 5-2: Baseline parameters used in the simulation experiments

Adaptive Quality (AQ): Each node runs the value-based caching algorithm, and the CP runs the AQ algorithm to provide nodes with a sub-optimal quality for minimizing the overall provisioning cost as presented in Section 5-5.

Requested Quality (RQ): Each node runs the value-based caching algorithm, and the CP provides segments with the same quality as what has been requested by each node. This is a naïve approach that AQ is to be compared with.

Traditional Caching: Algorithms including Least Frequently Used (LFU), Least Recently Used (LRU), and Random Replacement are implemented and compared with AQ and RQ

Unless stated otherwise, all parameters are set to the baseline values as shown in Table 5-2, and videos are assumed to be sequentially played (i.e., $p_{seq} = 1$ in Figure 5-2). The default video quality distribution in the user requests is shown in Figure 5-4. This distribution is modeled after [71], which shows most of the users prefer high quality videos including HD, Full HD (FHD) and 4K.

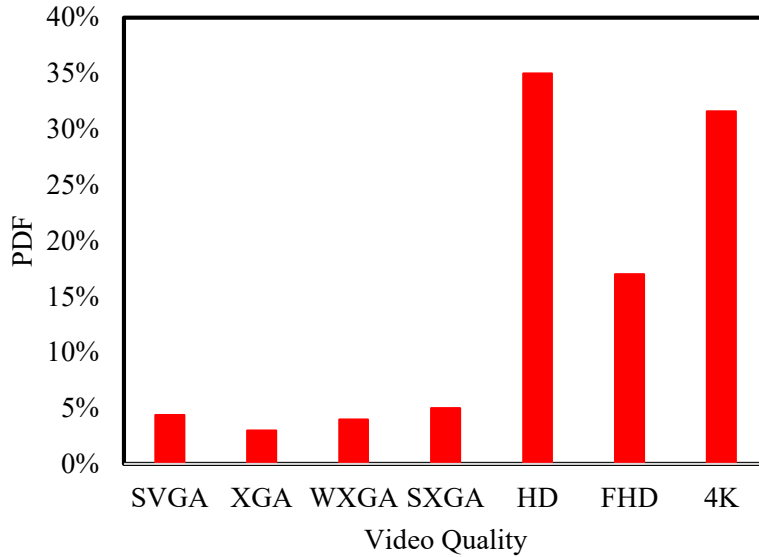


Figure 5-4: Default video quality preference distribution

The popularity of videos is assumed to follow a Zipf distribution with default $\alpha = 0.8$, which indicates the skewness of the distribution. Number of videos is set to *1000*, and each video is divided into *120* segments on average. The length of each segment for each video is set to *10* seconds [72]. The number of pre-fetching segments is set to *60*, which is assumed to be able to fit in the available amount of buffer in each EC's device or node. The simulation is terminated after *200,000* segment requests, after which all performance numbers were observed to have converged in the reported experiments.

5.6.2 Impacts of Cache Availability and Terrain Size on Cost Saving

As shown in Figure 5-5, provisioning costs corresponding to all the algorithms expectedly decrease with an increase in available cache size. Also, provisioning costs of all the algorithms are similar when the available cache size is extremely small or extremely large. For a very small cache size, only a few objects can be cached in the network. Thus, the benefits of caching are low across

the board. In case of a very large cache, most of the popular objects can be cached in the network irrespective of the applied caching algorithm.

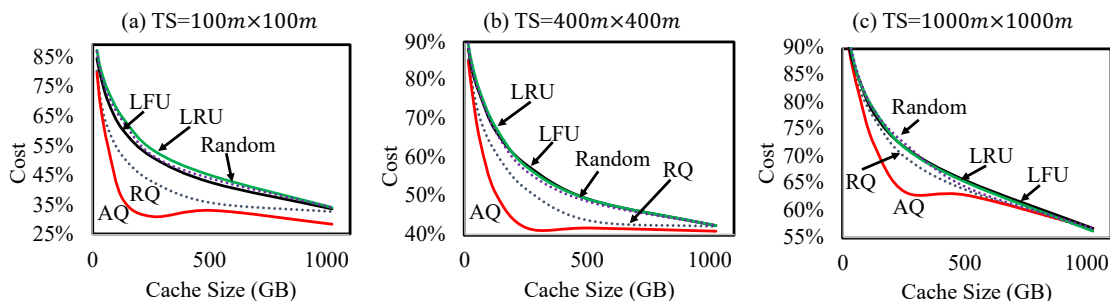


Figure 5-5: Provisioning cost with different terrain sizes and per-node cache availability

Moreover, for a given node-count (i.e., 40) different terrain sizes (TSs) make different Device-to-Device connection densities which are represented by the average number of neighbors encountered per node as shown in Table 5-3. A smaller terrain size makes higher connection density and produces generally lower provisioning costs. This is because the nodes in this case are able to visit more peer ECs during the TAD for obtaining requested objects from those peers.

Terrain Size	Average No. of Neighbors Encountered Per Node
100m × 100m	38.9
400m × 400m	14.8
1000m × 1000m	4.1

Table 5-3: Average no. of neighbor ECs encountered per node for different terrain sizes (number of nodes is set to 40)

Another observation in Figure 5-5 is that the provisioning cost of the Adaptive Quality (AQ) algorithm is the lowest across all available cache and terrain sizes that have been experimented with. This is because, with AQ, nodes may cache objects with higher quality than they actually request. Such objects can satisfy requests coming from other nodes, which may need the object with a quality equal or lower than that of the available one. Provisioning cost for the Requested Quality (RQ) policy is generally between AQ and the traditional caching algorithms. Since the value-based caching's criteria in RQ involves both object popularity and quality demand profiles, compared

with traditional caching algorithms, RQ can better indicate which objects should be cached to maintain a lower provisioning cost. However, RQ fails to reduce the provisioning cost further because it only allows caching video segments with quality equal to the requested quality. This explains why its performance is lower than that of AQ.

Figure 5-6 depicts the local and remote hit rates for the algorithms in different network terrain sizes. For a fixed terrain size, a larger cache size results in higher local and remote hit rates. A smaller terrain size causes a higher remote hit rate because of higher node density. The impacts of terrain size on local hit rates is not that significant. Observe that the remote hit rate of the AQ policy is the highest among all the caching algorithms among various cache and terrain sizes. That is because, with AQ, a node can share more objects with other nodes by caching higher quality video segments than what is needed for itself. Similarly, the remote hit rate of RQ is higher than that for the traditional caching algorithms because the value-based caching applied in RQ is more effective for indicating the real user demands.

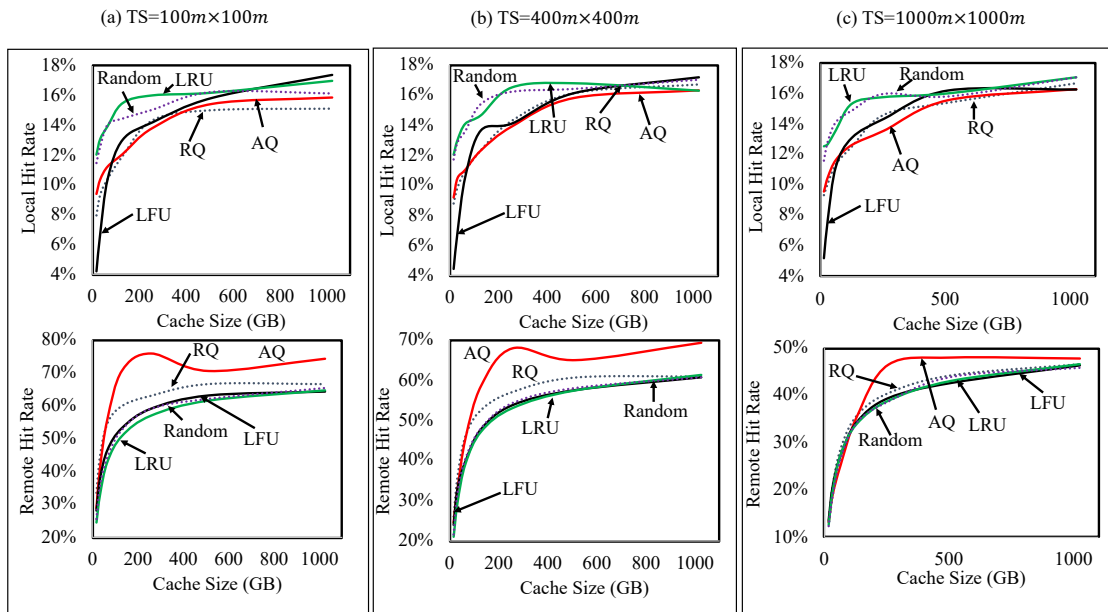


Figure 5-6: Local and remote hit rates for different network terrain sizes

Figure 5-7 shows the total value of cached objects S_v in the entire network. It is defined as:

$$S_v = \sum_{n \in \mathbb{N}} \sum_{o_i \in \mathbb{O}_n} \text{value}_{CP}(o_i, Q(o_i)) \quad (5-11)$$

while \mathbb{N} is the set of all the nodes in the network, and \mathbb{O}_n is the set of all the objects cached at node n . Additionally, $Q(o_i)$ indicates the quality of object o_i . Note that S_v is computed when the network reaches a steady state. As expected, the S_v of each caching mechanism increases with an increase in available cache size for all different terrain sizes. This is because each node is able to cache a greater number of objects with a larger cache size. Moreover, the S_v of AQ is the largest across all the mechanisms because the value of objects cached at each node in AQ is maximized by Eqn. 5-10. As a result, the total value of cached objects at all the nodes (i.e. S_v) is also the largest across all the algorithms. On the other hand, the S_v of RQ is larger than all the other caching algorithms except AQ. Since the value-based cache replacement policy (see Algorithm 5-1) is applied in RQ, the object with a larger value is never replaced. Another observation is that S_v of each mechanism decreases with a larger terrain size. This is because the N in Eqn. 5-4, which represents the expected number of peers a node encounters, decreases when the terrain size is larger (see Table 5-3), thus S_v is smaller according to Eqn. 5-4, 5-5 and 5-11.

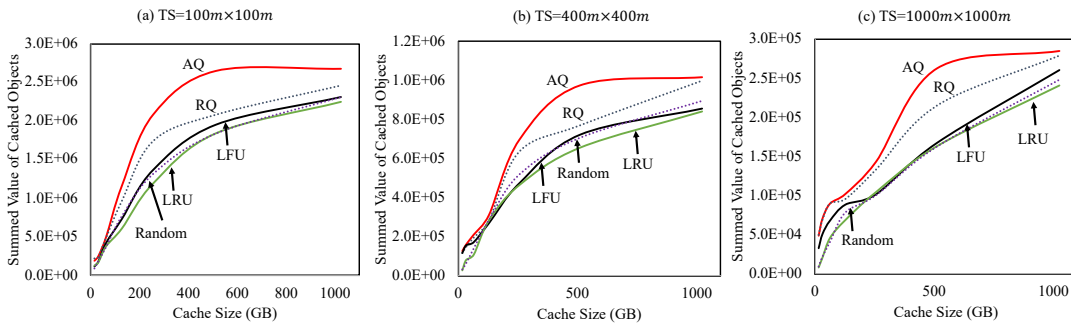


Figure 5-7: Total value of cached objects in the entire network

The bandwidth overhead of caching is defined as the bandwidth usage of CSP's cellular links. Such overheads of various caching algorithms in different terrain sizes are shown in Figure 5-8. It can be observed that a larger terrain size generally makes a higher bandwidth overhead across all

the algorithms. This is because, in a larger terrain size, each node has a lower remote hit rate (see Figure 5-6). As a result, a node has to download more video segments using the CSP's network. However, the bandwidth overhead of AQ is always the lowest for all the terrain sizes, because in AQ the nodes enjoy the highest remote hit rate by caching video segments with higher quality than their actual demands. On the other hand, the bandwidth overhead of RQ is lower than all the traditional caching algorithms. This is also because nodes with RQ have higher remote hit rates than with the traditional algorithms.

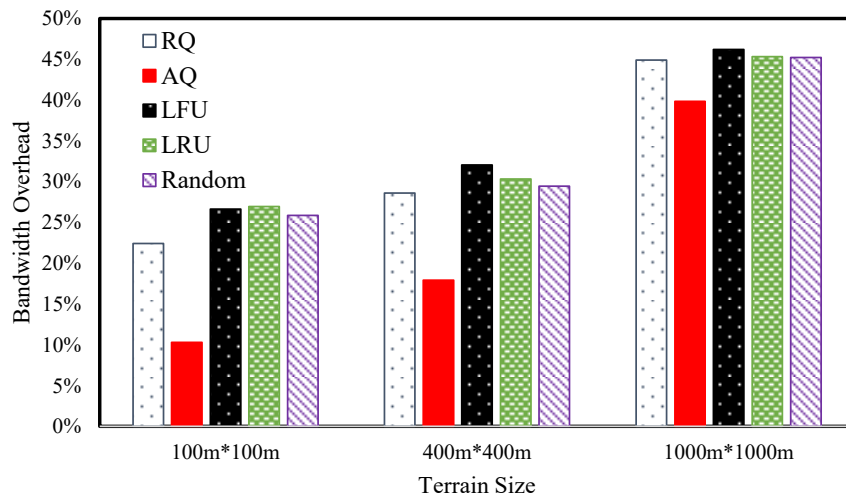


Figure 5-8: Bandwidth overhead for different network terrain sizes

5.6.3 Evolution of Provisioning Cost and Hit Rates over Time

Figure 5-9 explains how the provisioning cost and hit rates builds up in the network over a period of 60 hours. For both AQ and RQ policies, the provisioning costs are 100% at the beginning. However, as shown in Figure 5-9 (a), the costs significantly reduce over time as more and more video segments are cached in the D2D network of the ECs. This is also reflected in Figure 5-9 (b) and (c), where the local and remote hit rates for both policies increase over time. Over time, the cost of AQ reduces faster than RQ, while the remote hit rate of AQ increases more than that of RQ. This is because with AQ, the nodes cache video segments with higher quality, which results in a

higher remote hit rate and resultingly lower cost. However, there is no significant difference between the local hit rates of AQ and RQ as shown in Figure 5-9 (b).

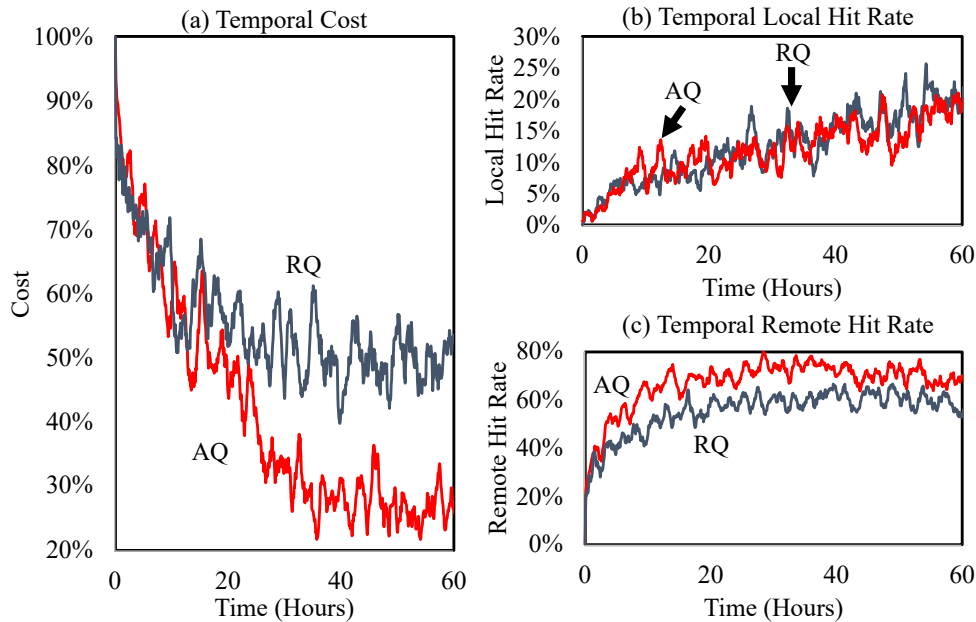


Figure 5-9: Provisioning cost, and local and remote hit rates over time

5.6.4 Content Quality Density of the Cached Objects

Quality density is defined as the probability density function of video segments with specific qualities that are cached within the network. Figure 5-10 depicts the density of video segments of all seven available qualities (see Figure 5-4) for the policies AQ and RQ. Such density is a function of quality preference distribution in user demands and the corresponding sizes of objects for each quality. For example, density of SVGA (i.e. the lowest quality in our experiments) in both RQ and AQ is the highest. This is because an object's size with this quality is extremely small. As a result, the nodes are able to cache many such segments. Specifically, in RQ, the densities of HD, FHD and 4K video segments are the second highest since they are the most popular qualities, following the default preference distribution as shown in Figure 5-4. Densities of these popular qualities in AQ, however, are extremely low except for the 4K. That is because the video source encoded in 4K is

able to satisfy the requests for HD and FHD. That reduces the provisioning cost with AQ by increasing the ratio of content sharing (i.e., the remote hit rate) in the D2D network of ECs.

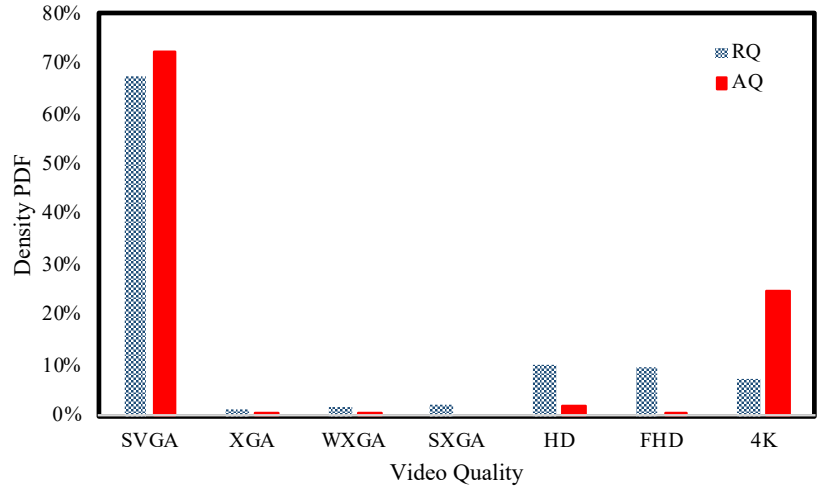


Figure 5-10: Cached video quality density for AQ and RQ

5.6.5 Content Delivery Latency

Content Delivery Latency (CDL) is the duration between when a node generates a request for a specific segment and when it actually obtains the segment. In Section 5.3.2, it was stated that during the Tolerable Access Delay (TAD), which is computed using Eqn. 5-1, a node first searches a requested segment in its local cache and then in its neighboring peers. It follows that the CDL for a video segment is acceptable only if it is less than the TAD for that segment. For example, when a user starts to play a video, if the length of each segment is $10s$, then according to Eqn. 5-1, the TAD of the $30th$ segment is $290s$. In this case, if the CDL of the $30th$ segment is less than $290s$, the user can experience a smooth play.

Figure 5-11 depicts the average CDL of AQ and RQ over various cache sizes. The CDLs of both AQ and RQ decrease when the cache size increases. This is because, with a larger cache size, more requested segments can be found in the local caches. For such cached segments, the CDL for obtaining them is zero. Also, it should be observed that the CDL for AQ is lower than that of

RQ. This is because the remote hit rate of AQ is larger than that for RQ (see Figure 5-6), which means more requests in AQ can be served by the neighboring peer nodes before the TAD expires.

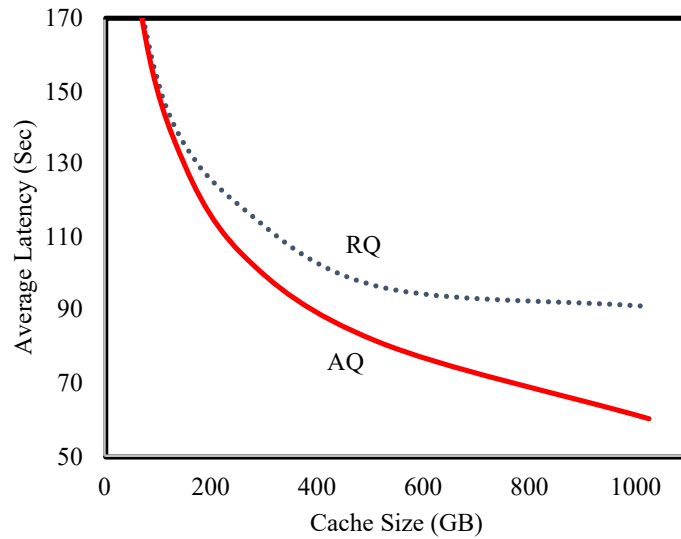


Figure 5-11: Content Delivery Latency (CDL) for the AQ and RQ policies

5.6.6 Impacts of Video Play Sequence

As shown in the video play model in Figure 5-2, besides sequentially playing a video, a user may fast-forward to a later segment or rewind to a previous segment of the current video. Moreover, the user can stop playing the current video and switch to a different one. Figure 5-12 depicts how the probabilities of these actions affect the provisioning costs of the proposed AQ and RQ caching mechanisms.

Generally, higher p_{ff} and p_{swi} (i.e., frequent fast forward and switching to a different video) cause fewer segments from the currently-playing video to be requested. For example, if a user wants to fast-forward to segment s_{15} after he/she finishes the first segment s_0 , then requests for pre-fetching the following segments after s_0 (i.e., s_1 to s_{14}) are deleted. Similar effects can be observed in case of switching to a different video. With frequent switching, fewer segments of each video need to be cached. In other words, with higher p_{ff} and p_{swi} , segments from more diverse videos

can be cached. This increases the local and remote hit rates while reducing the overall provisioning costs.

Frequent rewinds (i.e., high p_{rew}), on the other hand, can result in a high local hit rate due to repeated requests of locally cached video segments. This explains why the provisioning cost reduces with increasing occurrences of the rewinds. The provisioning cost of the AQ policy is consistently lower than RQ with various p_{ff} and p_{swi} values in Figure 5-12 (a) and (b) respectively. The situation is reverse for rewinding in Figure 5-12 (c). This is because higher p_{rew} causes more local caching. Since AQ tends to cache segments with higher qualities than RQ, fewer video segments can be cached for a given local cache size. As a result, in this rare case, RQ performs slightly better than AQ when the probability of rewinding is high. However, the local hit rate is not significantly higher in RQ compared to AQ when the probability p_{rew} is low; that is the reason why the provisioning cost of AQ is initially lower than RQ with low p_{rew} .

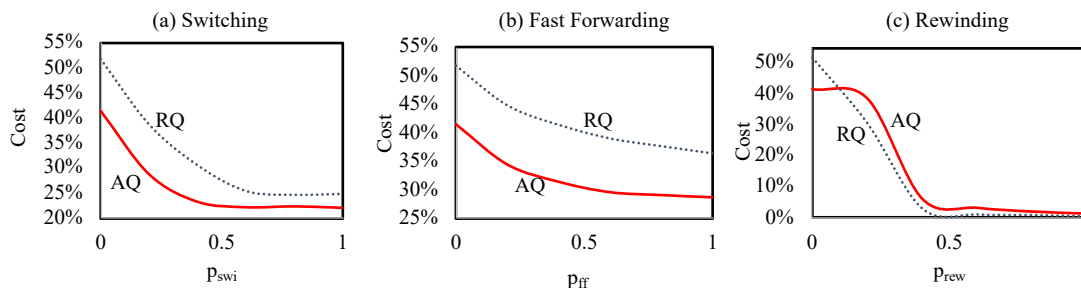


Figure 5-12: Impacts of different video play sequences on provisioning cost

5.7 Comparison with Reactive User Preference Profile Algorithm

Reactive User Preference Profile (R-UPP) [26] is a reactive caching algorithm that serves similar applications as done by the proposed mechanisms in this work. R-UPP, in its original form, is an infrastructure-based caching policy that caches popular content in infrastructures such as the edge servers. However, the mechanism can be adapted to a D2D mode in which caching is done within the ECs' devices as opposed to in any infrastructure. In R-UPP, the replacement policy is

based only on the popularity of objects without considering any quality preference distributions, as done in AQ and RQ. We implement a D2D version of R-UPP in the simulator ONE and compare its performance with AQ and RQ in a mobility scenario extracted from the traces of taxis in San Francisco [73].

This mobility model contains traces of 40 taxis over 70 hours in the city of San Francisco. The taxis communicate with each other using Dedicated Short-Range Communication (DSRC) [7] links with transmission range set to 1km, and the average number of neighbors per node (i.e. taxi) is around 20.7. The other parameters follow the baseline as shown in Table 5-2.

5.7.1 Impacts of Cache Size

In Figure 5-13 (a), provisioning costs corresponding to all three algorithms expectedly decrease with the increase in available cache size. The provisioning cost for the AQ strategy is the lowest and R-UPP is the highest across all cache sizes. This is because the cache replacement policy of R-UPP relies only on the popularity of objects without considering the quality preference, which is considered by both AQ and RQ. For example, in R-UPP, an object with low popularity but high preferred quality may be replaced with another object, whose popularity is slightly higher but its quality preference is much lower. An object with a more frequently-preferred quality may satisfy more number of future demands for the same content in a way that the total obtained value would be higher.

The local and remote hit rates for the three caching algorithms are depicted in Figure 5-13 (b) and (c). It can be seen that the local hit rates of all three policies are similar across all the cache sizes. The remote hit rate of R-UPP is the lowest, since with R-UPP a node may keep an object with high popularity and low preferred quality. This significantly reduces the probability that other nodes can obtain the requested content with such high quality from their peer ECs' devices.

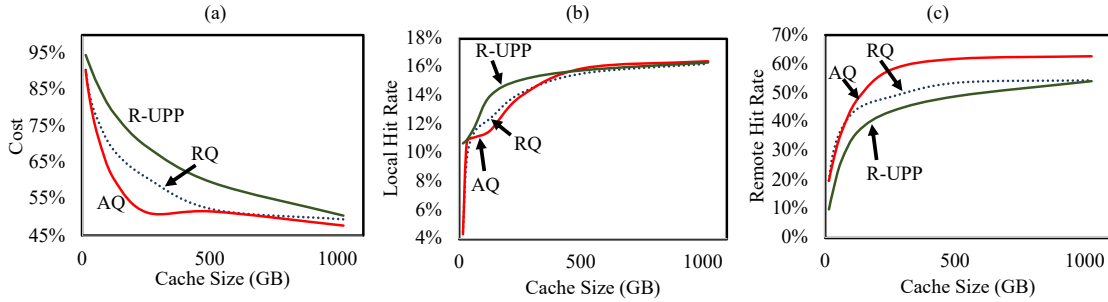


Figure 5-13: Provisioning cost and hit rates of R-UPP, RQ, and AQ

5.7.2 Impacts of Rebate-to-Download-Cost Ratio

Figure 5-14 (a) depicts the provisioning cost of R-UPP in comparison with AQ and RQ for different values of Rebate-to-Download-Cost Ratio β (see the pricing model in Section 5.2). For all three algorithms, the costs increase with higher β . This is because a higher β indicates a larger rebate C_r , which is the cost of obtaining a video segment from a peer device. Additionally, provisioning cost of the AQ is the lowest across all the β values, while R-UPP is the highest. This is because the cache replacement policy of R-UPP relies only on the popularity of objects and not the preference distribution of quality. According to the results in Figure 5-14 (b), the impacts of β on local hit rates are insignificant for all three mechanisms.

However, as shown in Figure 5-14 (c), the remote hit rates of AQ and RQ decrease with increasing β . This is because with higher β , C_r gets closer to the cost C_d which is incurred by downloading objects directly from the CP's server. According to Eqn. 5-5 and 5-7, if C_r is closer to C_d , the value of caching an object received from a peer device becomes similar to the value of caching an object received from the CP. Therefore, when the cache in a node is full, a segment that was originally obtained from a peer may not be replaced by segments downloaded from the CP's server. As a result, most of the objects cached in the network are from the peers in the network. In other words, all the nodes in the network finally cache the same objects, which reduces the remote

hit rates. On the contrary, the remote hit rate of R-UPP is not notably affected by β since its replacement policy does not involve any rebates.

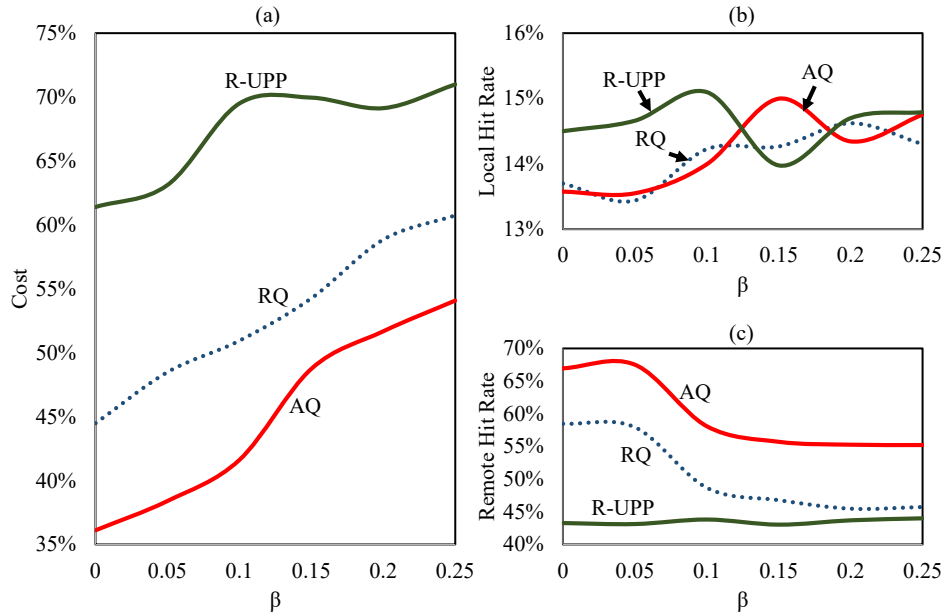


Figure 5-14: Provisioning costs for varying rebate-to-download-cost ratio

5.7.3 Impacts of Different Quality Preference Distributions

All presented results so far use the default video quality preference distribution as shown in Figure 5-4. In this section, we analyze the impacts of other possible quality preference distributions. Figure 5-15 depicts the provisioning costs for: 1) uniform (Figure 5-15 (a)), Gaussian-Like (GL) (Figure 5-15 (b)), and Default-Inverse (DI) (Figure 5-15 (c)) distributions. DI is the inverse of the default preference distribution (Figure 5-4) in which the low qualities are more popular. Compared to the performance with GL and DI, provisioning costs of all three algorithms are slightly higher for the uniform preference case. This is because the probability of requesting each video segment quality is equal in the uniform distribution, thus there is less benefit of caching objects with any specific qualities. The provisioning costs for all three algorithms are more similar for GL and DI cases compared to the uniform case. This is because, in GL and DI, the high-quality videos are not

notably popular. That is, the most popular quality is SXGA in GL, and the low qualities are more popular in DI. Thus, caching higher quality than the user’s preference is not advantageous. However, as expected, the provisioning cost of AQ is generally the lowest, while the provisioning cost of R-UPP is the highest under all three quality preference distributions.

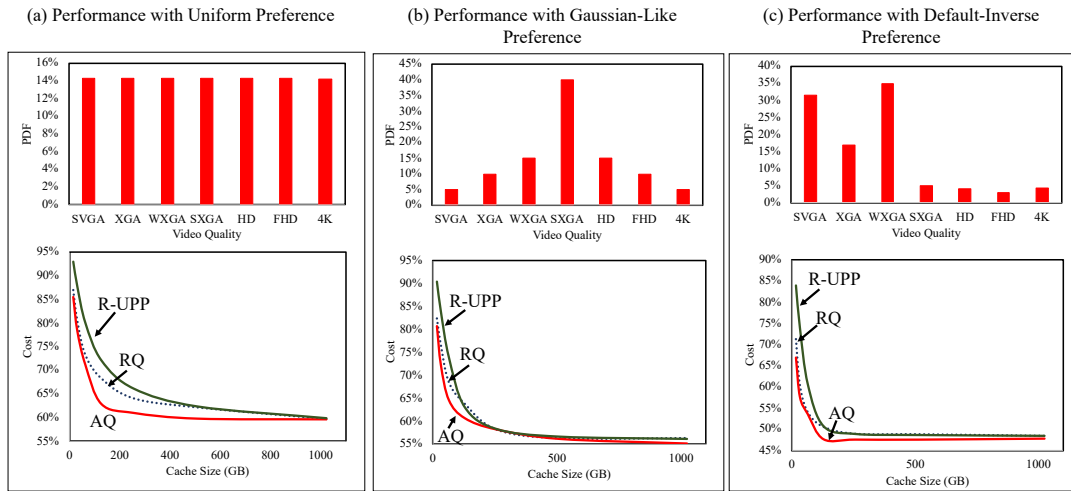


Figure 5-15: Provisioning cost with different distributions of video quality preferences

5.8 Summary

A value-based D2D collaborative caching strategy for hierarchically coded streaming video segments is proposed in this chapter. Based on video popularity and heterogeneous user demands, an adaptive quality provisioning algorithm, which works with a value-based caching, is developed for minimizing the overall video provisioning cost for a provider. The key idea is to decide the optimal quality of video segments to be cached in users’ devices so that the predicted future demands from users and their peers are satisfied, thus minimizing content provisioning costs. Simulation experiments were carried out in mobile wireless network scenarios, and the results indicate that the proposed mechanism is able to reduce the overall provisioning cost compared with traditional caching algorithms and a comparable caching algorithm, namely, R-UPP. The cost-

saving of the proposed mechanism is particularly visible when the connection density of nodes is high, and when the popularity of higher quality video content is high.

Chapter 6: Caching for Dynamic Map Dissemination in Vehicular Networks

6.1 Introduction

With increasing vehicle connectivity through Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), and cellular networks, many data-driven in-vehicle applications are emerging in recent years. Many such applications involve space- and time-varying information such as dynamic electronic maps used in vehicle navigation. Example scenarios that can create dynamic map include work zone related traffic diversion lasting for hours to days, temporary traffic lights during an event such as football game, and accident related traffic diversions. A map with such dynamic information is termed as a dynamic map [74].

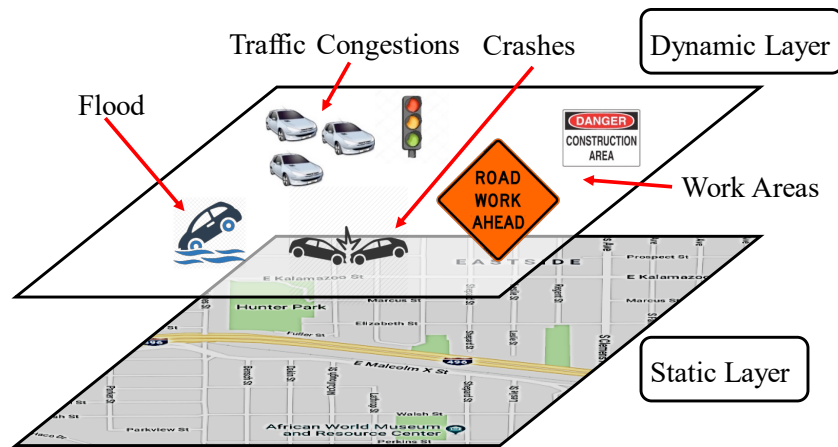


Figure 6-1: Layered abstraction of dynamic maps

Figure 6-1 depicts an example dynamic map, which includes two layers. The bottom one is static with relatively permanent components such as roads, building, open areas etc. The top layer includes dynamic data such as constructions, traffic congestions, crashes, events etc. In most cases, the vehicles are pre-loaded with the static component. The time-varying dynamic component is downloaded to the vehicles on an as-needed basis over the V2V, V2I, and cellular connections.

The current best practice is to push the dynamic map data from a cloud-based map application server (e.g., Google Map server) to the vehicles via cellular networks. However, this approach involves cellular bandwidth usage cost that has to be paid by the users to their mobile network operators (e.g. AT&T, Verizon, etc.). The objective of this chapter is to reduce cellular bandwidth usage for disseminating dynamic map data.

One plausible approach for such cost reduction would be to always download and cache dynamic map data from map application servers (MASs) to edge devices such as roadside units (RSUs) through fiber links, and relay that to the nearby vehicles via Dedicated Short-Range Communications (DSRC) links. In scenarios with very high RSU densities, this approach will work. In reality, however, the RSU deployments are expensive, and therefore, the RSU coverage is expected to be somewhat sparse in many settings. An alternative is to use the vehicles themselves as a distributed storage system in which dynamic maps can be intelligently cached based on the spatial and temporal localities of their needs. This chapter presents such a collaborative caching solution towards the goal of minimizing cellular bandwidth usage by leveraging inter-vehicle map sharing.

6.2 System Architecture

6.2.1 Network Model

All network and system components are shown in Figure 6-2. Each vehicle in the transportation network is able to access the MAS through paid cellular links (e.g. 4G or 5G, etc.). Vehicles can also connect to each other or to the RSUs via DSRC-based V2V and V2I links. These links are considered spatially limited and free of cost. Finally, the edge devices such as the RSUs are able to connect to the map server via high-speed fiber links such as Gigabit Ethernets.

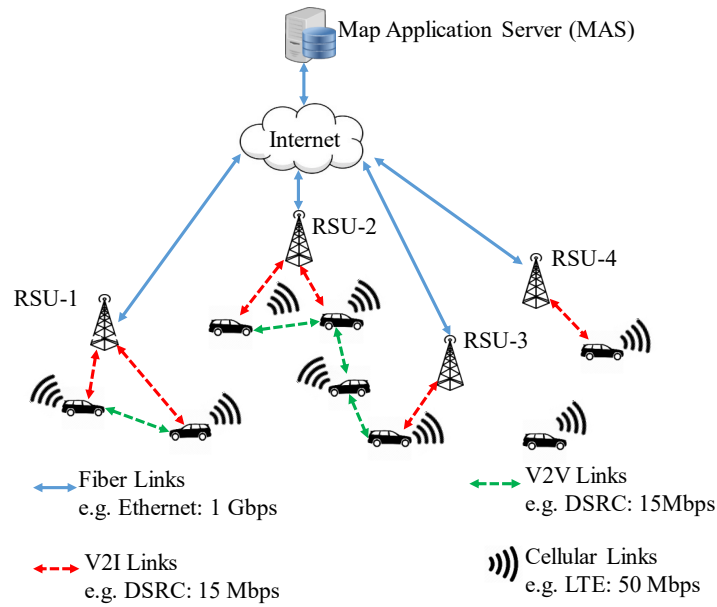


Figure 6-2: Generalized model with different networking components

6.2.2 Dynamic Map Data

Dynamic map data is always generated by the edge devices such as the RSUs. Generation of such data can be triggered by events such as accidents, traffic congestion, and road work. Such information can be input manually at the RSU or through the cloud. The edge unit might also be equipped with sensors that continually survey and identify events that need to be distributed. A dynamic map data module would contain two key information components, namely, its expiry time and the geographical scope of dissemination. The geographical scope indicates the region in which the data needs to be disseminated. The expiry time indicates till what point in time the dynamic map data needs to be disseminated among the vehicles within the specified geographical scope. In practice, the expiry time of a dynamic map can be determined based on the expected end-time of the corresponding event such as road work or a football game. Since the end time of an event may change (e.g., a crash is cleared earlier than expected or a road work is extended by few additional hours), the generation model accommodates a way to update the expiry time of a content.

The process of dynamic map generation and updates from an RSU is as follows. Distinct dynamic maps from an RSU are generated with a rate of λ_g maps per unit time. Once generated, updates for that map are generated with a rate of λ_u updates per unit time. An update can change the geographical scope and/or the time of expiry of the corresponding dynamic map item. Both map generation and update generation processes can be modeled with an appropriate probability distribution. Considering memory-less properties, for all our analysis in the following sections, exponential distribution is used for the process of dynamic map generation. However, to simplify the problem, in this chapter the update rate λ_u is set to 0.

6.3 Dissemination and Caching Mechanisms

Dynamic map dissemination constitutes three key mechanisms, namely, push, pull, and caching.

6.3.1 Dynamic Map Data Push

As shown in Figure 6-3, when an RSU generates a dynamic map data, it is referred to as the Originating RSU (O-RSU) with respect to that dynamic map data. Upon generated, the O-RSU pushes the data to the nearby vehicles (i.e., vehicles within the V2I transmission range of the O-RSU), and also uploads it to the MAS via terrestrial wired links.

Depending on the geographical scope of a specific dynamic map data module, the MAS can push it to distant vehicles through the RSUs that are local to those vehicles. These RSUs are referred to as the Gateway RSUs or G-RSUs with respect to that dynamic map data. Upon receiving the dynamic data from the MAS, the G-RSUs push it to the vehicles around them via V2I links.

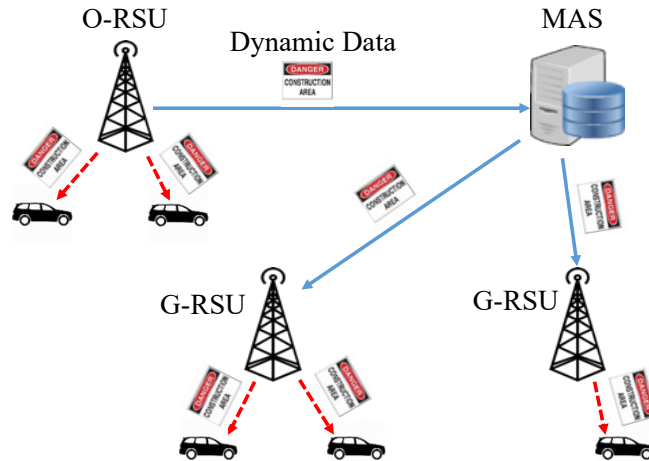


Figure 6-3: Push mechanism of dynamic data dissemination

It should be noted that any RSU can serve the role of either an O-RSU or a G-RSU depending on where a dynamic map data has originated. With respect to the data that was originated locally, the RSU is an O-RSU, and with respect to the data that was originated remotely, the RSU is a G-RSU.

6.3.2 Dynamic Map Data Pull and Caching

The push mechanism described above can deliver dynamic map data to all the vehicles that are within the V2I transmission ranges of the O-RSU and all the G-RSUs corresponding to the data item. For all other vehicles, a pull mechanism is needed. Consider an example scenario in which a vehicle is driving towards a geographical area and it requires any dynamic map data available for that area before it reaches the area so that an efficient route can be computed. Also, the vehicle is out of range of the O-RSU and G-RSUs that are corresponding to the destination area of interest. In such a situation, the vehicle would have to make an active request for that piece of dynamic data. In order to make such pull operations effective, the data item of interest needs to be cached within the vehicular network so that a vehicle can possibly retrieve it from another vehicle which may have cached it upon an earlier request.

The key architectural concept is to develop such an in-vehicle and in-RSU content caching mechanism that can be leveraged to reduce the usage cost of the on-vehicle cellular link.

6.3.2.1 Demand Model

Let $p(V_i, D_{R_j})$ be the probability that a vehicle V_i may request the dynamic data D_{R_j} that is of a specific geographic area and generated by its O-RSU R_j in that area. It is assumed that $p(V_i, D_{R_j})$ is inversely proportional to the current distance between the vehicle V_i and the RSU R_j as follows:

$$p(V_i, D_{R_j}) = \frac{C}{\text{distance}(V_i, R_j)} \quad (6-1)$$

where C is a constant. The rationale behind such inverse relationship is that if the vehicle V_i is currently far away from the RSU R_j , then V_i is commensurately less likely to request the dynamic map data item D_{R_j} . It should be noted that the geographical location information is extracted from the static layer of the integrated map information as shown in Figure 6-1. The static layer of the map is assumed to be pre-loaded in the vehicle as a part of the navigational map system such as google map etc.

6.3.2.2 Data Search Model

Upon originating a request, a vehicle first performs a local search for the requested dynamic data in its own cache. If that fails, the vehicle performs a remote search among the nearby vehicles and RSUs via V2V and V2I links. If the data component is not found after those searches within a pre-defined Tolerable Access Delay (TAD), the vehicle sends a request to the MAS and pulls the requested data through the cellular link. TAD represents the duration that a vehicle is willing to wait before the request is successfully served. TAD for a dynamic map data item is set at the application level when the request for it is originated at the application such as a vehicle's

navigation software App. It is assumed that the TAD for a request (V_i, D_{R_j}) would depend on the distance between the requested vehicle V_i and the O-RSU of the data item D_{R_j} . Formally stated:

$$TAD(V_i, D_{R_j}) = r \times distance(V_i, R_j) \quad (6-2)$$

where r is a constant. The physical meaning is that the TAD for a request will be larger if the requesting vehicle is far away from the geographical origin of the dynamic map data component. In other words, if the vehicle is away from the geographical area, it can possibly wait for some time (i.e., TAD) before the dynamic map component is fetched from other vehicles and RSUs. As a result, the chances of having to fetch the data from the map server using cellular link and its corresponding expenditure can be reduced.

6.3.2.3 Cache Replacement Policy

A key architectural component that determines the effectiveness of caching is how a dynamic map data is replaced from a vehicle's cache after its usage is over. When a vehicle V_i obtains a new dynamic map data D_{R_j} , it may cache the data locally if there is sufficient empty cache space. If space is not available, the following policy is executed to replace an existing map item by this newly acquired one.

We introduce a notion of caching value for each dynamic data, which is defined as how much cellular bandwidth usage can be avoided by caching the data item within in-vehicle cache. The caching value of data item D_{R_j} within vehicle V_i is defined as follows:

$$value(V_i, D_{R_j}) = p(V_i, D_{R_j}) life(D_{R_j}) size(D_{R_j}) \quad (6-3)$$

where $life(D_{R_j})$ is the remaining life of D_{R_j} defined as the duration between current time and its expiry time. The quantity $size(D_{R_j})$ represents the size of D_{R_j} . The expected savings of cellular bandwidth usage cost (i.e., the cache value) associated with map item D_{R_j} in vehicle V_i 's cache is

higher for D_{R_j} with: larger size, longer expiry time, and higher request probability from V_i . The notion of value is used as follows.

```

1:  Input: received dynamic data  $D_{R_j}$  by vehicle  $V_i$ 
2:  if ( $V_i.remaining\_cache\_size \geq size(D_{R_j})$ ) then
3:     $V_i.cache(D_{R_j});$ 
4:  else
5:    for each data  $D_{R_k}$  cached in  $V_i$ 
6:       $D_{R_k}.value = value(V_i, D_{R_k});$ 
7:    end
8:     $initialize(S_{least\_value});$ 
9:    while ( $V_i.remaining\_cache\_size < size(D_{R_j})$ )
10:      $D_{min} =$  data with the least value in  $V_i$ ;
11:      $S_{least\_value}.add(D_{min});$ 
12:      $V_i.remove(D_{min});$ 
13:   end
14:   if ( $D_{R_j}.value > S_{least\_value}.totalValue$ ) then
15:      $V_i.cache(D_{R_j});$ 
16:   else
17:      $drop(D_{R_j});$ 
18:     put data in  $S_{least\_value}$  back to cache of  $V_i$ ;
19:   end
20: end

```

Algorithm 6-1: In-vehicle and in-G-RSU caching/replacement policy

When a vehicle V_i fetches a new dynamic map data D_{R_j} , it first computes the value of caching for all the existing items in its cache. It then identifies the s lowest valued items in the cache such that the total size of those items is equal to or higher than the size of the new data D_{R_j} . The set containing those cached items is referred to as S_{least_value} . At this stage, the sum of the values of all the data items in S_{least_value} is compared with the value of D_{R_j} . If the value of D_{R_j} is larger, then all the data in S_{least_value} are replaced with D_{R_j} . The rationale behind this action is as follows. Since the value of the new data is larger, by caching it, the expected savings in the cellular link usage cost from the vehicle will be also larger. By the same token, when the value of D_{R_j} is comparatively

lower, the existing cache items have higher potential to save cellular link usage cost than the new dynamic map item D_{R_j} . Therefore, instead of replacing any existing cache items, it should be dropped. The full logic of the caching and replacement is summarized in Algorithm 6-1.

Dynamic map data items can also be cached in the G-RSUs. When a G-RSU receives an item from the MAS, it may opt to cache the item following a very similar caching and replacement logic. The difference here is the way the demand (i.e., Eqn. 6-1) is computed. For computing the value of dynamic map data D_{R_j} cached on a G-RSU R_i , the probability part $p(V_i, D_{R_j})$ in Eqn. 6-1 needs to be changed to $p(R_i, D_{R_j})$, which represents the probability that D_{R_j} is requested by the vehicles in the vicinity of RSU R_i .

6.4 Performance Evaluation

We evaluated the performance of the proposed caching mechanism using the Delay-Tolerant Networking (DTN) simulator ONE. A 50-vehicle network with mobility traces from taxis in San Francisco has been used. As for the RSUs, they are uniformly randomly placed within the geographical scope of the city. The overall architecture of the network is the same as what is presented in Section 6.2.1. Unless stated otherwise, all the parameters are set to the baseline values as shown in Table 6-1.

Parameter	Default Value
No. of RSUs in the network	10
No. of G-RSUs for each dynamic data	9
No. of vehicles	50
λ_g for dynamic data generation	3e-4/s
λ_u for dynamic data update	0 updates/s
Life of dynamic data	6 hours
Average size of dynamic data	10MB
Cache size on RSU	80MB
Cache size on vehicle	80MB
Transmission range of V2V and V2I (e.g. DSRC)	1000 meters
Data transfer rate of V2V and V2I	15Mbps
Data transfer rate of cellular (e.g. LTE)	50Mbps

Data transfer rate of fiber links (e.g. Ethernet)	1Gbps
Ratio r for TADs	0.005s/m
Simulation duration	100 hours

Table 6-1: Baseline parameters used in the simulations

Each RSU generates dynamic data components once in every 1 hour (i.e. $\lambda_g = 3 \times 10^{-4}/s$) with no updates by default (i.e., an update rate of 0 updates/s). The RSU pushes the generated data to all the other G-RSUs (i.e. 9 G-RSUs by default) using the push mechanism described in Section 6.3.1. Upon generation, the life span of each dynamic data is set to 6 hours. The size of dynamic data is chosen randomly between $0.5MB$ to $19.5MB$, thus the average size is $10MB$. In the simulation, it is assumed that the V2V and V2I links are DSRC, and the cellular network is LTE. Moreover, the fiber links between the MAS and RSUs are Ethernet. The constant r for computing TADs in Eqn. 6-2 is $0.005s/m$. All the presented results correspond to a simulation duration of 100 hours.

6.4.1 Mobility Characteristics

Before we delve into performance characterization of the proposed caching architecture, it is useful to understand the nature of vehicle mobility in our experimental setting. Figure 6-4 (a) shows the number of vehicles encountered per hour averaged across all vehicles. Figure 6-4 (b) depicts the same metric as perceived by the RSUs. First notable observation is that on an average, a vehicle always encounters many more other vehicles compared to the RSUs. Low encounter density of the RSUs are explained by the fact that the RSUs are stationary while the vehicles are moving. Also, the number of RSUs is smaller than the number of vehicles in the system. The higher encounter rates for the vehicles in Figure 6-4 indicate that in-vehicle caching is expected to be much more effective for finding map components in the caches compared to in-RSU caching.

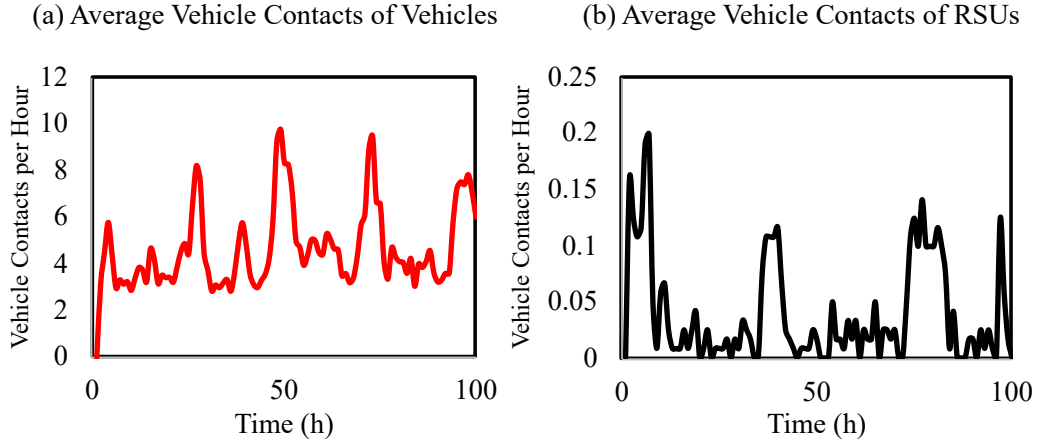


Figure 6-4: Mean vehicle encounter rates for the vehicles and RSUs

6.4.2 Impacts of In-vehicle Caching on Bandwidth Usage (BU)

This section investigates how the proposed caching technique affects the bandwidth usages of various links. For all reported results, bandwidth usage of a link l is shown in percentage which is defined as $PB(l)$ in the following equation:

$$PB(l) = \frac{AB(l)}{\sum_{s \in S} AB(s)} \times 100\% \quad (6-4)$$

where $AB(l)$ is the absolute bandwidth usage (in MB) of the link l through the whole simulation, and $S = \{Cellular, Local, V2V, Fiber Links, V2I\}$ which is the set of all the available links in the simulation. Additionally, the available cache size is reported as a multiple of the average size of the dynamic map components.

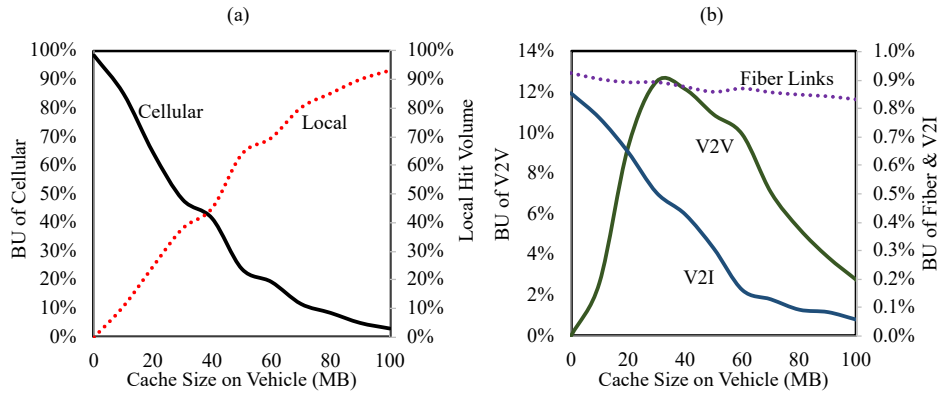


Figure 6-5: Bandwidth usages with different in-vehicle cache sizes

As shown in Figure 6-5 (a), with increasing allocated cache size in the vehicles, the bandwidth usage of cellular links reduces. This is due to the fact that the caching mechanism becomes more effective with larger available caches. As a result, vehicles often find requested map contents from within the cache of its own, other vehicles, and RSUs without having to reach the Map Application Server (see Figure 6-2) via in-vehicle cellular links. Part of this is confirmed by the local-hit statistics in Figure 6-5 (a). The graph for local hit volume indicates that with larger available cache size, more and more dynamic map data can be stored in the in-vehicle local cache. The other reasons for reducing cellular bandwidth usage can be explained using the graphs in Figure 6-5 (b). As for the V2V bandwidth usage, first it increases with increasing cache size since larger cache size sparks more inter-vehicular caching related traffic. However, when the allocated cache size is very large (i.e., larger than *30MB*), the vehicles are able to cache a lot of content within their own local cache, resulting in lower inter-vehicle caching dependency and the related traffic volume. The V2I bandwidth usage represents the traffic between the RSUs and the vehicles. With increasing allocated cache size, more dynamic map data requests are satisfied by items cached within the vehicles. As a result, the dependency on the RSUs reduces. This explains the downward trend of the V2I bandwidth usage graph in Figure 6-5 (b). Finally, the allocated cache size has very little impact on the bandwidth usage of the fiber links between the RSUs and the MAS. Those links are used mainly when an O-RSU uploads a dynamic map data to the MAS, and when the MAS pushes that data to a set of appropriate G-RSUs. Vehicle caching does not impact the usage of those links.

6.4.3 Caching Dynamics

The graphs in Figure 6-6 demonstrates the impacts of cache build-up in the network over time. The temporal bandwidth usage of a link l at time h in Figure 6-6 is also shown in percentage that is defined as $PTB(l, h)$ computed by the following equation:

$$PTB(l, h) = \frac{ATB(l, h)}{\sum_{i=0}^{100} ATB(l, i)} \times 100\% \quad (6-5)$$

while $ATB(l, h)$ is the absolute temporal bandwidth usage (in MB) of the link l at time h .

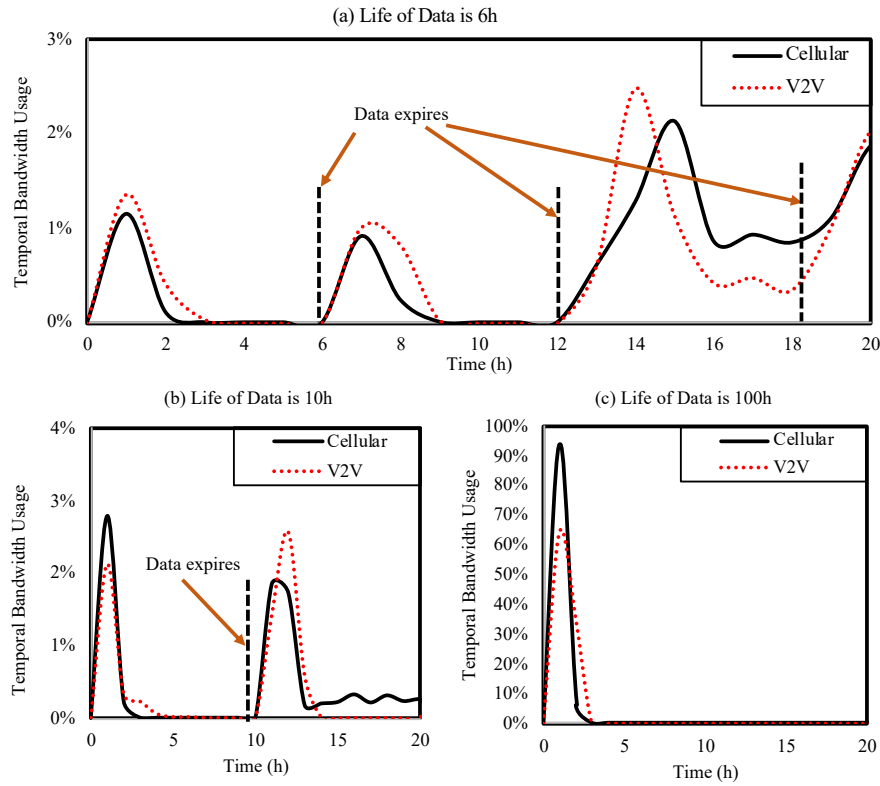


Figure 6-6: Temporal bandwidth usages of V2V and cellular links

Figure 6-6 (a) depicts the cellular and V2V bandwidth usage over time for a specific scenario in which each dynamic map data item expires after a duration of six hours. Upon each expiry of an item, the network has to clear out the corresponding entry from existing caches and needs to rebuild it. Observe that during such cache rebuilding instances, there is an initial surge of cellular bandwidth usage indicating direct download from the MAS over the cellular link. Such surges are immediately followed by high V2V bandwidth usage spurts, indicating in-vehicle caching transactions. Note that the cellular and V2V bandwidth usage could be on a relatively low level but non-zero between each pair of continuous rebuilding instances, this is because the vehicles still keep requesting and fetching the dynamic map data via cellular and V2V links during this time, since they are not able

to cache all the data in their in-vehicle caches. Figures 6-6 (b) and 6-6 (c) demonstrates similar effects when the expiry time of the content is extended to *10* hours and *100* hours respectively. Specifically, there is only one rebuilding instance shown in Figure 6-6 (c), because the next instance will be after *100* hours that is out of the range of x-axis (i.e., *20* hours) shown here. It can be observed in Figure 6-6 that the peak values of both the cellular and the V2V link bandwidths are more for higher content expiry times. The reason for that trend is as follows. In Figure 6-6 (c) there is only one rebuilding instance through the whole simulation (i.e. *100* hours), while there are multiple such instances in Figure 6-6 (a) and (b). According to Eqn. 6-5, the more the number of rebuilding is, the lower the height of each surge is. As an extreme example, in Figure 6-6 (c) the only surge of cellular link contributes almost *100%* bandwidth usage to the total $\sum_{i=0}^{100} ATB(l, i)$ in Eqn. 6-5.

6.4.4 Impact of in-RSU Caching

Figure 6-7 shows how the bandwidth usage of different links are affected by various allocated cache sizes in the RSUs. There were *10* RSUs deployed in the network for this result. Unlike caching in the vehicles, allocated cache size variation in RSUs do not significantly affect the bandwidth usages of any of the network links. This is because the coverage of the RSUs is limited as indicated by their low vehicle encounter rates shown in Figure 6-4. Such infrequent encounters provide very few opportunities to the vehicles for accessing dynamic map data cached within the RSUs.

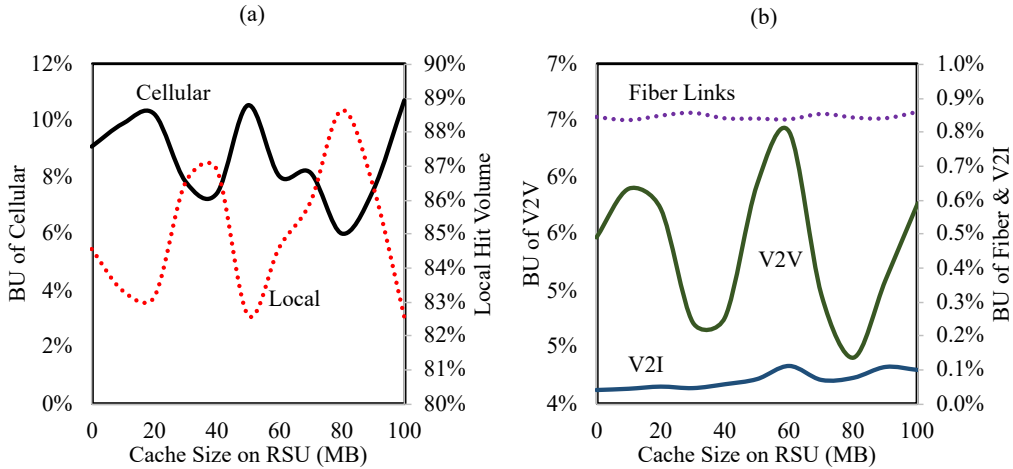


Figure 6-7: Impacts of allocated cache sizes in the RSUs

6.4.5 Larger RSU Coverage

One way of leveraging in-RSU caching for cellular bandwidth cost reduction would be to increase the RSU's V2I coverage range. Figure 6-8 shows the average vehicle encounter rate of an RSU when its range is extended to 5000 meters compared with when the range was only 1000 meters. It can be observed that the vehicle encounter rate has increased here by more than 10 times by increasing the coverage to 5000 meters.

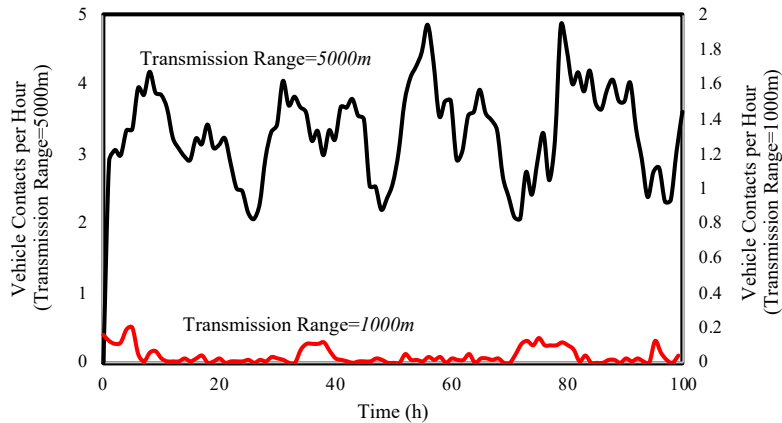


Figure 6-8: Mean vehicle encounter rate of RSUs

With such higher frequency encounters, the vehicles are able to leverage in-RSU caching more effectively. This is evident from the results in Figure 6-9 in which various network bandwidth

usages are affected in a way that is similar to the way bandwidth usages are affected by in-vehicle caching in Figure 6-5. Most notably, it is possible to reduce the cellular bandwidth usage cost by allocating sufficient amount of cache space in the RSUs. This improves in-RSU cache performance and increases the V2I bandwidth usage for the RSU-to-Vehicle downloads of dynamic map data cached in the RSUs. However, the trends of the local-hit and V2V bandwidth do not remarkably affected by in-RSU caching, since they mainly depend on in-vehicles caching. Additionally, in-RSU caching does not affect bandwidth usage of RSU-MAS fiber links, which is similar to Figure 6-7.

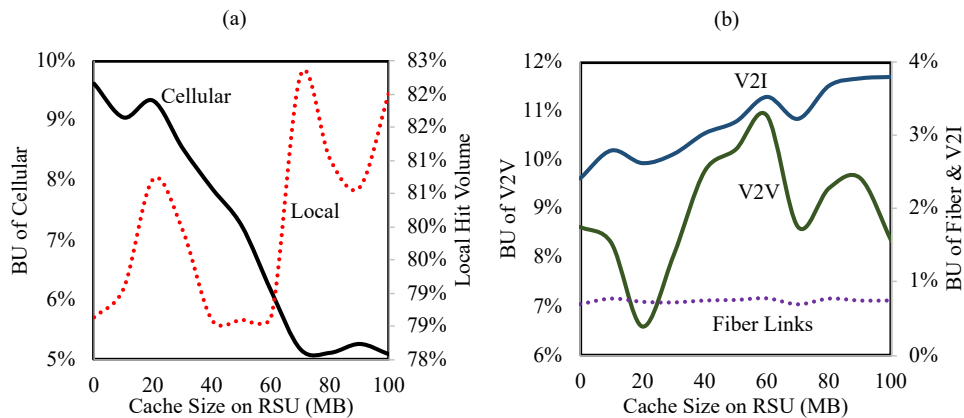


Figure 6-9: Impacts of allocated cache sizes in the RSUs with long-range V2I

The impacts of more G-RSUs for disseminating each dynamic map data in this increased V2I range scenario is shown in Figure 6-10. As expected, the cellular bandwidth usage here does go down with more G-RSUs participating in dissemination of dynamic map. The reason is efficiency of more G-RSUs with long-range V2I links. The V2I bandwidth increases due to higher frequency downloads of dynamic map data cached in the G-RSUs. Similar to Figure 6-9, more G-RSUs causes higher usage of the fiber links but does not notably impact local-hit and V2V bandwidth.

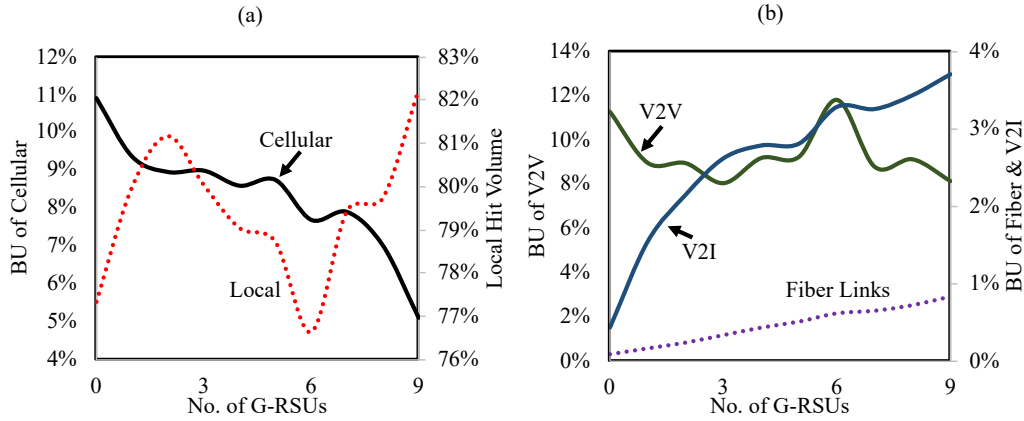


Figure 6-10: Bandwidth usage with varying number of G-RSUs with long-range V2I

6.4.6 Map Fetch Latency

The fetch latency is defined as the duration between when a vehicle generates a request for a specific dynamic data, and when it actually receives it. In Section 6.3.2, it was stated that during the Tolerable Access Delay (TAD), which is computed using Eqn. 6-2, a vehicle first searches the requested data in its local cache and then in other vehicles and the RSUs nearby before downloading it from the MAS through the on-vehicle cellular link. It follows that the fetch latency for a data is acceptable only if it is less than the TAD defined for that dynamic map data item.

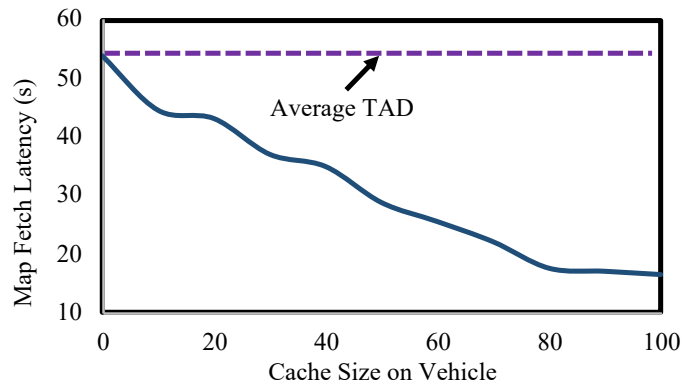


Figure 6-11: Map fetch latency with various in-vehicle cache sizes

Figure 6-11 shows the impacts of allocated in-vehicle cache size on the map fetch latency. The latency reduces monotonically as more and more cache space is added. The reason for this is as follows. Larger allocated in-vehicle cache space enables more effective caching, thus improving

the odds of finding the requested map components locally within the requesting vehicle or the nearby vehicles. Finding the data locally provides lower fetch latency compared to downloading it from the MAS over cellular links.

6.5 Summary

The chapter presented a mechanism for collaborating vehicular content caching in the context of navigational map dissemination. The research goal is to intelligently cache content in RSUs and vehicles such that the cellular bandwidth usage is minimized. We have developed a model for dynamic map data generation and used that model for designing caching algorithms for both the RSUs and the vehicles. We run detailed simulations using the DTN simulator ONE, based on a generalized network architecture. The results indicate that the proposed collaborative caching mechanism is able to reduce the cellular bandwidth usage and map fetching delay compared to infrastructure-based caching strategies.

Chapter 7: Content Dissemination Through Mobile Edge Cache

Servers in Vehicular Networks

7.1 Introduction

Besides the dynamic map described in Chapter-6, another typical application in vehicular networks is air software update for in-vehicle modules including infotainment, navigation, autonomous driving and many others. While some of the updates are non-time-critical and can wait till a vehicle is at home or at work, some are time-critical and need to be performed while on the road. This is more so now when the software for many new vehicle functions and emerging and require frequent bug-fixes and functionality upgrades. For example, a critical bug-fix for the braking system software may not be delayed till the vehicle reaches home or work. It may have to be done by temporarily pulling over, or even while it is on the move. Other example of needed on-the-road software updates includes firmware update of autonomous driving system, etc.

The current best practice is to push Software Update Packages (SUPs) from a cloud-based Manufacturer Software Provider's (MSP's) server (e.g., Ford's software update/App server) to target vehicles via cellular networks. This approach, however, involves cellular bandwidth usage cost that has to be paid to their mobile network operators (e.g. AT&T, Verizon, etc.) either by the vehicle owners or by the vehicle manufacturer depending on any service arrangements. The objective of this chapter is to explore a specific type of content caching using connection-less caching servers in order to alleviate cellular network usage costs for software updates and other vehicular contents that may require on-road downloads.

Caching of such content can usually be performed in vehicles themselves and/or roadside cache servers that are co-located with Roadside Service Units or RSUs. The vehicular caching approach relies on in-vehicle caching and vehicle-to-vehicle content sharing, which may or may

not be possible due to users' privacy concerns. More importantly, in order for such caching to be effective, V2V connections need to be sufficiently dense which may not be the case for rural transportation scenarios. Connectivity can also be sparse depending on the time of the day (e.g., nights), weather conditions, and other factors.

In the in-RSU caching approach, cellular usage cost reduction would require the roadside cache servers to anticipate demand and download content from the MSP's server through backhaul links including fibers and broadband wireless when applicable. Downloaded contents such as SUPs can then be delivered to appropriate vehicles using Dedicated Short-Range Communication (DSRC) links. While avoiding the sparse vehicle issue of V2V caching, this approach relies on sufficient number of permanently installed roadside cache servers which may not always be feasible. It is more so when the high cost of such cache servers with backhaul connectivity is considered. Even for the RSUs it is not yet clear as to which of the stakeholders, namely, township/municipalities, network service providers, or the vehicle manufacturers will eventually bear the capital investment and operating costs of the RSUs. If anything, installing permanent roadside cache servers in addition to the issues can aggravate and complicate this issue.

To address this issue, the concept of a Connectionless Edge Cache Servers (CECSs) is introduced in this chapter. The idea is to use such servers without incurring the cost of backhaul connectivity, while gaining the ability to make the cache servers mobile. Such mobility can provide a great deal of flexibility in temporarily placing them in areas with low vehicle and RSU densities and high content demands. Putting the CECS on vehicles and placing them on-demand can cater to events such as games, accidents, weather conditions, etc. Without backhaul connectivity to an MSP, the CECSs can cache content collected via DSRC links from the current-passing vehicles and provide them to future-passing vehicles over DSRC, and reduce the cellular bandwidth usage of the

vehicles in that process. As shown in Figure 7-1, the CECSs can coexist with RSUs (i.e., cache servers with backhaul connectivity) as when such infrastructures are available.

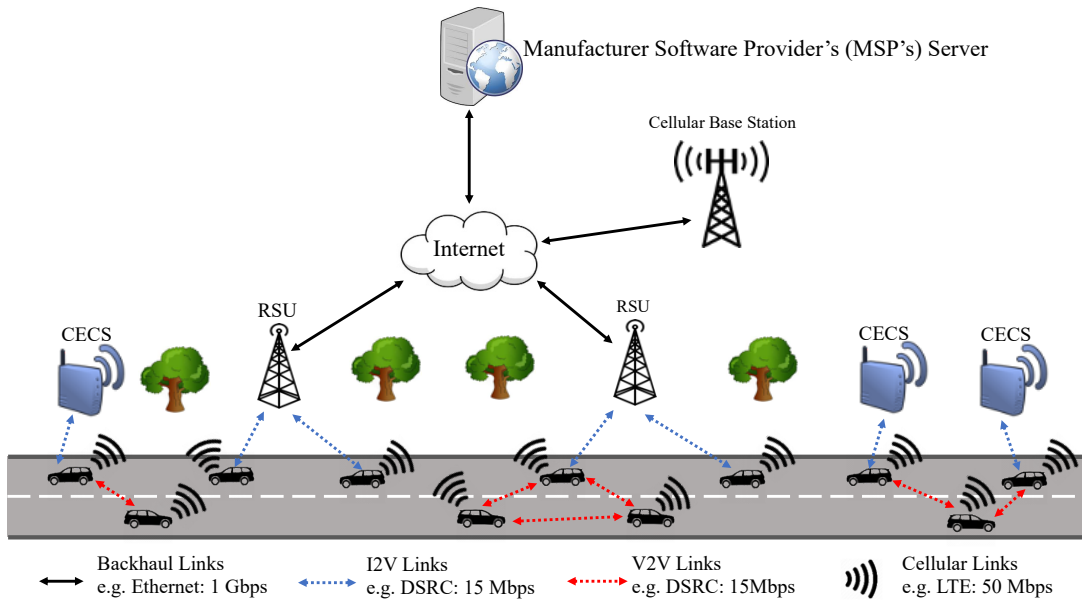


Figure 7-1: Content dissemination and caching using edge infrastructures

7.2 Content Search and CECS Operation

In the rest of the chapter, we will use vehicle software updates as introduced in the Section 7.1 as the target content type, and will describe the operation of CECS in that context. The model can be mostly extended for disseminating other content types such as navigational events, news and weather events, etc.

Once an MSP releases a new SUP for a vehicle model and year, it pushes an update notification to all the relevant vehicles over the cellular network. This notification is of small size and acts as a metadata containing ID and version number, etc., for the full SUP. Upon receiving the notification, a vehicle can start the SUP search process by generating a request for the SUP with the appropriate information received in the metadata. This is the typical software update model used by the manufacturers of most wirelessly connected devices such as phones, tablets, and the likes.

As the first step of the SUP search process, a vehicle sends enquirers to its nearby (i.e., within the DSRC range) vehicles and all CECSs via V2V and V2I links. If that search fails after a pre-defined Tolerable Access Delay (TAD), the vehicle sends a request to the MSP's server and pulls the requested SUP through the cellular link. TAD represents the duration that a vehicle is willing to wait before the request is successfully served. TAD for a SUP item is set by the MSP based on its time criticality, and it is mentioned in the metadata notification. Since the second phase of the search is expensive in terms of cellular bandwidth cost, the goal of a caching architecture will be to place the SUP within the appropriate CECSs so that cellular usage can be avoided or minimized.

Figure 1-3 in Chapter-1 summarizes the operational sequence of the CECS units. Vehicles follow a policy (i.e., to be defined later) for selectively uploading SUPs to the CECSs that they encounter on their ways. The CECSs cache such SUPs based on a set of caching protocols to be presented in Section 7.3. Finally, a vehicle can download a requested SUP from an encountered CECS whose cache contains the SUP.

7.3 Caching Mechanism

7.3.1 Content Segment Distribution

For given vehicle speed and DSRC transmission range, a single V2I or V2V contact duration may not always be sufficient to transfer a full SUP. To address this, a SUP is divided into multiple segments with size that is likely to be able to be transferred during a single contact. Let a SUP S_i be divided into n segments $\{s_{i,0}, s_{i,1}, s_{i,2}, \dots, s_{i,n-1}\}$. The segment size would be dimensioned such that in most sections it is possible to transfer one or more segments during a single V2I or V2V connection. Also, to get a complete SUP a vehicle does not need to get all its segments from a single vehicle or CECS, or even from the MSP over the cellular links. Instead, a vehicle can get different

segments via different methods and installs the SUP after constructing it from all the received segments.

According to the search model described in Section 7.2, the cellular bandwidth usage by a vehicle can be minimized depending on how many segments of a SUP is obtained by the vehicle from other vehicles and the CECSs before the TAD specified for the SUP expires. As outlined in Section 7.1, the CECS-based architecture is expected to be effective in sparse vehicle density scenarios in which the chances of getting SUP segments from other vehicles are quite low. As a result, being able to cache the correct SUPs in the correct CECSs is a key to cellular bandwidth usage cost reduction. In other words, the main algorithm problem is how to cache the SUP segments at the CECSs such that each vehicle can obtain the maximum number of SUP segments from the CECSs that it encounters within the specified TAD for that SUP.

Completeness Index (CI): This is defined as a measure to evaluate the goodness of a specific distribution of SUP segments cached across installed CECSs. After generating a SUP request at a location- l , a vehicle can move a maximum distance r within time TAD computed as follows:

$$r = v \times TAD \quad (7-1)$$

, where v is the average vehicle velocity. Let $E_{l,r}$ be the set of distinct CECSs geographically placed within the range of r from location- l . This represents the CECSs that the vehicle may be able to access during the TAD. Figure 7-2 depicts an example transportation network in which a vehicle's location when requesting a SUP and its possible locations after the TAD are shown. The CECSs within the set $E_{l,r}$ (i.e., CECSs in the red rectangle in Figure 7-2) in this scenario are also depicted. Note that the vehicle may not be able to contact every one of these CECSs in $E_{l,r}$ during the TAD, since the $E_{l,r}$ only represents the possible CECSs that the vehicle may contact during the TAD.

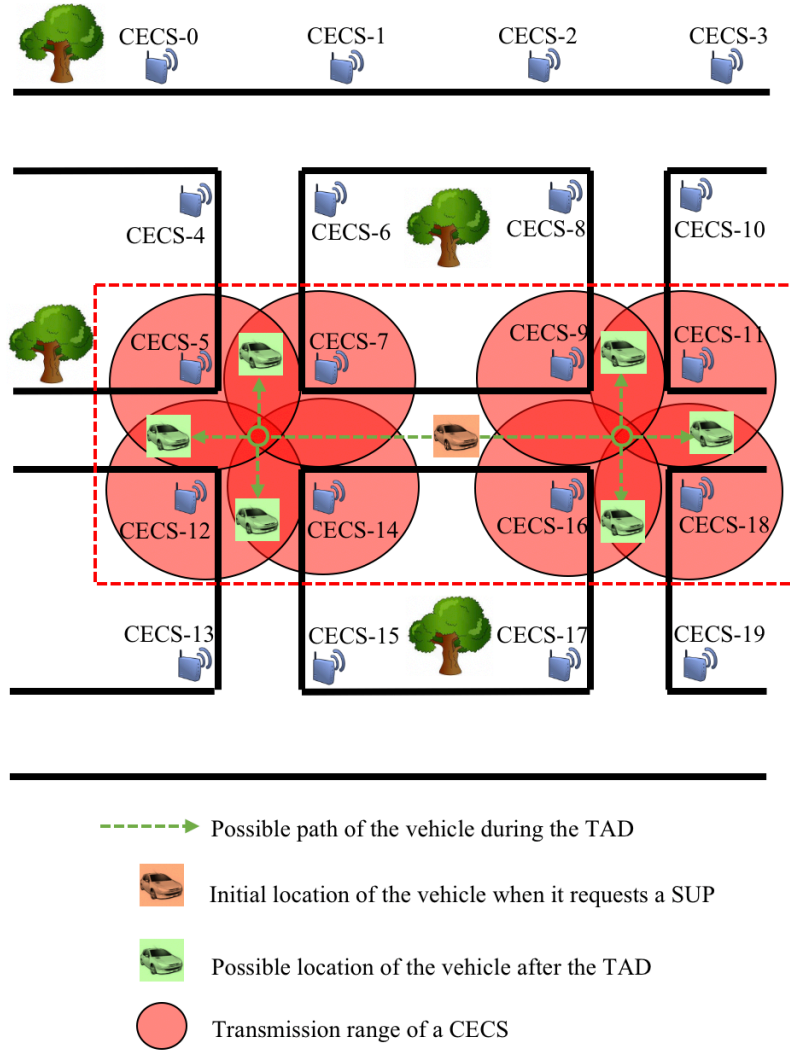


Figure 7-2: Example of the accessible CECSs to a vehicle during the TAD

The Completeness Index (CI) is defined as:

$$\mu_{E_{l,r}} = \frac{U_{E_{l,r}}}{T} \quad (7-2)$$

where the quantity $U_{E_{l,r}}$ is the number of distinct segments cached in the CECSs within the set $E_{l,r}$, and T is the total number of segments from the SUPs that can be requested.

Lemma 7-1. A higher $\mu_{E_{l,r}}$ indicates that a vehicle can obtain more segments from CECSs within the range of r from location- l .

Proof. A higher $\mu_{E_{l,r}}$ must be resulted by a higher ratio of $U_{E_{l,r}}$ to T which means more unique segments out of total available segments are cached in the CECSs in the set $E_{l,r}$ within the range of r from location- l . Thus, a higher $\mu_{E_{l,r}}$ indicates that the vehicle may obtain a greater ratio of requested segments from these CECSs within the TAD period.

The average CI μ_r of a geographical area can be computed as:


$$\mu_r = \frac{\sum_{E_{l,r} \subseteq E} \mu_{E_{l,r}}}{K} \quad (7-3)$$

, where E is the set of all the CECSs placed in that area, and each $E_{l,r}$ represents a distinct subset of E that is formed by several adjacent CECSs within the range of r from each location- l in the area. The quantity K is the total number of such distinct subsets $E_{l,r}$. Similar to Lemma 7-1, a higher μ_r indicates that on an average, a vehicle can obtain more segments of requested SUPs from the CECSs within the TAD period.

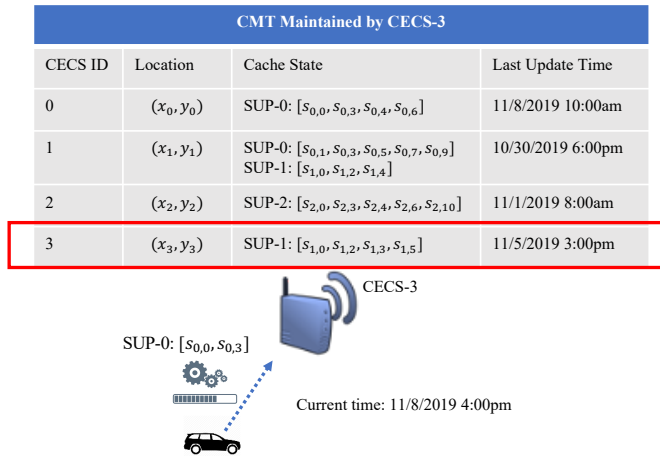
Lemma 7-2. For given T and K , μ_r can be maximized by minimizing the number of duplicate segments cached in the CECSs in each $E_{l,r}$.

Proof. For the CECSs in each $E_{l,r}$, the minimum number of duplicate segments cached represents the maximum value of $U_{E_{l,r}}$ in Eqn. 7-2. Thus $\mu_{E_{l,r}}$ in Eqn. 7-2 can be maximized by maximizing $U_{E_{l,r}}$ for a given T . Similarly, μ_r in Eqn. 7-3 can be maximized by maximizing $\mu_{E_{l,r}}$ for each $E_{l,r}$.

CMT Maintained by CECS-3			
CECS ID	Location	Cache State	Last Update Time
0	(x_0, y_0)	SUP-0: $[s_{0,0}, s_{0,3}, s_{0,4}, s_{0,6}]$	11/8/2019 10:00am
1	(x_1, y_1)	SUP-0: $[s_{0,1}, s_{0,3}, s_{0,5}, s_{0,7}, s_{0,9}]$ SUP-1: $[s_{1,0}, s_{1,2}, s_{1,4}]$	10/30/2019 6:00pm
2	(x_2, y_2)	SUP-2: $[s_{2,0}, s_{2,3}, s_{2,4}, s_{2,6}, s_{2,10}]$	11/1/2019 8:00am
3	(x_3, y_3)	SUP-1: $[s_{1,0}, s_{1,2}, s_{1,3}, s_{1,5}]$	11/5/2019 3:00pm




(a) CMT maintained by CECS-3 before accepting new segments of SUPs



(b) A vehicle uploads some segments of SUP-0 to CECS-3

CMT Maintained by CECS-3			
CECS ID	Location	Cache State	Last Update Time
0	(x_0, y_0)	SUP-0: $[s_{0,0}, s_{0,3}, s_{0,4}, s_{0,6}]$	11/8/2019 10:00am
1	(x_1, y_1)	SUP-0: $[s_{0,1}, s_{0,3}, s_{0,5}, s_{0,7}, s_{0,9}]$ SUP-1: $[s_{1,0}, s_{1,2}, s_{1,4}]$	10/30/2019 6:00pm
2	(x_2, y_2)	SUP-2: $[s_{2,0}, s_{2,3}, s_{2,4}, s_{2,6}, s_{2,10}]$	11/1/2019 8:00am
3	(x_3, y_3)	SUP-0: $[s_{0,0}, s_{0,3}]$ SUP-1: $[s_{1,0}, s_{1,2}, s_{1,3}, s_{1,5}]$	11/8/2019 4:00pm



(c) CECS-3 updates the entry about itself in the CMT after accepting and caching the new segments

Figure 7-3: Example updates of the CMTs within CECSs

7.3.2 Cache Metadata Tables (CMTs)

According to Lemma 7-1 and 7-2, the chances of finding SUP segments in the CECSs can be maximized, and therefore the cellular bandwidth usage for downloading the segments can be minimized, by minimizing the number of duplicate segments cached at the CECSs in each $E_{l,r}$. This requires that each CECS must know the cache information of other CECSs in the same $E_{l,r}$. Based on such information, a CECS can minimize or even avoid duplications. In the absence of backhaul connectivity to these connectionless servers, the only way such information can be obtained and disseminated across the CECSs is by exploiting the vehicles as information carriers.

In order to realize SUP segment ferrying by the vehicles, a data structure, namely Cache Metadata Table (CMT), is maintained in the vehicles as well as in the CECSs. As shown in Figure 7-3, each entry in a CMT corresponds to a specific CECS. It includes the corresponding CECS ID, its geographical location, the state of each cached SUP in terms of a list of the SUP's cached segments, and a Last Update Time (LUT). The CMT in a vehicle is initialized to be empty, and the CMT in a CECS is initialized with an entry about its own cache status. Once a CECS receives and caches some segments uploaded from a vehicle, the CECS may update the entry about its own cache status in the CMT (see Figure 7-3).

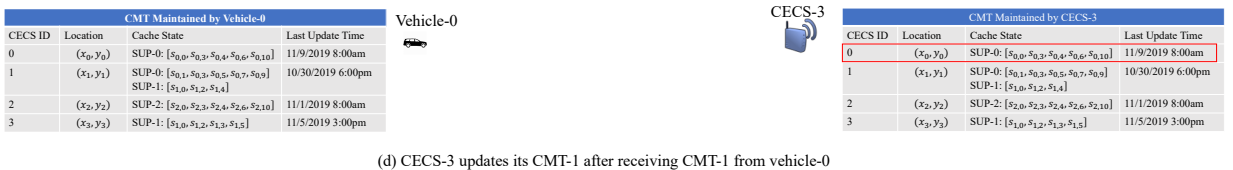
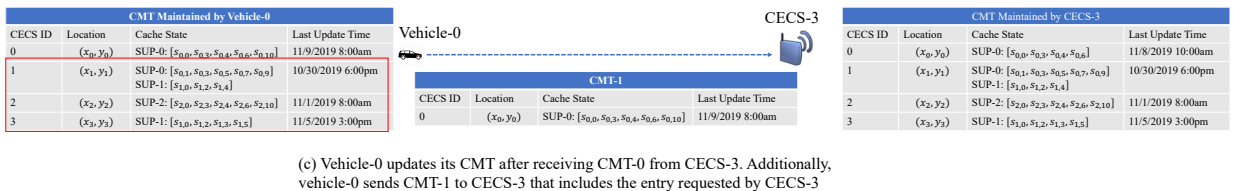
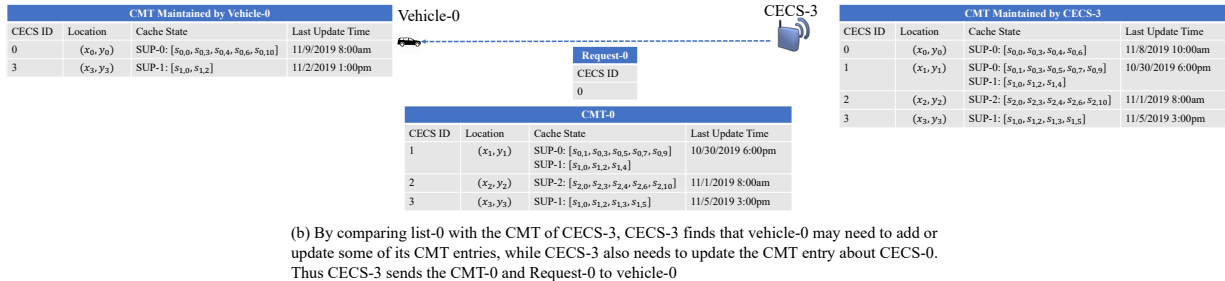
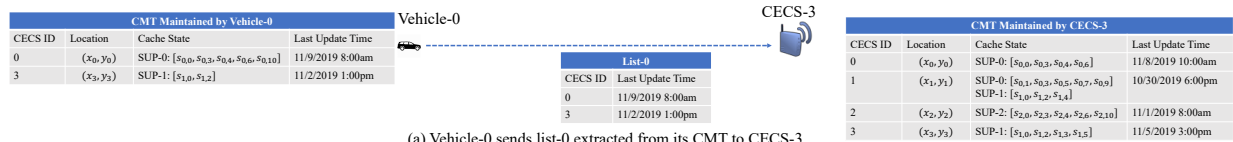


Figure 7-4: Example of CMT exchange between a vehicle and a CECS

Since a CECS is connectionless, the updates for the entries about other CECSs in its CMT happen as a result of information received from passing vehicles. Whenever a vehicle passes a CECS or another vehicle, the following information exchange happens. Figure 7-4 depicts an example of such exchange between *vehicle-0* and *CECS-3*. The exchange is initiated by *vehicle-0* by sending a summary list (i.e., metadata) *list-0* to *CECS-3*. The list includes the CECS IDs and LUTs of all the entries in *vehicle-0*'s CMT (see Figure 7-4 (a)). Once received, by comparing *list-0* with its own CMT, *CECS-3* knows that *vehicle-0* may need some entries that are either not available or out of date in *vehicle-0*'s CMT. Moreover, *CECS-3* itself may need to update or add some entries in its own CMT by receiving information from *vehicle-0*. At this point, *CECS-3* sends back two things to *vehicle-0*: 1) *CMT-0* which includes the entries that *vehicle-0* may need, and 2)

Request-0 which includes the CECS IDs of the entries that *CECS-3* may need (see Figure 7-4 (b)). Then *vehicle-0* can update its own CMT by receiving *CMT-0* from *CECS-3*. It also sends back a *CMT-1* which includes the entries requested by *Request-0* from *CECS-3* (see Figure 7-4 (c)). The exchange process is completed after *CECS-3* receives *CMT-1* from *vehicle-0* and updates its CMT (see Figure 7-4 (d)).

The above CMT exchange mechanism is designed after the Delay Tolerant Network epidemic routing [75,76] which ensures that the cache information about the SUPs are selectively disseminated from moving vehicles to the roadside connectionless cache servers. A similar CMT exchange and update process take place when two vehicles encounter each other. The full logic of the exchange of CMT information is summarized in Algorithm 7-1.

Algorithm (a): CMT exchange initiated by vehicle v_i

```

1: Input: contacting CECS  $e_j$ 
2: initialize(list-0);
3: for each entry in  $CMT_{v_i}$  at  $v_i$ 
4:   list-0.add({entry.id, entry.LUT});
5: end
6: send(list-0, ej);

```

Algorithm (b): Sending CMT entries and request list from e_j to v_i

```

1: Input: contacting vehicle  $v_i$ , summary list list-0 from  $v_i$ 
2: initialize(CMT-0);
3: initialize(Request-0);
4: //put entries to CMT-0 and request-0
5: for each row in list-0 from  $v_i$ 
6:   entry  $\leftarrow CMT_{e_j}.findById(row.id)$ ;
7:   if entry.LUT < row.LUT
8:     Request-0.add(entry.id);
9:   else if entry.LUT > row.LUT
10:    CMT-0.add(entry);
11:   end
12: end
13: // entries that vi does not have are also added to CMT-0
14: for each entry in  $CMT_{e_j}$  at  $e_j$ 
15:   if entry.id not in list-0

```

```

16:     CMT-0.add(entry);
17:   end
18: end
19: send(CMT-0, vi);
20: send(Request-0, vi);

```

Algorithm (c): updating CMT at v_i and sending requested CMT entries to e_j

```

1:   Input: contacting CECS  $e_j$ , CMT-0 and Request-0 from  $e_j$ 
2:   // update CMT at  $v_i$ 
3:   for each entry in CMT-0
4:     if entry.id in CMT $v_i$ 
5:       CMT $v_i$ .update(entry.id, entry);
6:     else
7:       CMT $v_i$ .add(entry);
8:     end
9:   end
10:  // put requested entries to CMT-1 and send it to  $e_j$ 
11:  initialize(CMT-1);
12:  for row in Request-0
13:    entry ← CMT $v_i$ .findById(row.id);
14:    CMT-1.add(entry);
15:  end
16:  send(CMT-1,  $e_j$ );

```

Algorithm (d): updating CMT at e_j

```

1:   Input: CMT-1 from  $v_i$ 
2:   // update CMT at  $e_j$ 
3:   for each entry in CMT-1
4:     if entry.id in CMT $e_j$ 
5:       CMT $e_j$ .update(entry.id, entry);
6:     else
7:       CMT $e_j$ .add(entry);
8:     end
9:   end

```

Algorithm 7-1: CMT exchange algorithm between a vehicle and a CECS

7.3.3 Cache Replacement Policy

While the CMT exchange policy above outlines the distribution of SUP information in CECS caches, it does not specify the cache replacement policies in the presence of limited storage space. The objective of cache replacement policies should be to maximize the diversity of cached SUP segments (i.e., minimize shared cached items) within each CECS subset as defined in Lemma 7-2. Such a policy can maximize the Completeness Index $\mu_{E_{l,r}}$, and lead to minimum possible cellular bandwidth usage.

Replacement Policy at a CECS: Once a CECS e_i receives a segment s_j from a vehicle, it computes the priority $P_{s_j}^{e_i}$ of s_j relative to e_i as:

$$P_{s_j}^{e_i} = \frac{1}{\theta_{s_j}^{e_i} + 1} \quad (7-4)$$

The quantity $\theta_{s_j}^{e_i}$ is the number of copies of s_j cached at the CECSs in the set of $E_{l(e_i),r}$ in which $l(e_i)$ indicates the location of e_i . A larger $P_{s_j}^{e_i}$ indicates lower number of copies of s_j cached at the CECSs within the range of r from e_i , thus the number of shared segments cached at these CECSs can be reduced by caching segments with higher priority values. Note that the CECSs in the set $E_{l(e_i),r}$ can be figured out based on the location information in the CMT at e_i . Particularly, $P_{s_j}^{e_i}$ is 1 when there is no other copy of s_j is cached at any CECS in the set $E_{l(e_i),r}$.

After receiving the segment s_j , CECS e_i caches it if there is sufficient storage space available. If not, it uses the computed priority of s_j to make a replacement decision as follows. If the priority of s_j is larger than that of the smallest priority cached segment s_k , the CECS replaces s_k by the newly received segment s_j . Otherwise, segment s_j is dropped. Note that in order to keep them up to date, the priorities of all the cached segments are updated after each CMT transfer as described in Section 7.3.2.

Replacement Policy at a Vehicle: A vehicle v_i executes its cache replacement following the same priority-based scheme used by the CECSs. However, the priorities within a vehicle cannot be computed with respect to itself, since the priority as defined in Eqn. 7-4 is computed with respect to a CECS. To address this, the priority for each cached segment in vehicle v_i is computed with respect to currently geographically nearest CECS e_c as $P_{s_j}^{e_c}$. For this logic, since the location of vehicle v_i is expected to change, before every cache replacement, the vehicle is required to recompute the priority values of all its cached segments.

7.4 Performance Evaluation

We evaluate the performance of the proposed caching mechanism using the Delay-Tolerant Networking (DTN) simulator ONE. We have chosen low vehicle-density backroads in East Lansing, Michigan as one of the test scenarios with 100 vehicles on the road. As shown in the map in Figure 7-5, 48 CECSs are placed at most of the road intersections. A DSRC transmission range of 1000 meters have been used between vehicles, and between vehicles and the roadside CECSs.

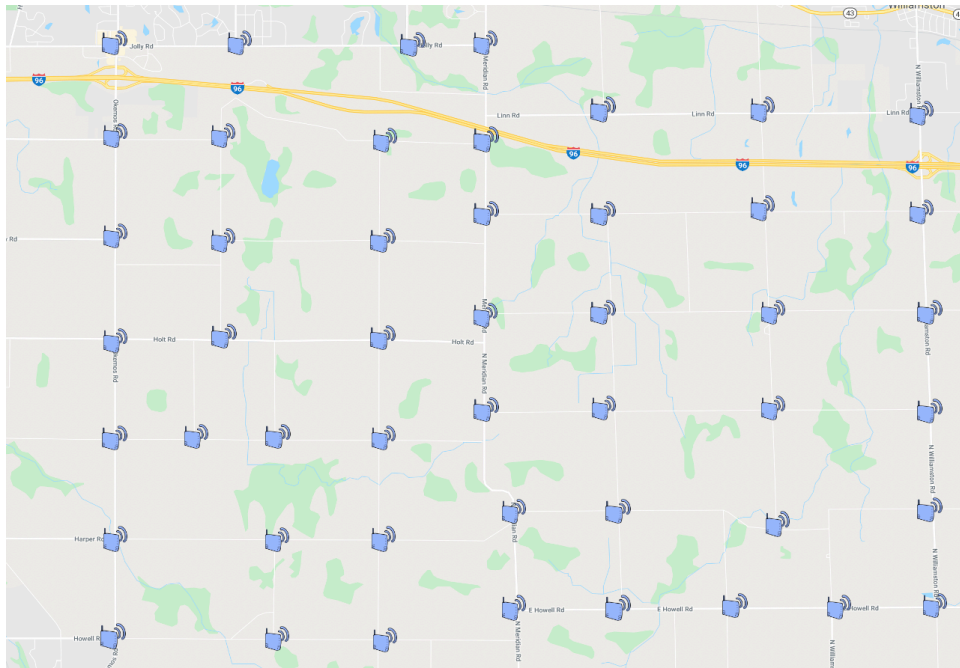


Figure 7-5: Map of the simulation scenario in the East Lansing area

The following caching mechanisms are implemented and evaluated.

Naïve Caching: In this strategy, each vehicle attempts to upload all SUP segments in its local cache to each CECS that the vehicle comes to contact with. Also, each CECS attempts to cache every segment it receives from the passing vehicles. If the cache space in a CECS is full, the CECS always replaces the first segment in its cache with the newly received segment. The vehicle also runs the same naïve cache replacement policy as done by the CECSs.

Round-robin Upload and Demand-based (RD) Caching: In this policy, each vehicle attempts to upload SUP segments from its local cache to each CECS it encounters following a round-robin policy. For example, a vehicle uploads the *0th* to *9th* segments from its cache to the first CECS it encounters, and then it uploads the *10th* to *19th* segments to the second CECS it encounters, and so on. Each CECS attempts to cache every SUP segment it receives. Additionally, each CECS also counts the number of requests from vehicles for each segment. That number is used for estimating the demand of a segment. The segment with a lower demand is replaced with one higher demand when the cache space on a CECS is full. The vehicles under this policy run the replacement policy used in Naïve Caching as stated above.

CMT-based Caching: This is the proposed smart caching mechanism in this chapter. Under this policy, each vehicle attempts to upload all the segments in its local cache to each CECS that it encounters. Each CECS attempts to cache every segment it receives. However, if the cache space in a CECS or in a vehicle is full, then it runs the CMT-based cache replacement policies presented in Section 7.3.3.

Pre-filled Cache at CECS: This policy renders theoretical best caching performance. Here, the cache of each CECS is manually pre-filled with SUP segments such that the Completeness Index μ_r in Eqn. 7-3 is maximized before an experiment begins. Such segment placements are static in that no

cache replacement is done after the pre-filling, and the vehicles never upload any segment to the CECSs so that the optimal pre-filled cache configurations are never changed. The vehicles simply run the Naïve caching policy described earlier. Before a simulation experiment begins, the CECSs caches are pre-filled, while the vehicle caches are initialized empty. The latter gradually gets filled up using Naïve caching.

Parameter	Default Value
No. of CECSs	48
No. of vehicles	100
No. of available SUPs	10
No. of sub-segments in each SUP	100
size of each sub-segment	1MB
Ratio about cache size α on each CECS	0.5
Ratio about cache size β on each vehicle	0.2
Transmission range of V2V and V2I (e.g. DSRC)	1000 meters
Data transfer rate of V2V and V2I	16Mbps
Data transfer rate of cellular (e.g. LTE)	50Mbps
Speed of each vehicle	64km/h (40 mph)
TAD for each request	600s
Simulation duration	30 hours

Table 7-1: Baseline parameters used in the experiments

Unless stated otherwise, all parameters are set to the baseline values as shown in Table 7-1. The mobility traces of the vehicles are generated based on the East Lansing road map shown in Figure 7-5. Each vehicle enters into the area approximately every 16 minutes, and then moves along the roads using a random walk model until it leaves the area network. The vehicle entry interval is deliberately kept high so that the resulting vehicle density is as low as it is observed in rural backroads such as the East Lansing area. The experiment is run for a total of ten different Software Update Packages (SUPs) that the vehicles may require/request for all its internal electronics and processing modules. Each SUP is divided into 100 segments of size 1MB [77]. Each vehicle may request all the SUPs when it enters the network area as shown in Figure 7-5. For different

experiments, the vehicles and the CECSs are made to run different caching mechanisms as described above.

The cache space on each CECS is represented by the parameter α which indicate the ratio of the available space (i.e., in terms of number of segments) to the total number of segments in the system, which is 1000. For example, when α is 1, each CECS can cache all the available segments in the network. Similarly, the cache size on each vehicle is represented by the parameter β defined in the same manner.

7.4.1 Impacts of Cache Space in the CECSs (α)

For all the protocols, Figure 7-6 (a) depicts the impacts of available CECS cache storage size (i.e., α) on the content retrieval rate via the cellular network, which the proposed caching mechanism attempts to reduce. As expected, the figure shows the decrease of cellular network usage with increasing cache space in the CECSs. As shown in Figure 7-6 (b), this reduction in cellular usage is caused due to larger local hit rates at the CECSs. Meaning, with larger α values, each CECS can cache a greater number of sub-segments that can be retrieved by the vehicles. The practical implications of these results are that fewer requested SUP segments are downloaded from the MSP's server through the cellular networks, leading to cellular usage reduction.

It can be observed that the proposed mechanism CMT performs better than all other strategies except the manually pre-filled one, which represents the performance upper bound. Better performance of CMT is due to its priority computation strategy, which helps reducing the overall cached segment duplications in the CECSs, thus improving the completeness index, as defined in Section 7.3. The cellular retrieval rate in the round robin approach (i.e., RD) is lower than Naïve, because the number of shared/duplicated segments cached at the CECSs in RD is reduced by its round-robin upload mechanism and demand-based replacement policy. Such mechanisms in RD,

however, does not perform as good as the priority-based approach in CMT, which achieves a better complete less index by distributing SUP segments across the CECSs more evenly.

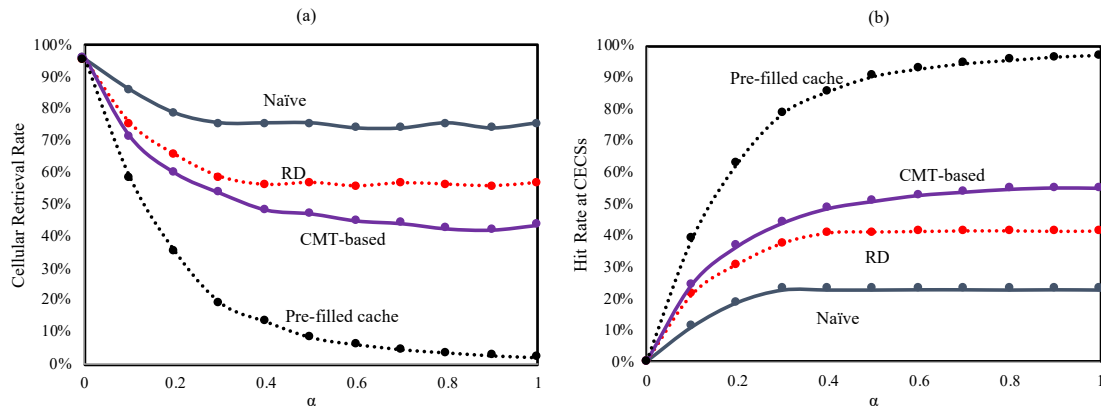


Figure 7-6: Cellular retrieval rate and hit rate at CECSs with various α

The average completeness indices (CIs), as computed in Eqn. 7-3, for all the protocols for different in-CECS cache size are shown in Figure 7-7. It can be seen that the CI values and their trends with varying in-CECS cache sizes for different protocols are consistent with the cellular usage results presented in Figure 7-6.

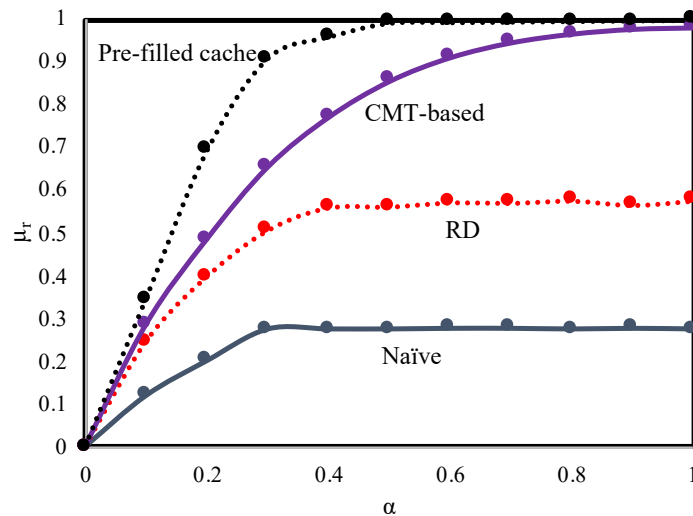


Figure 7-7: Impacts of CECS cache space α on Completeness Index μ_r

Content Delivery Latency (CDL) for all the protocols are shown in Figure 7-8. CDL is defined as the interval between when a vehicle first puts out the request for a SUP segment and when it

acquires the segment. As presented in Section 7.2, the vehicle avoids fetching the content over cellular link for the Tolerable Access Delay (TAD) specified in the metadata for the corresponding SUP. In the event that the vehicle cannot fetch using non-cellular V2V and V2I links during TAD, it gets the SUP from the MSP over the cellular link. It follows that the CDL for a segment is bounded by the TAD plus a short latency (i.e., negligible in comparison to TAD) to get the content from the MSP. For the results in Figure 7-8, the TAD was set to 600 seconds or 10 minutes. It should be noted that the CDL above is defined on a per-segment basis as opposed to on a per-SUP basis. The latter defines the delay between when the request for the first segment of a SUP is produced and when the last segment of the SUP is obtained

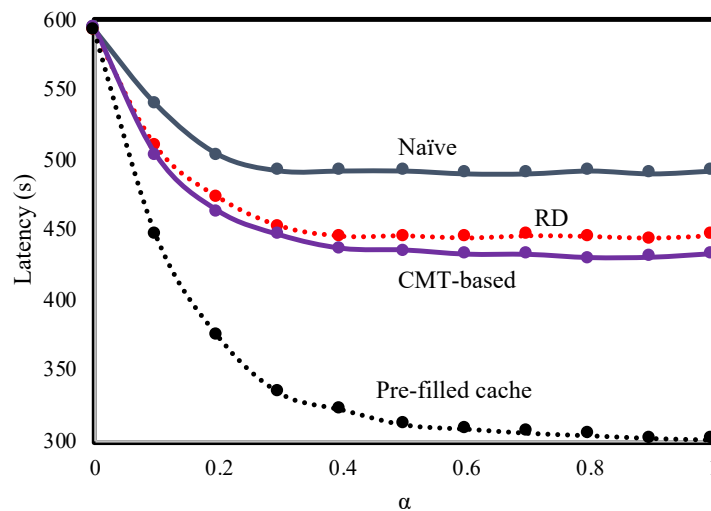


Figure 7-8: Content Delivery Latency for different caching mechanisms

As expected, CDLs for all the algorithms in Figure 7-8 are bounded by the TAD of 600 seconds, and they improve with higher available cache size in the CECSs. This is mainly due to higher hit rates as depicted in Figure 7-6. Also to be observed that the proposed CMT scheme offers the best latency numbers, which is also due to its higher hit rates as can be seen in Figure 7-6. Higher hit rates allow more frequent SUP segment accessing from the CECSs, rather than waiting for the TAD and fetching from the MSP via cellular networks.

A notable overhead of the proposed CMT based mechanism is the bandwidth used for exchanging the Cache Metadata Tables (i.e., CMTs) themselves. Figure 7-9 shows the usage of CMT-exchange bandwidth as a percentage of the bandwidth used for downloading the user data, which is the SUPs. As expected, the overhead does increase with increased cache storage space in the CECSs due to more CMT transfers between the vehicles and the CECSs. The absolute overhead, however, is limited to less than a percent even when the available caches storage space in the CECSs is very high.

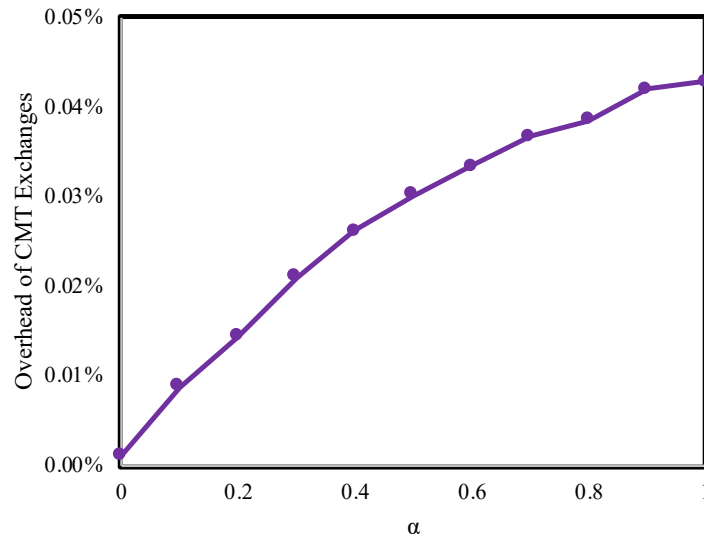


Figure 7-9: CMT exchange overhead with varying storage in the CECSs

7.4.2 Impacts of Available In-vehicle Cache Storage Space (β)

Available cache storage space in the vehicles is indicated by the factor β , as defined in Section 7.4. As shown in Figure 7-10, with larger β , the cellular retrieval rate for all the caching mechanisms are lower due to higher hit rates at the CECSs. This is because with larger available cache storage space, each vehicle can cache more SUP segments, thus being able to upload more of them to the CECSs that it encounters. Subsequently, more segments can be fetched from the CECSs by other vehicles.

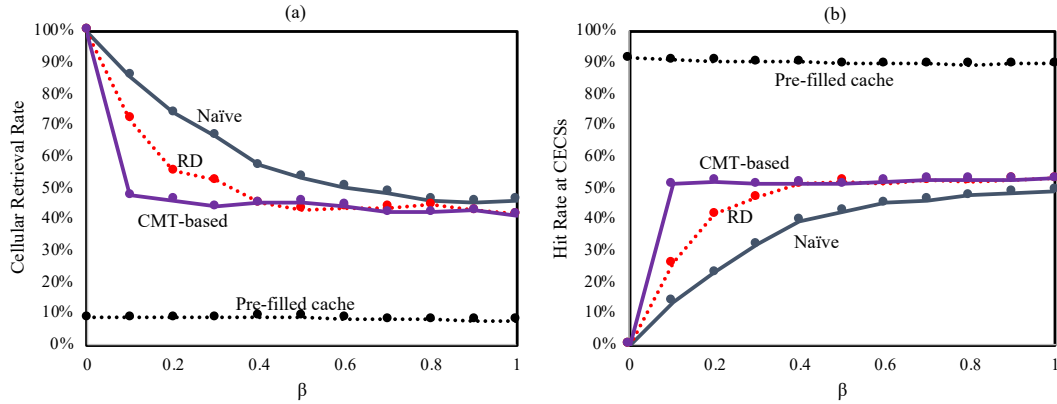


Figure 7-10: Impacts of varying in-vehicle cache storage space β

Also to be noted that the cellular retrieval rate of CMT-based caching is lower than RD and Naïve, while its hit rates at CECSs are higher. This further indicates the benefits of CMT's priority based replacement policy as observed in the results in Figure 7-10. For the pre-filled caching strategy, however, since no vehicle-CECS content transfer takes place, its performance in terms of cellular retrieval rate does not depend on the factor β .

The final observation in Figure 7-10 is that the usage of cellular network saturates and becomes almost the same for all the caching mechanisms when β becomes larger than about 0.4. This is because with large available in-vehicle storage space, a vehicle is able to cache most of the available segments and upload them to the encountered CECSs. In this case, the benefit of CMT-based caching mechanism is no longer significant.

Better caching performance of the proposed CMT-based approach can be further validated by its superior Complete Index larger μ_r , as shown in Figure 7-11. Completeness Index indicates how the SUP segments are uniformly distributed across the CECSs with minimum duplication. By using the priority-driven cache replacements, the CMT mechanism can place segments more uniformly, and in that process is able to reduce the cellular link usage compared to the other mechanisms.

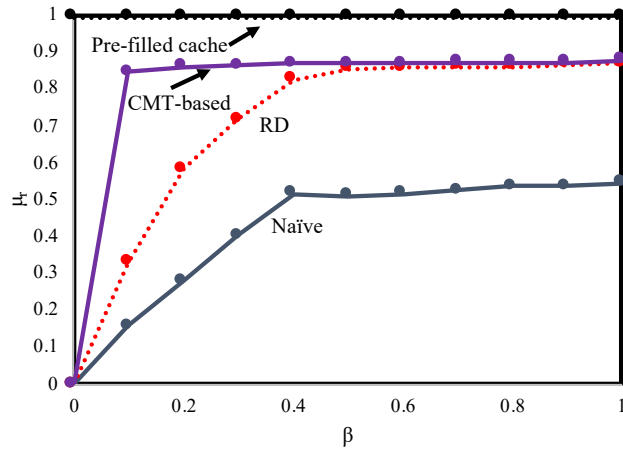


Figure 7-11: Impacts of in-vehicle cache storage space on average CI (μ_r)

The impacts of available in-vehicle cache storage space on Content Delivery Latency (CDL) are shown in Figure 7-12. The Tolerable Access Delay (TAD) is set to 600 seconds. As expected, larger cache storage allows more SUP segments to be retrieved before the TAD expires, thus leading to smaller CDL. Also, the CDL for CMT-based caching is lower than the others except the Pre-filled cache for reasons explained in Figure 7-10.

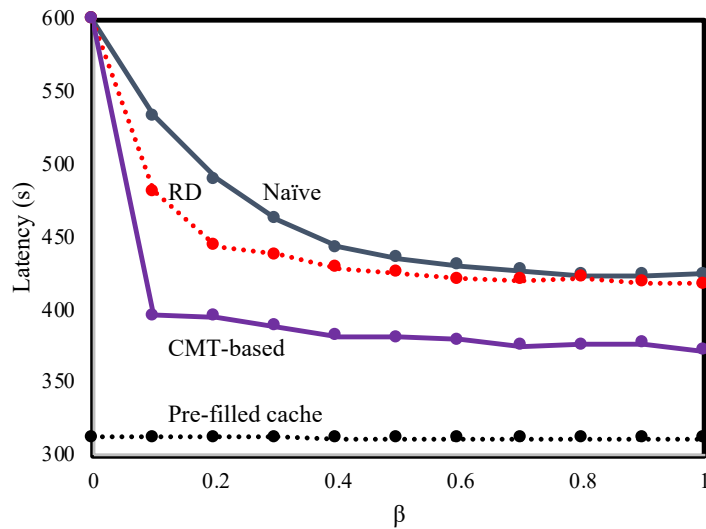


Figure 7-12: CDL with different in-vehicle cache storage space β

Figure 7-13 depicts that the overhead of CMT table exchange generally reduces with larger in-vehicle cache storage. This is because with smaller storage space, each vehicle caches a small

number of SUP segments and uploads them to the encountered CECSs. In such scenarios, most of the vehicles caches non-duplicated SUP segments, whose information needs to be exchanged during the CMT exchange process. This increases the CMT exchange overheads. With larger cache storage space, there exist more duplicate segments among the vehicles and CECSs, leading to less amount of CMT exchange, thus leading to reduced overheads. It should be observed that even when the CMT exchange is the maximum, it is still very small, only less than approximately 0.04% of the actual data volume of the downloaded content.

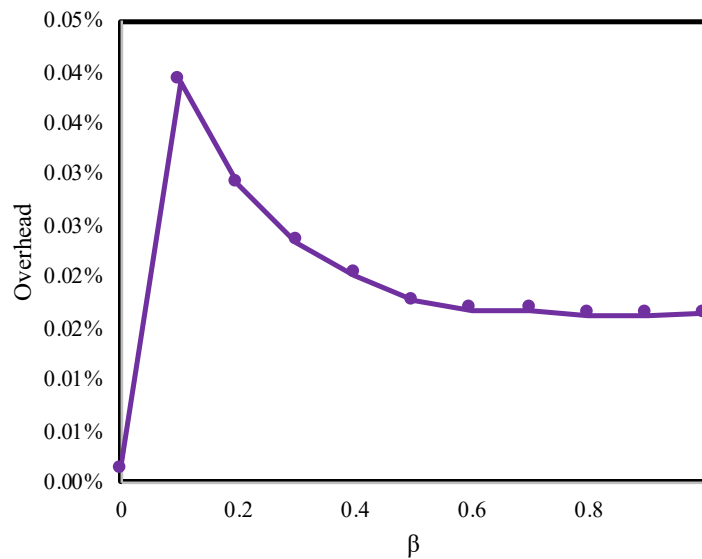


Figure 7-13: CMT exchange overhead with varying vehicle cache space

The plots in Figure 7-14 demonstrate the dynamics of cache build-up in the network over time. It can be observed that the cellular retrieval rate and μ_r of all the caching mechanisms except Pre-filled cache start to converge after around 800 minutes in the experiment. The performance of all the protocols maintain the same relative trend as observed in earlier graphs.

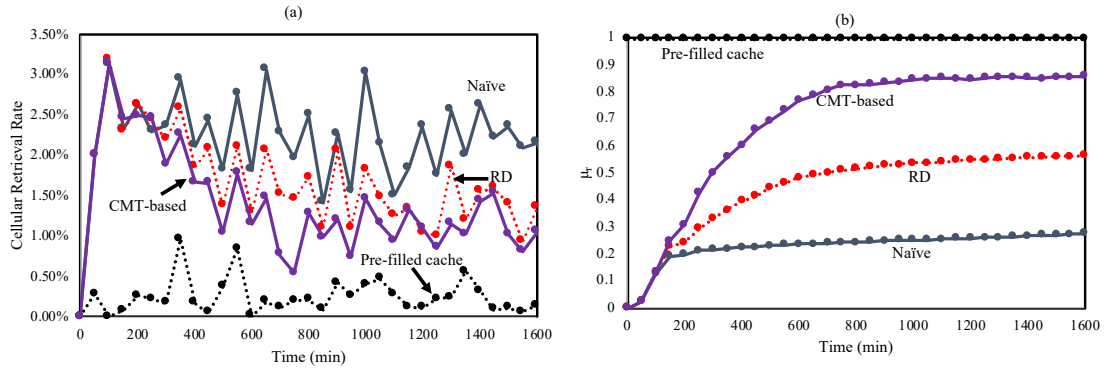


Figure 7-14: Evolution of cellular retrieval rate and the completeness index μ_T

7.4.3 Impacts of Tolerable Access Delay (TAD)

Figure 7-15 depicts how TAD, which is the duration before which a vehicle downloads a SUP segment from the Manufacturer Software Provider's (MSP's) server, affects caching performance. Initially, a larger TAD allows vehicles more time to fetch segments from other vehicles and the CECSs, thus leading to higher cache hit rates and lower cellular usage. Consistent with the prior results, for all TAD values, the CMT-based strategy does better compared to RD and Naïve caching due to its better priority based replacement policy.

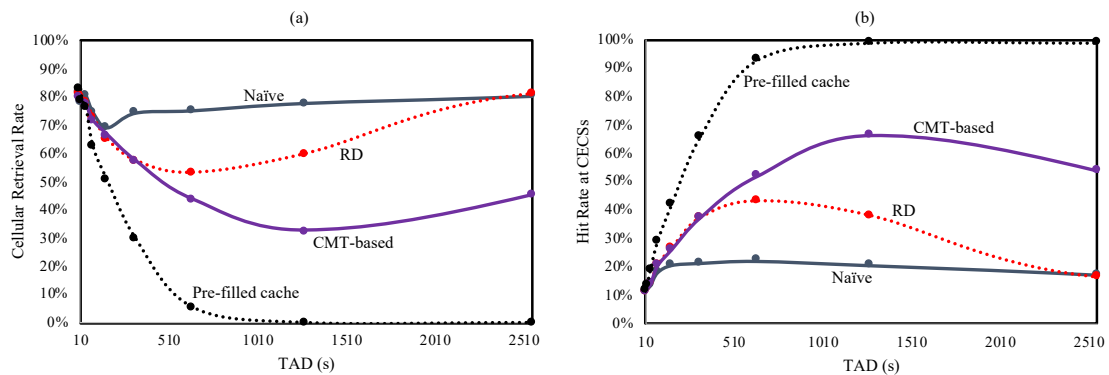


Figure 7-15: Impact of TAD on cellular retrieval rate and hit rate at CECSs

When the TAD becomes too large, however, the trend reverses. For example, with TADs larger than 600s for RD, and 1200s for CMT, the cellular retrieval rates increase with TAD. The only exception is the pre-filled case. This trend reversal happens due to reason explained below.

While larger TADs allow more time to fetch content via non-cellular mechanisms, it also reduces the rate of cache growth in the vehicles and CECSs. For smaller TAD values, cellular usage reduces because the first effect of larger TAD dominates. As TADs become large enough to compare with the vehicles' sojourn time in the network, the second effect of large TAD dominates. This results in partially empty vehicle caches (i.e. the vehicles leave the network before their caches get a chance to be filled up). Subsequently, the CECS caches also remain partially empty. These cause overall reduction of cache effectiveness, thus leading to higher cellular usage with increasing TADs. Since pre-filling is done manually, it is immune from these effects.

7.4.4 Comparison with Sparse-CECSs Network

Results in this section report caching performance with smaller number of cache edge servers (i.e., 24 in Figure 7-16) compared to what has been used for the results presented so far (i.e., 48 in Figure 7-5).

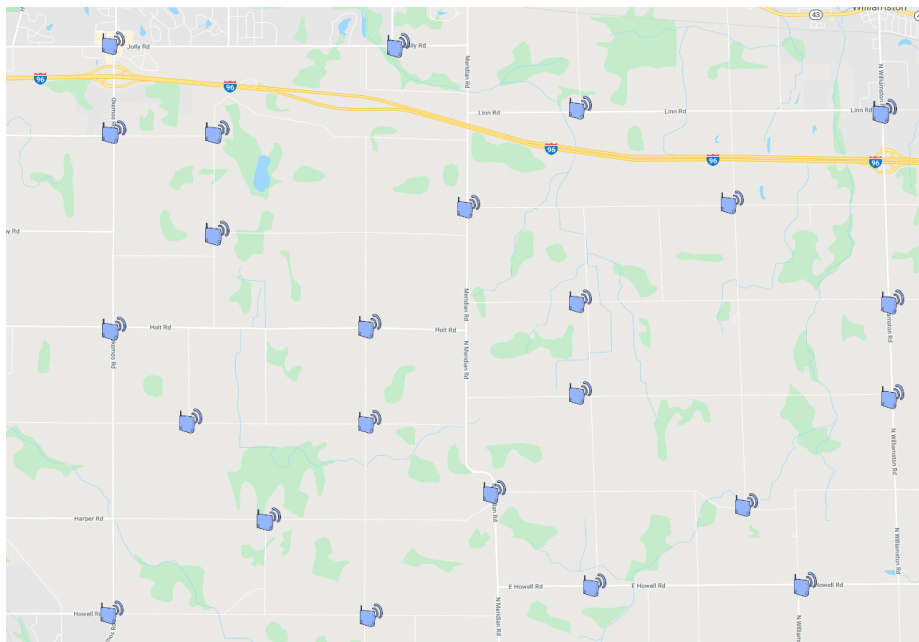


Figure 7-16: East Lansing scenario with reduced number of CECSs

Figure 7-17 depicts caching performance for both high- and low-CECS-count scenarios with varying amount of available cache storage space in the CECSs. While the cellular usage rates follow very similar trends with varying cache storage space, the overall cellular usage is lower for higher CECS-count scenario. This is intuitive since with more edge cache servers in the network, the caching efficiency is higher, thus leading to lower cellular usage. It should also be observed that the benefits of the proposed CMT-based mechanism compared to the other caching schemes also shrinks due to insufficient room for caching due to fewer available cache edge servers.

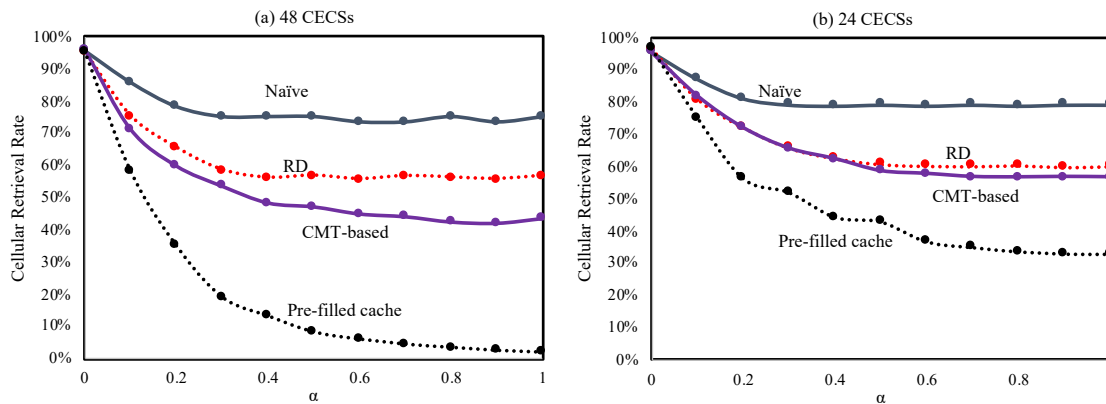


Figure 7-17: Cellular usage with various α with different CECS-counts

Very similar performance trends can also be observed in Figure 7-18, which shows the impacts of varying in-vehicle caching storage space. The only exception here is that the proposed CMT-based mechanism can retain its cellular usage advantages even for the low CECS-count scenario.

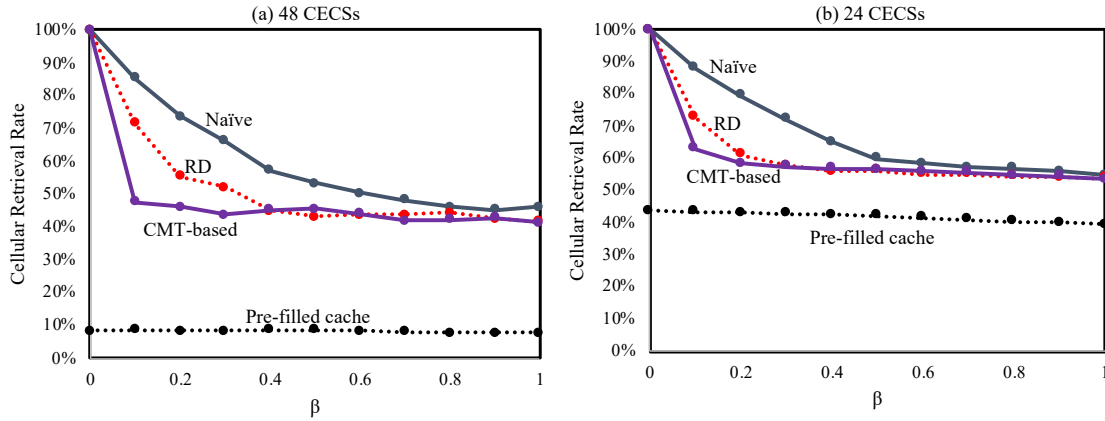


Figure 7-18: Cellular usage with β with different CECS-counts

7.4.5 Caching Performance in a Synthetic Network

In order to demonstrate repeatability of the prior results, caching experiments are done in a synthetic network scenario as follows. As shown in Figure 7-19, 36 CECSs are uniformly placed at every intersection (i.e., separated by $2km$) of a transportation network. Vehicle mobility is generated following the same model as in East Lansing scenario. All the other settings are kept the same as in Table 7-1.

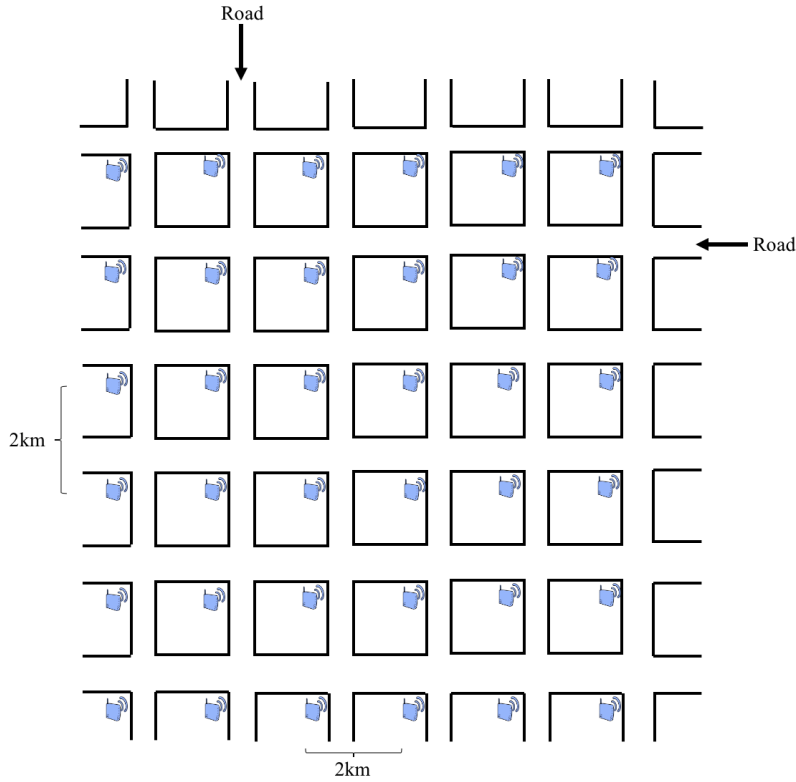


Figure 7-19: CECS placement in a synthetic network scenario

Figure 7-20 and Figure 7-21 depict how the available cache space in the CECSs and the vehicles (i.e., α and β respectively) affect caching performance in this scenario. These results show trends similar to those in Figures 7-6 and 7-10 for the East Lansing scenario. One notable observation is that the proposed CMT-based caching outperforms all the non-pre-filled schemes due to its novel priority mechanism.

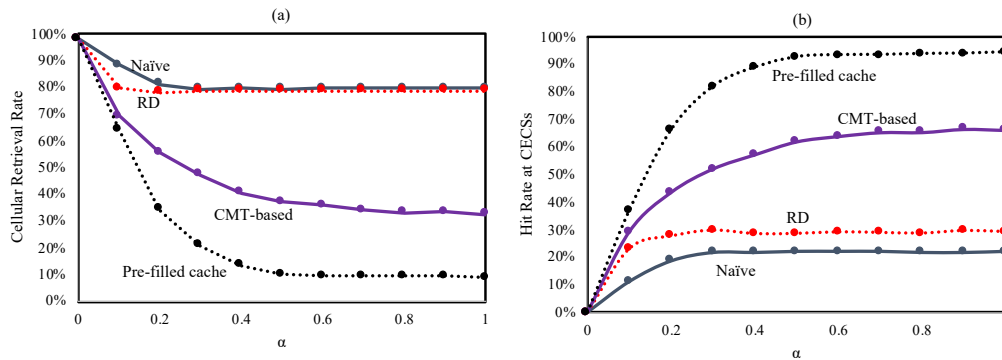


Figure 7-20: Impacts of CECS cache space in the synthetic scenario

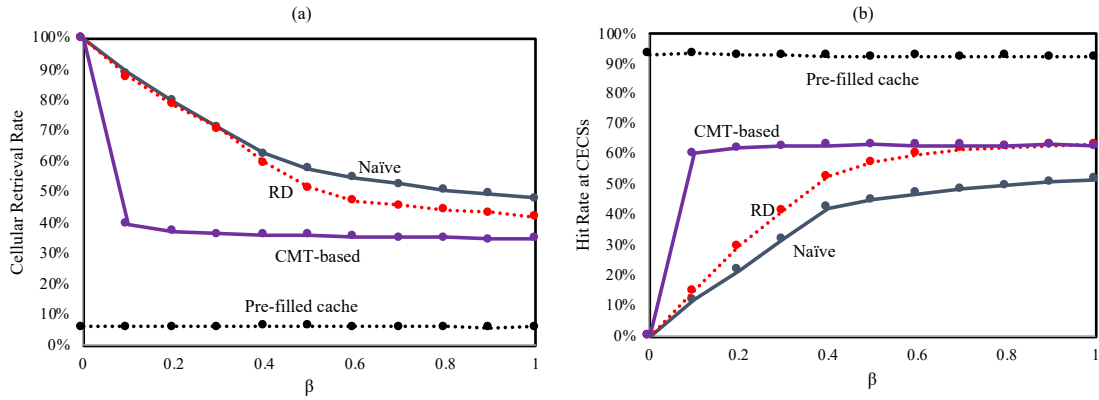


Figure 7-21: Impacts of vehicle cache space in the synthetic scenario

7.5 Summary

This chapter presents a novel caching mechanism based on connectionless roadside edge cache servers in vehicular networks. The goal is to intelligently cache content within the vehicles and the edge servers so that majority of the vehicle-requested content can be obtained from those caches, thus minimizing the amount of cellular network usage. The mechanism is specifically designed and investigated in the context of vehicle software update packages (SUPs) that can be divided into multiple segments, and the segments are considered to be the basic unit of caching. A novel caching mechanism is developed in which the cache space in the edge server is filled up by uploading SUP segments from the vehicles. In the absence of backhaul connectivity in the edge servers, the vehicles ferry content across the edge servers to build their optimal distribution so that the cellular usage from the vehicles is minimized. We have implemented the scheme using ONE simulator and compared it with various other caching mechanisms including a manually pre-filled technique that provides a performance upper bound. It was shown that the proposed mechanism outperforms the other schemes in two different network scenarios.

Chapter 8: Future Work

8.1 Introduction

The objective of the thesis is to design optimal caching mechanisms such that under different network topologies and node mobilities the network-wide content provisioning cost is minimized. A key question for content caching is how to store content in nodes so that the overall content provisioning cost in the network is minimized.

In Chapter-3, an incentive based cooperative content caching framework is developed for Social Wireless Networks (SWNETs) in which content demands are hierarchically heterogeneous. In Chapter-4, the caching mechanism proposed in Chapter-3 is applied on the scenario of mobility wireless networks. Unlike Chapter-3, in this scenario the connection between each pair of nodes is not stable anymore because a node may dynamically join or leave a network. In Chapter-5, a D2D cooperative caching framework is proposed for streaming video with heterogeneous quality demands in SWNETs. In Chapter-6, a vehicular content caching mechanism is presented for disseminating navigational maps while minimizing cellular network bandwidth usage. Finally, in Chapter-7 a content caching method based on the connectionless edge cache servers is proposed for reducing the cellular usage for dissemination of software update packages.

The research in this thesis can be extended along the following directions.

8.2 Machine Learning Models for Content Caching and Dissemination

The content demand models in Chapter-3, Chapter-4 and Chapter-5 are all based on Zipf distributions, while the demand model in Chapter-6 is based on a simple inverse proportion. Additionally, in Chapter-7 it is assumed that the content popularity is equal for every content. While providing reasonable models for caching architectures, these demand models do not represent the real scenarios in which content preferences may follow various distributions. On the other hand, a

popularity distribution of a set of content may dynamically change over time. Additionally, none of the developed caching mechanisms in the thesis involves the mobility pattern of nodes. To improve the performance of caching, a future work direction on this topic is to develop machine learning models that can be involved in caching strategies for predicting: 1) content demand in the networks, and 2) spatiotemporal localities of node movements

8.3 Placement of Edge Cache Servers

In Chapter-7, a novel edge cache server is proposed for reducing the cellular usage for dissemination of software update packages. Although the study in Chapter-7 is under given placements of such edge cache servers, it is useful to know how a placement of edge cache servers affects the performance of a caching mechanism. Therefore, a future work direction is to investigate the impacts of edge cache server placement on caching performance, and develop mechanisms for edge cache server placement for performance optimization.

8.4 Handling Selfishness

The potential for earning peer-to-peer rebate may promote selfish behavior. A selfish user is one that deviates from the network-wide optimal caching policy in order to earn more rebates. Any deviation from the optimal policy is expected to incur higher network-wide provisioning cost. A future work direction is to investigate the impacts of selfishness on the performance of caching mechanisms. This can be accomplished by defining the number of selfish nodes, and the level of selfishness of such nodes. Based on these settings, a solution for detecting and combating the selfishness should be developed.

BIBLIOGRAPHY

BIBLIOGRAPHY

- [1]. F. A. Silva, et. al., "Vehicular Networks: A New Challenge for Content-Delivery-Based Applications." *CSUR* 49, no. 1, 2016.
- [2]. C. Loebbecke, A. Soehnel, S. Weniger, and T. Weiss. "Innovating for the mobile end-user market: Amazon's Kindle 2 strategy as emerging business model." In *Mobile Business and 2010 Ninth Global Mobility Roundtable (ICMB-GMR)*, 2010 Ninth International Conference on, pp. 51-57. IEEE, 2010.
- [3]. R. TB Ma, D. M. Chiu, J. CS Lui, V. Misra, and D. Rubenstein. "On cooperative settlement between content, transit, and eyeball internet service providers." *IEEE/ACM Transactions on networking* 19, no. 3 (2011): 802-815.
- [4]. B. M. Maggs, and R. K. Sitaraman. "Algorithmic nuggets in content delivery." *ACM SIGCOMM Computer Communication Review* 45, no. 3 (2015): 52-66.
- [5]. Yoon, Jongwon, Peng Liu, and Suman Banerjee. "Low-cost video transcoding at the wireless edge." In *2016 IEEE/ACM Symposium on Edge Computing (SEC)*, pp. 129-141. IEEE, 2016.
- [6]. C. M. Silva, A. LL Aquino, and W. M. Jr. "Deployment of roadside units based on partial mobility information." *Computer Communications* 60 (2015): 28-39.
- [7]. J. Guo, and Nathan Balon. "Vehicular ad hoc networks and dedicated short-range communication." University of Michigan (2006).
- [8]. S. Goel, T. Imielinski, and K. Ozbay. "Ascertaining viability of WiFi based vehicle-to-vehicle network for traffic information dissemination." In *Proceedings. The 7th International IEEE Conference on Intelligent Transportation Systems (IEEE Cat. No. 04TH8749)*, pp. 1086-1091. IEEE, 2004.
- [9]. Y. Zhang, E. Pan, L. Song, W.Saad, Z. Dawy, and Z. Han. "Social network aware device-to-device communication in wireless networks." *IEEE Transactions on Wireless Communications* 14, no. 1 (2015): 177-190.
- [10]. Y. Zhang, L. Song, C. Jiang, N. H. Tran, Z. Dawy, and Z. Han. "A social-aware framework for efficient information dissemination in wireless ad hoc networks." *IEEE Communications Magazine* 55, no. 1 (2017): 174-179.
- [11]. L. Wang, I. Moiseenko, and L. Zhang. "Ndnlive and ndntube: Live and prerecorded video streaming over ndn." *NDN*, Technical Report NDN-0031 (2015).
- [12]. Conklin, Gregory J., et al. "Video coding for streaming media delivery on the Internet." *Circuits and Systems for Video Technology*, *IEEE Transactions on* 11.3 (2001): 269-281.

- [13]. L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang. "Named data networking." *ACM SIGCOMM Computer Communication Review* 44, no. 3 (2014): 66-73.
- [14]. K. Shah, A. Mitra, and D. Matani. "An $O(1)$ algorithm for implementing the LFU cache eviction scheme." no 1 (2010): 1-8.
- [15]. A. I. Vakali, "LRU-based algorithms for Web cache replacement." In *International conference on electronic commerce and web technologies*, pp. 409-418. Springer, Berlin, Heidelberg, 2000.
- [16]. K. Psounis, and B. Prabhakar. "A randomized web-cache replacement scheme." In *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*, vol. 3, pp. 1407-1415. IEEE, 2001.
- [17]. S. Li, J. Xu, M. V. D. Schaar, and W. Li. "Popularity-driven content caching." In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pp. 1-9. IEEE, 2016.
- [18]. M. Khouja. "A joint optimal pricing, rebate value, and lot sizing model." *European Journal of Operational Research* 174, no. 2 (2006): 706-723.
- [19]. H. L. Cadre, M. Bouhtou, and B. Tuffin. "A pricing model for a mobile network operator sharing limited resource with a mobile virtual network operator." In *International Workshop on Internet Charging and QoS Technologies*, pp. 24-35. Springer, Berlin, Heidelberg, 2009.
- [20]. F. J. Arcelus, S. Kumar, and G. Srinivasan. "Pricing and rebate policies in the two-echelon supply chain with asymmetric information under price-dependent, stochastic demand." *International Journal of Production Economics* 113, no. 2 (2008): 598-618.
- [21]. Q. Liu, R. Safavi-Naini, and N. P. Sheppard. "Digital rights management for content distribution," In *Proceedings of the Australasian information security workshop conference on ACSW frontiers*, 2003, pp. 49-58.
- [22]. C. Y. Chuang, Y. C. Wang, and Y. B. Lin. "Digital right management and software protection on Android phones." In *Vehicular Technology Conference (VTC 2010-Spring)*, 2010 IEEE 71st, pp. 1-5. IEEE, 2010.
- [23]. M. Taghizadeh, and S. Biswas. "Impacts of user-selfishness on cooperative content caching in social wireless networks." *Ad Hoc Networks* 11, no. 8 (2013): 2423-2439.
- [24]. M. Taghizadeh., K. Micinski, C. Ofria, E. Torng and S. Biswas, "Distributed cooperative caching in social wireless networks," *Mobile Computing, IEEE Transactions on* 12.6, 2013, pp. 1037-1053.
- [25]. L. Breslau, P. Cao, L. Fan, G. Phillips, and Scott Shenker. "Web caching and Zipf-like distributions: Evidence and implications," In *INFOCOM*, 1999, pp. 126-134.

- [26]. H. Ahlehagh, and S. Dey. "Video caching in radio access network: impact on delay and capacity." *Wireless Communications and Networking Conference (WCNC), 2012 IEEE.* IEEE, 2012
- [27]. M. Taghizadeh, and S. Biswas, "Minimizing content provisioning cost in heterogeneous social wireless networks," *Communication Systems and Networks (COMSNETS), 2011 Third International Conference on, IEEE, 2011, pp. 1-10.*
- [28]. J. Wang. "A survey of web caching schemes for the internet." *ACM SIGCOMM Computer Communication Review* 29, no. 5 (1999): 36-46.
- [29]. W. Ali, S. M. Shamsuddin, and A. S. Ismail. "A survey of web caching and prefetching." *Int. J. Advance. Soft Comput. Appl* 3, no. 1 (2011): 18-44.
- [30]. I. Abdullahi, S. Arif, and S. Hassan. "Survey on caching approaches in information centric networking." *Journal of Network and Computer Applications* 56 (2015): 48-59.
- [31]. G. Zhang, Y. Li, and T. Lin. "Caching in information centric networking: A survey." *Computer Networks* 57, no. 16 (2013): 3128-3141.
- [32]. M. Zhang, H. Luo, and H. Zhang. "A survey of caching mechanisms in information-centric networking." *IEEE Communications Surveys & Tutorials* 17, no. 3 (2015): 1473-1499.
- [33]. Y. Liu, et al. "Dynamic adaptive streaming over CCN: a caching and overhead analysis." *Communications (ICC), 2013 IEEE International Conference on. IEEE, 2013.*
- [34]. C. Bernardini, T. Silverston, and O. Festor. "MPC: Popularity-based caching strategy for content centric networks." In *2013 IEEE international conference on communications (ICC)*, pp. 3619-3623. IEEE, 2013.
- [35]. K. Liang, and H. F. Yu. "Adjustable two-tier cache for IPTV based on segmented streaming." *International Journal of Digital Multimedia Broadcasting* 2012 (2012).
- [36]. K. Cho, M. Lee, K. Park, T. T. Kwon, Y. Choi, and S. Pack. "Wave: Popularity-based and collaborative in-network caching for content-oriented networks." In *2012 Proceedings IEEE INFOCOM Workshops*, pp. 316-321. IEEE, 2012.
- [37]. K. S. Candan, W. Li, Q. Luo, W. Hsiung, and D. Agrawal. "Enabling dynamic content caching for database-driven web sites." In *ACM SIGMOD Record*, vol. 30, no. 2, pp. 532-543. ACM, 2001.
- [38]. G. Ma, Z. Wang, M. Zhang, J. Ye, M. Chen, and W. Zhu. "Understanding performance of edge content caching for mobile video streaming." *IEEE Journal on Selected Areas in Communications* 35, no. 5 (2017): 1076-1089.
- [39]. P. T. Joy, and K. P. Jacob. "A Comparative Study of Cache Replacement Policies in Wireless Mobile Networks." *Advances in Computing and Information Technology.* Springer Berlin Heidelberg, 2012. 609-619.

- [40]. E. Bastug, et al. "Centrality-based caching for mobile wireless networks." In 1st KuVS Workshop on Anticipatory Networks. 2014.
- [41]. J. Dai, et al. "Collaborative caching in wireless video streaming through resource auctions." *Selected Areas in Communications, IEEE Journal on* 30.2 (2012): 458-466.
- [42]. K. Kanai, T. Muto, H. Kisara, J. Katto, T. Tsuda, W. Kameyama, Y. Park, and T. Sato. "Proactive content caching utilizing transportation systems and its evaluation by field experiment." In 2014 IEEE Global Communications Conference, pp. 1382-1387. IEEE, 2014.
- [43]. K. Kanai, T. Muto, J. Katto, S. Yamamura, T. Furutono, T. Saito, H. Mikami et al. "Proactive content caching for mobile video utilizing transportation systems and evaluation through field experiments." *IEEE Journal on Selected Areas in Communications* 34, no. 8 (2016): 2102-2114.
- [44]. J. Qiao, Y. He, and X. S. Shen. "Proactive Caching for Mobile Video Streaming in Millimeter Wave 5G Networks." *IEEE Trans. Wireless Communications* 15, no. 10 (2016): 7187-7198.
- [45]. R. Ding, T. Wang, L. Song, Z. Han, and J. Wu. "Roadside-unit caching in vehicular ad hoc networks for efficient popular content delivery." In *Wireless Communications and Networking Conference (WCNC), 2015 IEEE*, pp. 1207-1212. IEEE, 2015.
- [46]. G. Mauri, M. Gerla, F. Bruno, M. Cesana, and G. Verticale. "Optimal Content Prefetching in NDN Vehicle-to-Infrastructure Scenario." *IEEE Transactions on Vehicular Technology* 66, no. 3 (2017): 2513-2525.
- [47]. M. Amadeo, C. Campolo, and A. Molinaro. "Information-centric networking for connected vehicles: a survey and future perspectives." *IEEE Communications Magazine* 54, no. 2 (2016): 98-104.
- [48]. Z. Al-Arnaout, Q. Fu, and M. Frenn. "On the local popularity impact on object replica placement over WMNs." *WoWMoM*, 2014.
- [49]. S. Aggarwal, J. Kuri, and C. Saha. "Give-and-take based peer-to-peer content distribution networks." *Sadhana* 39.4 (2014): 843-858.
- [50]. H. Narimatsu, H. Kasai, and R.i Shinkuma. "Area-based collaborative distributed cache system using consumer electronics mobile device." *Consumer Electronics*, 57.2 (2011): 564-573.
- [51]. J. Ahn, et al., "Optimizing content dissemination in vehicular networks with radio heterogeneity." *Mobile Computing, IEEE Transactions on* 13, no. 6 (2014): 1312-1325.
- [52]. L. Hu. "Mobile Peer-to-Peer Data Dissemination over Opportunistic Wireless Networks." PhD diss., Technical University of Denmark, 2009.

- [53]. N. Dimokas, D. Katsaros, L. Tassiulas, and Y. Manolopoulos. "High performance, low complexity cooperative caching for wireless sensor networks." *Wireless Networks* 17, no. 3 (2011): 717-737.
- [54]. W. Gao, G. Cao, and M. Srivatsa. "Cooperative caching for efficient data access in disruption tolerant networks." *Mobile Computing, IEEE Transactions on* 13, no. 3 (2014): 611-625.
- [55]. Z. Liu, et al. "Small World P2P overlay for video sharing service." *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*. IEEE, 2012.
- [56]. F. Sun, B. Liu, F. Hou, H. Zhou, J. Chen, Y. Rui, and L. Gui. "A qoe centric distributed caching approach for vehicular video streaming in cellular networks." *Wireless Communications and Mobile Computing* 16, no. 12 (2016): 1612-1624.
- [57]. N. Kumar, S. Zeadally, and J. J. Rodrigues. "QoS-aware hierarchical web caching scheme for online video streaming applications in internet-based vehicular ad hoc networks." *IEEE Transactions on Industrial Electronics* 62, no. 12 (2015): 7892-7900.
- [58]. S. Hatakeyama, Y. Sakata, and H. Shigeno. "Cooperative Mobile Live Streaming Considering Neighbor Reception." *Advanced Information Networking and Applications (AINA), 2014 IEEE 28th International Conference on*. IEEE, 2014.
- [59]. J. Bruneau-Queyreix, M. Lacaud, and D. Négru. "A Hybrid P2P/Multi-Server Quality-Adaptive Live-Streaming Solution Enhancing End-User's QoE." In *Proceedings of the 25th ACM international conference on Multimedia*, pp. 1261-1262. ACM, 2017.
- [60]. L. Liu, D. Xie, S. Wang, and Z. Zhang. "CCN-based cooperative caching in VANET." In *Connected Vehicles and Expo (ICCVE), 2015 International Conference on*, pp. 198-203. IEEE, 2015.
- [61]. N. Kumar, and J. Lee, 2014. "Peer-to-peer cooperative caching for data dissemination in urban vehicular communications". *IEEE Systems Journal*, 2014, 8(4), pp.1136-1144.
- [62]. Z. Hu, Z. Zheng, T. Wang, L. Song, and X. Li. "Game theoretic approaches for wireless proactive caching." *IEEE Communications Magazine* 54, no. 8 (2016): 37-43.
- [63]. H. Tian, M. Mohri, Y. Otsuka, Y. Shiraishi, and M. Morii. "Lce in-network caching on vehicular networks for content distribution in urban environments." In *Ubiquitous and Future Networks (ICUFN), 2015 Seventh International Conference on*, pp. 551-556. IEEE, 2015.
- [64]. W. Zhao, Y. Qin, D. Gao, C. H. Foh, and H. Chao. "An efficient cache strategy in information centric networking vehicle-to-vehicle scenario." *IEEE Access* 5 (2017): 12657-12667.
- [65]. K. Zhang, S. Leng, Y. He, S. Maharjan, and Y. Zhang. "Cooperative Content Caching in 5G Networks with Mobile Edge Computing." *IEEE Wireless Communications* 25, no. 3 (2018).

- [66]. A. Keränen, O. Jörg, and T. Kärkkäinen. "The ONE simulator for DTN protocol evaluation." In Proceedings of the 2nd international conference on simulation tools and techniques, p. 55. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009.
- [67]. J. Summers, T. Brecht, D. Eager, T. Szepesi, B. Cassell, and B. Wong. "Automated control of aggressive prefetching for HTTP streaming video servers." In Proceedings of International Conference on Systems and Storage, pp. 1-11. ACM, 2014.
- [68]. E. Delen, J. Liew, and V. Willson. "Effects of interactivity and instructional scaffolding on learning: Self-regulation in online video-based environments." *Computers & Education* 78 (2014): 312-320.
- [69]. I. Sodagar. "The mpeg-dash standard for multimedia streaming over the internet." *IEEE MultiMedia* 4 (2011): 62-67.
- [70]. S. Eetha, S. Agarwal, and S. Neelam. "Zynq FPGA Based System Design for Video Surveillance with Sobel Edge Detection." In 2018 IEEE International Symposium on Smart Electronic Systems (iSES)(Formerly iNiS), pp. 76-79. IEEE, 2018.
- [71]. "Screen Resolution Statistics (January 2017)," [online]. Available: https://www.w3schools.com/browsers/browsers_display.asp
- [72]. R. Pantos, and W. May. HTTP live streaming. No. RFC 8216. 2017.
- [73]. "Trace set of mobility data of taxi cabs in San Francisco, USA." [online]. Available: <https://crawdad.org/epfl/mobility/20090224/cab/>
- [74]. H. Shimada, A. Yamaguchi, H. Takada, and K. Sato. "Implementation and evaluation of local dynamic map in safety driving systems." *Journal of Transportation Technologies* 5, no. 02 (2015): 102.
- [75]. T. Abdelkader, K. Naik, A. Nayak, N. Goel, and V. Srivastava. "A performance comparison of delay-tolerant network routing protocols." *IEEE Network* 30, no. 2 (2016): 46-53.
- [76]. T. Choksatid, W. Narongkhachavana, and S. Prabhavat. "An efficient spreading epidemic routing for delay-tolerant network." In 2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC), pp. 473-476. IEEE, 2016.
- [77]. D. Hales. "Distributed computer systems." In *Simulating Social Complexity*, pp. 563-580. Springer, Berlin, Heidelberg, 2013.