

DATA REPLICATION AND SCHEDULING  
FOR CONTENT AVAILABILITY  
IN VEHICULAR NETWORKS

by

Shyam N Kapadia

---

A Dissertation Presented to the  
FACULTY OF THE GRADUATE SCHOOL  
UNIVERSITY OF SOUTHERN CALIFORNIA  
In Partial Fulfillment of the  
Requirements for the Degree  
DOCTOR OF PHILOSOPHY  
(COMPUTER SCIENCE)

May 2007

Copyright 2007

Shyam N Kapadia

## **Dedication**

This dissertation is dedicated to my mother, father, brother, and grand-mother.

## Acknowledgements

First and foremost, I am especially thankful to Bhaskar for the countless nights that we spent together working on some part or another of this thesis. Over the years, he has been a tremendous source of inspiration and strengthened my belief that with hard work and dedication any goal can be achieved. His own work ethic and discipline have inspired me in more ways than one. I would also like to thank my co-advisor, Shahram; we have collaborated for over 5 years now. I have learnt a lot under his guidance as well. Bhaskar pushed me more toward complimenting my research with rigorous mathematical analysis and adopted a more “hands-on” to “hands-off” approach to advising as I moved toward the final years of my PhD. Shahram on the other hand was more hands-on in his advising concentrating on the minutest details in terms of paper submissions, presentations etc. I would also like to thank Dr. Helmy for exposing me to the research in wireless ad-hoc networks, the initial collaboration with him is what motivated me to process toward a PhD. My committee members, namely, Dr. Psounis, and Dr. Zimmerman, also provided valuable advice which helped in improving this thesis.

I would also like to thank Dr. Gully Burns, a research assistant professor in the Neurobiology department at USC, currently at ISI, for not only financially supporting me during my Masters and earlier part of my PhD but also the training and experience I received while collaborating with him was invaluable. I learnt the art of writing high quality code for release to the outside world. He gave me a lot of flexibility in terms of my working hours and gave me challenging projects to work on. I have also had a lot of philosophical discussions with him about life and my progress toward the PhD for which I would be eternally grateful. Both on a personal and professional front I found his attitude extremely warm and helpful.

I would also like to express my gratitude for the members of the ANRG group and other close friends and researchers whose work either implicitly or explicitly influenced my work. Specifically, our alumni: Nara, Gang, and Yang have been extremely helpful in giving me kind advice about how to go about my job search. Marco, who will be graduating along with me, has been doing really good work both in his research and also for the group, he definitely has been an excellent student role-model for others in the group. Numerous interesting conversations both on the research and non-research front have provided valuable take-away lessons. Have also been involved in a variety of discussions with Kiran, especially toward the end of my PhD, he has been extremely helpful in more ways than one.

Continuing on, Avinash, the code-guru of the ANRG group, has imparted and trained a lot of personnel in our group especially in tiny OS related programming stuff. Sundeep contributed equally for the same, we spent a lot of night outs together in the laboratory working on a collaborative group project, that time was by far the most fun time I have had during my PhD. I had a good time with the interns, Maulik and Xiaofan, that joined our group for the summer of 2006, the experience provided an opportunity to adopt the role of a mentor. Also special thanks to other members of the group: Hua, Joon, Yi, Pai-Han, Peter, and Amitabh.

Among other friends, would like to start by thanking Karthik, we have shared numerous courses together since the beginning of my tenure at USC. Hence, have had a lot of study sessions as well as usual discussions on the professional as well as personal fronts over the past few years. Then come the two Ramakrishnas, one called Ram and the other Ramki, the former has been a great help in my job search, the latter in giving lots of advice on a lot of topics related to research and career development.

I would like to mention some other friends who I have known for a long time now, Mithu, Puneet, Vidula, Rohyt, Kiran, and Sumit, anytime I needed a break from my PhD, I always flew down to the east coast to meet them. And every single time they made me feel like a VIP, most of these visits were stress relievers because of which I could come back and restart my research with renewed enthusiasm.

## Table Of Contents

<b>Dedication</b>	<b>ii</b>
<b>Acknowledgements</b>	<b>iii</b>
<b>List Of Tables</b>	<b>viii</b>
<b>List Of Figures</b>	<b>ix</b>
<b>Abstract</b>	<b>xiii</b>
<b>Chapter 1: Introduction</b>	<b>1</b>
1.1 Overview of Case Studies . . . . .	4
1.1.1 PAVAN . . . . .	4
1.1.2 Static Replication schemes . . . . .	5
1.1.3 Zebroids . . . . .	5
1.2 Organization of this thesis . . . . .	6
<b>Chapter 2: Common Assumptions and Architectural Framework</b>	<b>7</b>
2.1 Assumptions . . . . .	7
2.2 Preliminaries . . . . .	8
2.3 Architectural Framework . . . . .	9
2.4 Simulation Model . . . . .	11
<b>Chapter 3: PAVAN</b>	<b>14</b>
3.1 PAVAN variants . . . . .	14
3.1.1 Spatio-Temporal Lookahead (STL) parameter . . . . .	17
3.2 Simulation Study . . . . .	19
3.2.1 Experimental Set-up . . . . .	19
3.2.2 Utility Models . . . . .	21
3.2.3 Results . . . . .	21
3.3 Summary . . . . .	25
<b>Chapter 4: Static Replication Schemes</b>	<b>27</b>
4.1 Family of Replication Policies . . . . .	27
4.2 Data items with display time one and long client trip duration . . . . .	29
4.2.1 Analysis . . . . .	29

4.2.1.1	Sparse Scenario . . . . .	30
4.2.1.2	Dense scenario . . . . .	34
4.2.2	Simulation results . . . . .	37
4.2.2.1	Scale-up experiments . . . . .	38
4.2.2.2	Variation in car density . . . . .	39
4.2.2.3	Variation in storage per car . . . . .	42
4.2.2.4	Variation in data item repository size . . . . .	42
4.3	Data items with display time one and short trip duration . . . . .	42
4.3.1	Analysis . . . . .	45
4.3.1.1	Sparse Approximation . . . . .	45
4.3.1.2	Dense Approximation . . . . .	46
4.3.2	Simulation Results . . . . .	46
4.4	Data items with higher display time and long client trip duration . . . . .	48
4.5	Data items with higher display time and short client trip duration . . . . .	50
4.5.1	Aggregate availability latency as a function of car density ( $N$ ) . . . . .	53
4.6	Evaluation with a real map . . . . .	54
4.6.1	Results with replication schemes . . . . .	56
4.7	Evaluation with real movement traces . . . . .	57
4.7.1	UMassDieselNet Traces . . . . .	59
4.7.2	Experimental Set-up . . . . .	61
4.7.3	Results . . . . .	61
4.8	Summary . . . . .	64
<b>Chapter 5: Zebroids</b>		<b>67</b>
5.1	Overview of Zebroids . . . . .	68
5.2	Solution Approach . . . . .	69
5.3	Carrier-based Replacement policies . . . . .	71
5.4	Analysis Methodology . . . . .	73
5.4.1	One-instantaneous zebroids . . . . .	74
5.4.2	z-relay zebroids . . . . .	75
5.5	Simulation Methodology . . . . .	76
5.6	Results . . . . .	79
5.6.1	Zebroid replacement schemes . . . . .	79
5.6.2	Zebroids performance improvement . . . . .	81
5.6.2.1	Analysis . . . . .	81
5.6.2.2	Simulation . . . . .	82
5.6.3	Zebroid overhead . . . . .	84
5.6.3.1	Analysis . . . . .	84
5.6.3.2	Simulation . . . . .	86
5.6.4	Zebroids with inaccurate route predictability . . . . .	88
5.6.4.1	Analysis . . . . .	88
5.6.4.2	Simulation . . . . .	89
5.6.5	Maximum improvement with zebroids . . . . .	89
5.6.5.1	Analysis . . . . .	89
5.6.5.2	Simulation . . . . .	90

5.6.6	Zebroid trade-offs with car density and storage per car . . . . .	91
5.6.6.1	Analysis . . . . .	91
5.6.6.2	Simulation . . . . .	92
5.6.7	Impact of different trip durations and repository sizes . . . . .	92
5.6.7.1	Analysis . . . . .	93
5.6.7.2	Simulation . . . . .	95
5.7	Evaluation with a real map . . . . .	96
5.7.1	Results with zebroids . . . . .	96
5.8	Evaluation with Real Traces . . . . .	99
5.8.1	Experimental Set-up . . . . .	101
5.8.2	Results . . . . .	101
5.8.2.1	Requests issued at the start of the day . . . . .	101
5.8.2.2	Requests issued at equal inter-arrival times during the day	102
5.9	Summary . . . . .	105
<b>Chapter 6: Related Work</b>		<b>109</b>
6.1	Other Components of an AutoMata Application . . . . .	109
6.2	Replication in MANETs . . . . .	110
6.3	Sparse Network Architectures . . . . .	112
6.4	Intelligent Transportation Systems (ITS) . . . . .	115
<b>Chapter 7: Conclusions</b>		<b>116</b>
7.1	PAVAN . . . . .	116
7.2	Static Replication Schemes . . . . .	117
7.3	Zebroids . . . . .	118
<b>Bibliography</b>		<b>119</b>

## List Of Tables

2.1	Terms and their definitions . . . . .	8
3.1	Four variants of PAVAN. . . . .	16
3.2	Three utility models to evaluate alternative variants of PAVAN. . . . .	20
4.1	Approximate optimal replication exponents for data items with higher data item display times. . . . .	50
6.1	Related studies on intermittently connected networks. . . . .	113



## List Of Figures

1.1	An example illustration of an AutoMata application. . . . .	2
1.2	Components of an AutoMata application . . . . .	3
2.1	A hierarchical architecture. . . . .	10
2.2	An example $6 \times 6$ map. . . . .	11
3.1	An overview of PAVAN, its inputs and output. . . . .	15
3.2	The numbers in the cells indicate STL value for the shaded cell numbered 1. . . . .	18
3.3	A comparison of PAVAN with different inputs for utility models 1 and 2 as a function of the degree of replication of the titles. . . . .	20
3.4	Difference in the availability latencies as a function of the degree of replication of the titles for different title display times. . . . .	23
3.5	Comparison of difference in the availability latencies of $SS_{only}$ and $PL_t$ for different AutoMata densities in a $10 \times 10$ map as a function of the degree of title replication. The title display time under consideration spans 5 cells. . . . .	24
4.1	Sparse analysis ( Equation 4.6) versus simulation obtained average availability latency for a data item as a function of its replicas for a $10 \times 10$ torus, when the number of cars is set to 100. . . . .	34
4.2	Figure 4.2(a) shows the validation of the analytical expression in Equation 4.25 for the probability that availability latency is zero. Figure 4.2(b) shows the probability that the availability latency is zero as a function of the replicas for the data item for 5 different car densities $\{50, 100, 150, 200, 250\}$ . . . . .	35
4.3	The complete picture depicting the availability latency for a data item obtained via simulations as compared with its sparse and dense approximation as a function of its replicas for a $10 \times 10$ torus, when the number of cars is set to 100. . . . .	36
4.4	Aggregate availability latency for different replication strategies for a $10 \times 10$ torus when $T = 100$ and $N = 50$ . Figures (a), (b), and (c) depict three different storage values per car: $\{4, 10, 25\}$ . . . . .	37

4.5	Scale-up experiments where the total storage to the data item repository size is held constant at $\frac{S_T}{T} = \frac{3000}{600}$ . The number of cars and the storage per car are varied to realize $S_T = 3000$ . . . . .	40
4.6	Aggregate availability latency for the three replication schemes as a function of the car density when the storage per car is fixed at 3. Here $T = 100$ . . . . .	41
4.7	Aggregate availability latency for the three replication schemes as a function of the storage per car when data item repository size is 50 and car density is 50. . . . .	43
4.8	Aggregate availability latency for the three replication schemes as a function of the data item repository size for a car density of 50 and storage per car of 15. . . . .	44
4.9	Average availability latency for a data item as a function of its replicas for a finite trip duration $\gamma$ of 10. The simulation curves are plotted along with the sparse and dense approximations for finite trip duration for a $10 \times 10$ torus, when the number of cars is set to 50. . . . .	47
4.10	Aggregate availability latency for different replication strategies for a $10 \times 10$ torus for a finite trip duration of 10 when $T = 100$ and $N = 50$ . Figures (a), (b), and (c) depict three different storage values per car: $\{4, 10, 25\}$ . . . . .	48
4.11	Average availability latency for a data item as a function of its replicas for different data item display times for a $10 \times 10$ torus. The latency is given by $\frac{C}{r_i^\sigma}$ where the exponent $\sigma$ increases with data item display time. . . . .	49
4.12	Figure 4.12(a) shows $\delta_{agg}$ of the sqrt, linear and random replication schemes versus $\alpha$ for $\Delta = 4$ and $N = 200$ . Figure 4.12(b) shows the % comparison of the linear and random schemes wrt the sqrt scheme for this scenario. Region I and Region II, respectively, indicate the parameter space where $n = 1$ and $n = 0.5$ perform the best. . . . .	51
4.13	A map of the San Francisco Bay Area obtained from <a href="http://maps.google.com">http://maps.google.com</a> is shown in Figure 4.13(a). Figure 4.13(b) superimposes a $15 \times 15$ grid on this map and labels the cells appropriately with the freeway IDs that they overlap with. . . . .	55
4.14	The intersection between freeways 880 and 85 is captured in the figure along with the equivalent probability transitions in the Markov model based on data obtained from Caltrans regarding the vehicular densities. . . . .	56
4.15	Performance of various replication schemes as a function of car density when $T = 25$ , $\alpha = 2$ , and $\gamma = 10$ . Figure 4.15(b) shows the performance wrt the linear scheme. . . . .	57
4.16	Performance of various replication schemes as a function of storage per car when $T = 25$ , $N = 50$ , and $\gamma = 10$ . Figure 4.16(b) shows the performance wrt the linear scheme. . . . .	58

4.17	Performance of various replication schemes as a function of data item repository size when $N = 50$ , $\alpha = 2$ , and $\gamma = 10$ . Figure 4.17(b) shows the performance wrt the linear scheme. . . . .	58
4.18	The number of active buses for each trace representing the bus encounters for each day of a 60-day period. The buses operated from 7am to 5pm. . .	60
4.19	The CDF of the time between encounters averaged across all the traces for 2 different separation times 0s (Figure 4.19(a)) and 20s (Figure 4.19(b)). .	60
4.20	Aggregate availability latency for satisfied requests and the aggregate unsatisfied request metric for the random, square-root, and linear replication schemes are shown in Figure 4.20(a) and (b) respectively. The ratio of the storage per car to the data item repository size, $\frac{\alpha}{T}$ is maintained as 1:5. .	62
4.21	Aggregate availability latency for satisfied requests (Figure 4.21(a)) and the aggregate unsatisfied request metric (Figure 4.21(b)) as a function of the storage per car for a data item repository size of 25. . . . .	63
4.22	Aggregate availability latency for satisfied requests (Figure 4.22(a)) and the aggregate unsatisfied request metric (Figure 4.22(b)) as a function of the data item repository size when storage per car $\alpha$ is fixed at 3. . . . .	63
4.23	CDF of the time between encounters from the Markov model for a $25 \times 25$ torus with a car density of 15. . . . .	65
4.24	Aggregate availability latency for satisfied requests and the aggregate unsatisfied request metric as obtained from an equivalent scenario employing the Markov model. The ratio of the storage per car to the data item repository size, $\frac{\alpha}{T}$ is maintained as 1:5. . . . .	65
5.1	Availability latency when employing one-instantaneous zebroids as a function of $(N, \alpha)$ values, when the total storage in the system is kept fixed, $S_T = 200$ . . . . .	80
5.2	Latency performance with one-instantaneous zebroids via simulations along with the analytical approximation for a $10 \times 10$ torus with $T = 10$ . . . .	81
5.3	Latency performance with z-relay zebroids via analysis and simulations for a $10 \times 10$ torus with $T = 10$ . . . . .	83
5.4	Latency performance with both one-instantaneous and z-relay zebroids as a function of the car density when $\alpha = 2$ and $T = 25$ . . . . .	84
5.5	Latency performance with both one-instantaneous and z-relay zebroids as a function of $\alpha$ when $N = 50$ and $T = 25$ . . . . .	85
5.6	Replacement overhead when employing one-instantaneous zebroids as a function of $(N, \alpha)$ values, when the total storage in the system is kept fixed, $S_T = 200$ . . . . .	86

5.7	Replacement overhead with zebroids for the cases when $N$ is varied keeping $\alpha = 2$ (figure 5.7.a) and $\alpha$ is varied keeping $N = 50$ (figure 5.7.b). . . . .	87
5.8	Availability latency, $\delta_{agg}$ , for different car densities as a function of the prediction accuracy metric with $\alpha = 2$ and $T = 25$ . . . . .	88
5.9	Improvement in availability latency with one-instantaneous zebroids as a function of $(N, \alpha)$ values, when the total storage in the system is kept fixed, $S_T = 200$ . . . . .	91
5.10	Improvement in $\delta_{agg}$ with one-instantaneous zebroids for different client trip durations in case of $10 \times 10$ torus with a fixed car density, $N = 100$ . . . . .	93
5.11	Shows improvement in availability latency as a function of the car density for different repository sizes with $\alpha = 2$ and $\gamma = 10$ . . . . .	94
5.12	Performance with zebroids as a function of different $(N, \alpha)$ values when the total storage in the system is held constant at $S_T = 200$ , $\gamma = 10$ . Figure (b) shows the percentage improvement with zebroids when compared to the no-zebroids case. . . . .	97
5.13	Performance with zebroids as a function of car density when $T = 25$ , $\alpha = 2$ , and $\gamma = 10$ . Figure (b) shows the percentage improvement with zebroids when compared to the no-zebroids case. . . . .	97
5.14	Performance with zebroids as a function of storage per car when $T = 25$ , $N = 50$ , and $\gamma = 10$ . Figure (b) shows the percentage improvement with zebroids when compared to the no-zebroids case. . . . .	98
5.15	Performance with zebroids as a function of data item repository size when $N = 50$ , $\alpha = 2$ , and $\gamma = 10$ . Figure (b) shows the percentage improvement with zebroids when compared to the no-zebroids case. . . . .	99
5.16	Aggregate Availability Latency for Linear, Sqrt, and Random Replication Schemes for Zipf values -0.5, -1.0, -1.5, and -2.0 . . . . .	100
5.17	Aggregate availability latency and normalized unsatisfied requests with zebroids for the case when the ratio of $T : \alpha$ is maintained as 5 : 1 and requests are issued as per a Zipf distribution at equal inter-arrival times. . . . .	104
5.18	Replacement overhead incurred by employing zebroids when the ratio of $T : \alpha$ is maintained as 5 : 1 and requests are issued as per a Zipf distribution at equal inter-arrival times. . . . .	104
5.19	Performance with zebroids as a function of the storage per car when the data item repository size is held constant at 10. Requests are issued as per a Zipf distribution at equal inter-arrival times. . . . .	106
5.20	Performance with zebroids as a function of the data item repository size when the storage per car is held constant at 3. Again, requests are issued as per a Zipf distribution at equal inter-arrival times. . . . .	107

## Abstract

On-demand delivery of audio and video clips in a vehicular network is a growing area of interest. A given repository of such data items, each with an associated popularity, may be available to the passengers of the vehicles. The vehicles themselves are equipped with a ‘TiVO’ like device that has several gigabytes of storage and a wireless interface allowing short range communication at 10s to 100s of Megabits per second. The goal is to minimize the latency between request issuance and the time till a copy of the requested item is encountered. This latency is termed the availability latency. This thesis explores two generic tools to alleviate availability latency: (a) data replication (b) data delivery scheduling.

With the replication study, we propose a general optimization formulation that seeks to minimize average availability latency subject to a storage constraint per vehicle. We explore the effects of a family of popularity-based replication schemes on availability latency. When the vehicles follow a 2D random walk based mobility model, via analysis and extensive simulations, we determine the optimal replication scheme that minimizes latency across a wide parameter space with major dimensions being data item size and client trip duration .

Once an appropriate static replication scheme has allocated replicas, the vehicles themselves may be employed as data carriers to further improve availability latency. These data carriers are termed zebroids. However, a zebroid’s local storage may be completely exhausted. Hence, to accommodate this new data item, it may need to evict an existing one. Various replacement policies such as LFU, LRU, random etc. are examined and their relative performance is studied. Via analysis and extensive simulations we study the behavior of zebroids as a function of large parameter space comprising data

item repository size, storage per vehicle, number of vehicles, popularity distribution of the data items, different replacement schemes for zebroids etc.

We validate the Markov model based observations with two independent validation phases employing (a) freeway traffic information on a city map (b) real world traces from a small bus network.

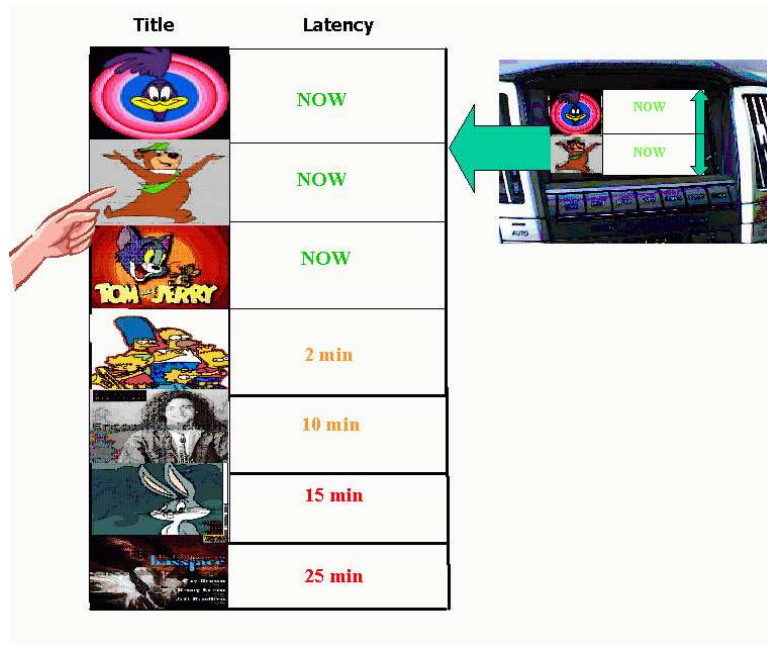
# Chapter 1

## Introduction

The notion of ‘entertainment on wheels’ is no longer a distant dream. With today’s technology, it is possible to present entertainment content in the form of audio and video clips to passengers as they travel in their vehicles in a city. Advances in technology, both in the area of storage and wireless communications, have contributed to the support of on-demand delivery of such content among mobile vehicles. Vehicles may be equipped with devices consisting of several gigabytes of storage, a fast processor, and a wireless interface with bandwidths of several 10s or 100s of Megabits per second. These devices are termed AutoMata [5] (formerly known as a C2P2 [21] for Car-to-Car-Peer-to-Peer) and they collaborate to form a mobile ad-hoc network to deliver the requested data to a client. The radio range of these devices is in the order of a few hundred feet.

The content exchanged between the vehicles may vary from traffic information such as accident notifications and emergency vehicle arrival notifications to multimedia for entertainment such as audio files, cartoons, movies and other video files. Without loss of generality, throughout this thesis, we will use the term data item from now on with the understanding that a data item can be an audio title, video title or any other useful content.

In a typical scenario, a client of this application operating over an AutoMata network is provided a list of available data items (see Figure 1.1). Several other components may be involved in realizing such an application. Figure 1.2 depicts a realization of a component diagram that may be used to realize such an application. We provide a very brief overview of the functionality of some of the components.










Title	Latency
	NOW
	NOW
	NOW
	2 min
	10 min
	15 min
	25 min

Figure 1.1: An example illustration of an AutoMata application.

Once a user initiates display of a data item, an admission control component [18] ensures availability of both resources and the referenced data. Next, a data delivery scheduling technique [20] utilizes resources as a function of time to deliver the data item to a requesting AutoMata device. This component, may switch between several candidate servers containing the referenced data item based on their proximity, current availability of resources, and network conditions. This component is tied closely to an ad-hoc network routing protocol which facilitates delivery of data between AutoMata devices. Example protocols are DSR [34], DSDV [47], AODV [48] to name a few. Another system component may monitor whether the system is providing a target AutoMata with the desired QoS and make adjustments as necessary. Besides the above components, there may be others responsible for addressing the security [64] and privacy [16] concerns of the user that may be mandatory for practical use of the system. Additionally, suitable physical and MAC layer optimizations may be needed to adapt to the wireless nature of the communication medium between the vehicles.



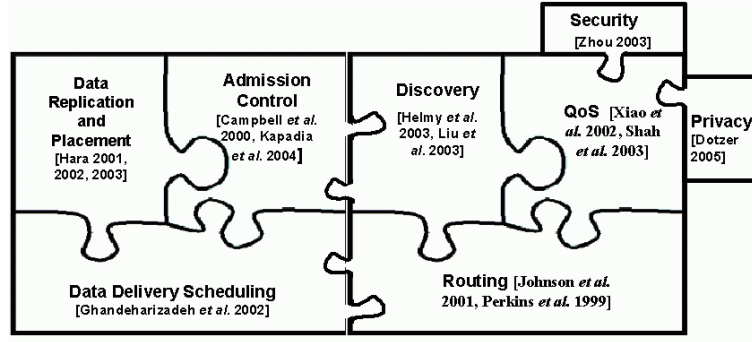


Figure 1.2: Components of an AutoMata application

While each of these components may warrant a separate thesis in itself, this thesis specifically explores possible realizations of three components, namely: Discovery (PAVAN), Data replication (Static Replication Schemes), Data delivery scheduling (Zebroids). Along the process, we explore various trade-offs that have influenced the design decisions for our proposed solution. Before we provide a brief overview of each of the three studies mentioned above, we briefly introduce the prime metric of interest in our study namely availability latency.

The time interval between when a request is issued by a client and a copy of the requested data item is encountered by it is referred to as availability latency (denoted by  $\delta$ ). A data item is available immediately when it resides in the local storage of the AutoMata device serving the request yielding  $\delta$  as zero. Users prefer lower availability latency associated with data items. Studies [39] with peer-to-peer network systems have indicated that users prefer a larger list of data items (files) each associated with a low latency. This latency is a function of a number of parameters: (i) current location of the client (ii) destination and travel path of the client (iii) mobility model of the AutoMata equipped cars (iv) number of replicas constructed for the different data items (v) placement of data item replicas across the AutoMata equipped cars. Without loss of generality and to simplify the discussion, we assume the term car refers to an AutoMata-equipped vehicle.

The biggest challenge in such an environment is mobility of the vehicles. This causes the network topology to change dynamically, hence traditional solutions proposed for static environments may not be directly applicable. Moreover, connectivity of the network at all times cannot be assumed as frequent partitioning may occur due to insufficient density of vehicles in certain parts of the network. However, knowledge of the mobility model dictating the vehicular movements may enable design of suitable schemes that allow an application to provide better estimates of the availability latency for the data items. In this thesis, we propose to incorporate knowledge of mobility into the design of various mechanisms that help enhance the availability latency to user desired content.

We first present PAVAN, a policy framework that, for a given data item level of replication, describes the procedure that outputs the list of available titles and their associated latency for users. One important finding of this study is that the degree of replication of the data items is the key parameter that influences availability latency. We then study the replication parameter in detail by proposing a family of static replication techniques and explore their effect on availability latency under different parameter settings. Having identified the performance improvements with static replication, we then study the effect of data carriers, termed zebroids, in providing further improvements in availability latency.

## 1.1 Overview of Case Studies

In this section, we summarize the findings of each of the three studies that are part of this thesis proposal.

### 1.1.1 PAVAN

In this study, we present PAVAN, a Policy for Availability in Vehicular Ad-hoc Networks. This policy framework outlines how the list of titles available to a client and their associated availability latency is computed. Here, we observe that when the degree of replication for the data items is below a certain threshold, the PAVAN variant that uses content density information and a predictive mobility model provides the best latency

performance. Identifying data replication as the key parameter that affects latency we next explore alternate replication strategies and their tradeoffs.

### 1.1.2 Static Replication schemes

In this study, we explore the effects of static replication schemes on availability latency. Given a data item repository, a certain vehicle density and a storage constraint per vehicle, we present an optimization formulation to determine the optimal number of data item replicas that minimize an average availability latency metric. We simplify the data placement issue by allocating the replicas to the vehicles uniformly at random, with the constraint that no two replicas of the same data item are placed in a vehicle. We analytically capture the variation in latency as a function of the data item replication levels when the vehicles obey a 2D random walk based Markov mobility model. We study the performance of a family of replication strategies in such an environment. Various design parameters are considered, such as size/display time of the data items, short/long client trip durations, and different data item repository sizes, and their effect on the optimal replication scheme is presented. This is followed by validation of the Markov model based observations with two independent validation phases employing (a) a real map of an urban environment that dictates the mobility transitions of the Markov model and (b) fine-grained mobility traces from a real environment comprising buses moving around a university campus area.

### 1.1.3 Zebroids

Once a static replication scheme has allocated replicas to the vehicles, zebroids can be used to further improve availability latency. A zebroid is a vehicle whose path rendezvous with both the client (data item requestor) and the server (vehicle containing that data item). Aided by this spatio-temporal overlap, zebroids can transport a data item from the server to the client. However, a zebroid's local storage may be completely exhausted. Hence, to accommodate this new data item, it may need to evict an existing one. Examples of replacement policies that determine what item to evict are LFU, LRU, random among others. The performance improvement in latency obtained with zebroids under conditions

of infinite storage and no interference is captured analytically. This is followed by an exhaustive simulation study that presents the behavior of zebroids as a function of large parameter space. As with static replication schemes, observations with the Markov model are validated by employing a Markov model derived from a real city map and also using the bus-based vehicular traces.

## **1.2 Organization of this thesis**

The rest of this thesis is organized as follows. Chapter 2 provides a description of the 2-tier architecture used in our study and introduces some common terminology and definitions used in this thesis. Chapter 3 describes PAVAN framework, results of the simulation study to evaluate the performance of the PAVAN variants. Chapter 4 introduces the family of frequency-based replication schemes that affect availability latency. presents the results of the experimental study that evaluates the performance of the various schemes under different parameter settings. Chapter 5 introduces zebroids as data carriers, describes the various environments used in this study and a classification of the different carrier-based replacement policies that are deployed in these environments, followed by detailed simulation results with zebroids deployed in the various environments with different policies. Chapter 6 gives a brief overview of the related work in the area. Chapter 7 concludes this document by highlighting the major contributions of this dissertation.

## Chapter 2

### Common Assumptions and Architectural Framework

In this chapter, we present the elements common to all the studies. In particular, they share a similar set of assumptions, a 2-tier architecture and the simulation model. Below we describe each in turn.

#### 2.1 Assumptions

- The list of data items comprising the database repository and their frequency of access is given and does not change. Moreover, the data items are not updated.
- There exists a 2-tier architecture comprising of (a) A high bandwidth data plane made up of the ad-hoc peer to peer network between the vehicles featuring bandwidths in the order of 10s to 100s of Mbps (b) A low bandwidth control plane similar to a cellular infrastructure between the vehicles and adjacent base stations which may be connected to the internet (see Section 2.3).
- Studies [49, 62, 52] that have explored the optimal number of neighbors, and optimal radio range to ensure connectivity in mobile wireless networks compliment our work. We assume that vehicles within radio range communicate directly and multi-hop transmissions are supported.
- We also assume the presence of suitable physical, MAC and routing layers and do not consider various low level wireless channel issues that have been studied

Database Parameters	
$T$	Number of data items.
$S_i$	Size of data item $i$
$\Delta_i$	Display time of data item $i$ .
$\beta_i$	Bandwidth requirement of data item $i$ .
$f_i$	Frequency of access to data item $i$ .
Replication Parameters	
$R_i$	Normalized frequency of access to data item $i$ , $R_i = \frac{(f_i)^n}{\sum_{j=1}^T (f_j)^n}$ ; $0 \leq n \leq \infty$
$r_i$	Number of replicas for data item $i$ , $r_i = \min(N, \max(1, \lfloor \frac{R_i \cdot N \cdot \alpha}{S_i} \rfloor))$
$n$	Characterizes a particular replication scheme.
$\delta_i$	Average availability latency of data item $i$
$\delta_{agg}$	Aggregate availability latency for replication technique using the $n^{th}$ power, $0 \leq n \leq \infty$ , $\delta_{agg} = \sum_{j=1}^T \bar{\delta}_j \cdot f_j$
AutoMata System Parameters	
$N$	Number of AutoMata devices in the system.
$\alpha$	Storage capacity per AutoMata.
$\gamma$	Trip duration of the client AutoMata.
$S_T$	Total storage capacity of the AutoMata system, $S_T = N \cdot \alpha$ .
$G$	Number of cells in the 2D torus.

Table 2.1: Terms and their definitions

extensively [14, 1]. Additionally, we also ignore wireless channel and contention issues.

## 2.2 Preliminaries

Here, we introduce some formalism in the notation used throughout this document. Table 2.1 summarizes the notation for the commonly used parameters. Assume a network of  $N$  mobile AutoMata devices, each with storage capacity of  $\alpha$  bytes. The total storage capacity of the system is  $S_T = N \cdot \alpha$ . There are  $T$  data items in the database, each with a display time of  $\Delta_i$  seconds and display bandwidth requirement of  $\beta_i$ . Hence the size of each data item is given by  $S_i = \Delta_i \cdot \beta_i$ . The frequency of access to data item  $i$  is denoted as  $f_i$  with  $\sum_{j=1}^T f_j = 1$ . Let the trip duration of the client AutoMata under consideration be  $\gamma$ . Let  $r_i$  represent the number of replicas for data item  $i$ .

The availability latency for a data item  $i$ , denoted as  $\delta_i$ , is defined as the time after which a client AutoMata will find at least one replica of the data item accessible to it, either directly or via multiple hops, for the data item display time ( $\Delta_i$ ). If this condition is not satisfied for a given request for data item  $i$ , then we set  $\delta_i$  to  $\gamma$  which indicates that data item  $i$  will not be available to the client during its journey. Also, if  $\Delta_i$  exceeds  $\gamma$  for a certain data item  $i$  then we set  $\delta_i$  to  $\gamma$ . Note  $\delta_i$  is the instantaneous availability latency for a given request for data item  $i$ .

We are interested in the average availability latency observed across all the data items. Hence, we weigh the average availability latency  $\bar{\delta}_i$  for every item  $i$  with its  $f_i$  yielding the aggregate availability latency ( $\delta_{agg}$ ) metric defined as follows:

$$\delta_{agg} = \sum_{i=1}^T \bar{\delta}_i \cdot f_i \quad (2.1)$$

The aggregate availability latency is the primary metric which we seek to optimize in our studies.

## 2.3 Architectural Framework

A vehicular ad-hoc network, such as AutoMata, may potentially cover a large geographical area, such as a metropolitan city. At such large distances, discovering available data items becomes a very challenging problem. It is easy to see that on-demand flooding/simple query-based approaches to resource discovery within the ad-hoc network will not scale well.

Our solution is to adopt a hierarchical architecture that also leverages the existing large scale heterogeneous wired-wireless cellular network infrastructure. This infrastructure aids in the collection of localized aggregate information that can be used to distribute the decision making. Our two-tiered architecture, shown in Figure 2.1, consists of separate data (edges labelled 3) and control networks (edges labelled 2). We now provide an overview of the various components of this architecture.

**Data network:** The data network consists of the vehicular ad-hoc network of AutoMata devices. The system storage is distributed among the various AutoMata devices

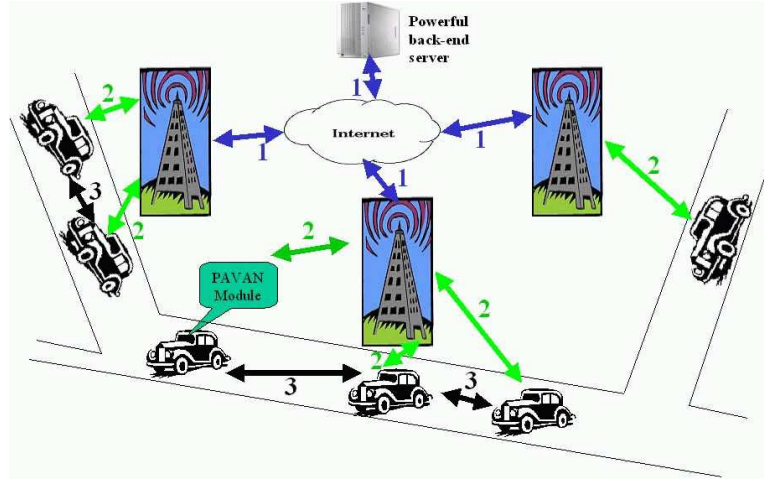


Figure 2.1: A hierarchical architecture.

within this network. At each instant, the communication is localized so that it is between nodes that are moving within the same cell. We assume that every AutoMata in the same cell is network connected. A typical path between two devices in the same cell may be multi-hop. This is because the range of a cellular base station is almost certainly much larger than the range of high bandwidth network devices (e.g., 802.11a [7]) employed by AutoMata devices. The number of hops is expected to be short, on the order of 3 to 4 hops.

**Control network:** The control network is a low data rate cellular network infrastructure, with base stations dividing a large geographical area into localized cells. It provides three key functionalities: (i) monitoring and collection of pertinent content and mobility information from individual car's AutoMata devices to the base station; (ii) regional consolidation and storage of this information into maps, mobility models and content information by nearby base stations and remote servers within the cellular network infrastructure; and (iii) periodic update of pertinent regional map, mobility, and content information of AutoMata devices within each cell. A base station may perform the last step by broadcasting information. Control messages are typically small and require a low data rate in the order of tens of Kilo bits per second (Kbps).



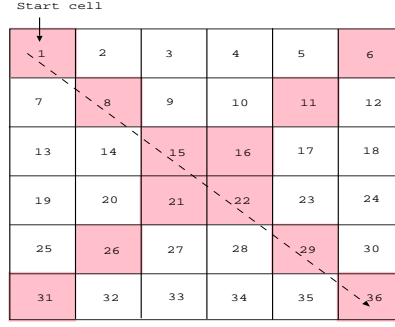


Figure 2.2: An example  $6 \times 6$  map.

Briefly, we now examine the different components of the control information being collected and broadcasted in each cell.

**1. Regional Maps and Mobility Model:** Several cells adjacent to each other can be grouped into a single regional map. Figure 2.2 illustrates such a map for a system with square cells. A base station locally monitors information about the number of AutoMata devices in its cell, which cell a device came from, and which cell a device is moving towards. This information from nearby cells is then used to construct a Markov inter-cell mobility transition table over this regional map (see Section 2.4).

**2. Data Item Replication Table:** Based on regional as well as global input, information is also maintained about the ID and duration of all data items, as well as their replication levels. This table would have  $T$  rows, one for each possible data item. While  $T$  might be potentially in the order of hundreds or thousands, note that each row is small and in the order of tens of bytes.

**3. Regional Lookahead Table:** Based on current data, a regional lookahead table is also created that maintains information about data items and AutoMatas within a certain cell's vicinity.

## 2.4 Simulation Model

We now describe the simulation model that is common to all our studies. We assume a repository of homogeneous data items with identical bandwidth requirement, display

time, and size ( $\beta_i = \beta$ ,  $\Delta_i = \Delta$ ,  $S_i = S$ ). Figure 2.2 shows an example map used in our study. The map is divided into fixed size cells. Only AutoMatas within a cell can communicate with each other either directly if they are in radio-range or via other AutoMatas using multi-hop transmissions. In other words, the AutoMatas within a cell form a connected sub-network. AutoMatas in adjacent cells cannot communicate with each other. Without any loss of generality, to reduce the dimensionality of the problem, we express the data item display time,  $\Delta$ , as the amount of time required by an AutoMata equipped vehicle to travel  $\Delta$  cells. We express  $\alpha$  as the number of storage slots per AutoMata. Each storage slot stores a data item fragment equivalent to a single cell worth of data item display time. Moreover, we assume the amount of data displayed in each cell is identical. Now, we represent both the size of a data item and the storage slots in terms of the number of cells. This means that a data item has a display time of  $\Delta$  cells and an AutoMata has  $\alpha$  units of cell storage. For example, a data item with display time of 4 cells ( $\Delta = 4$ ) requires 4 storage slots and an AutoMata provides 100 storage slots ( $\alpha = 100$ ).

The trip duration ( $\gamma$ ) is also expressed as the number of cells traversed by the client during its journey. We also define availability latency ( $\delta_i$ ) for data item  $i$  in terms of the number of cells. In other words,  $\delta_i$  is the number of cells after which a client AutoMata will encounter a replica of the data item  $i$ , either directly or via multiple hops, for the data item display time. Hence, the possible values of the availability latency are between 0 and  $\gamma$ . We only consider scenarios in which  $\Delta \leq \gamma$ . Assume that  $\gamma = 6$ . For a data item  $i$  with  $\Delta_i = 6$ ,  $\delta_i$  is either 0 or 6.  $\delta_i = 0$  means that at least one replica of that data item was present in each of the 6 cells along the path of the client.  $\delta_i = 6$  means that at least one cell along the path of the client was missing a replica of the data item. Similarly, for data item  $j$  with  $\Delta_j = 5$ ,  $\delta_j$  is either 0, 1 or 6. If  $\delta_j = 0$ , the client encountered at least one replica of data item  $j$  along each of the first 5 cells along its path. If  $\delta_j = 1$ , the client encountered at least one replica of the data item along the last 5 cells of its path, but not even a single replica in the first cell. Finally,  $\delta_j = 6$  indicates that there were at least 2 cells along the path of the client, in which no replicas of data item  $j$  were present.

As mentioned earlier, a Markovian mobility model describes the movement of the cars which is probabilistic in nature. The vehicles equipped with AutoMata devices perform a 2D random walk on the surface of a torus which constitutes the map. Each cell of the map constitutes a state. A map of size  $G \times G$  yields  $G^2$  states. These states are self-contained and a transition from one state to another is independent of the previous history of a car in that state. The mobility model is weighted toward the diagonal both from the left to right and vice-versa (indicated by the shaded boxes in Figure 2.2). The aggregate of the transitions from each cell (state) to every other state gives the  $G \times G$  probability transition matrix  $Q = [q_{ij}]$  where  $q_{ij}$  is the probability of transition from state  $i$  to state  $j$ . Using Markov chains, it is possible to estimate the distribution of the steady-state probabilities of being in the various cells, by solving  $\Pi = \Pi * Q$ , where  $\Pi$  is the vector representing the steady-state probabilities of being in the various cells (states).

We employ the Markov mobility model to dictate the movements of the vehicles. For the initial part of this thesis, we assume that the vehicles move about as per a random walk mobility model. Toward the latter half, we consider a Markov mobility model derived from an underlying real city map. The transitions of vehicles located in various cells are controlled by the freeway and side-street locations of the underlying city map. We collected real-time traffic data during different time periods during the day to obtain the percentage of vehicles transitioning between different freeways which in turn dictated the Markov model transition probabilities.

## Chapter 3

### PAVAN

In this chapter, we first define the availability problem in terms of the list of titles that will be available to a client during its journey. The client is presented the available list of titles with their associated latency. The idea is to present the client with as accurate a list as possible so as to avoid user frustration. We present PAVAN as a policy framework to generate this list, and evaluate how the different variants of PAVAN differ in their accuracy of the match between the predicted list and actual list.

#### 3.1 PAVAN variants

We now present the details of PAVAN and its alternate variants distinguished on the basis of the input information given to them. Taking examples of data items as video and audio clips, an example output produced by PAVAN is shown on the right hand side of Figure 3.1. In the following discussion, we will assume specific examples of data items as audio or video titles noting that it does not compromise the general applicability of PAVAN. The output of PAVAN is the available title list displayed on an interactive menu to the user showing all titles predicted to be available and their associated latency after which they will be available to the client. The prediction and presentation of the availability latency empowers users to make informed decisions.

The accuracy of PAVAN’s output depends on its provided information, i.e., its input. As noted in Section 2.3, there are three essential pieces of information that can be provided as input to PAVAN: the title replication table, the regional mobility table, and the regional

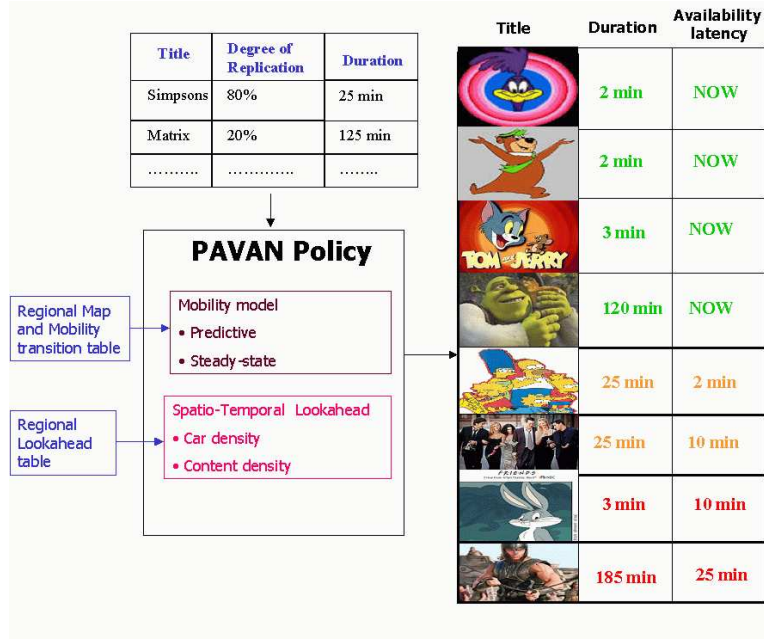


Figure 3.1: An overview of PAVAN, its inputs and output.

look-ahead table. The global title replication table (shown in Figure 3.1) is provided to all variants to PAVAN. These alternatives are different depending on whether PAVAN is provided with either the mobility table, lookahead table, or both. Intuitively, the “richer” the information input, the closer the output list is to the actual list (the list of titles produced by a oracle aware of all future movements of AutoMatas). There is a trade-off between obtaining richer information for the policy decision-making against the overhead of having this information broadcast from the base station to AutoMata devices. An additional input to PAVAN is the maximum delay tolerable by a client. This provides an upper bound on the availability latency.

The mobility model given to PAVAN may be categorized into two types. It may be either predictive ( $P$ ) where the transition matrix,  $Q$ , is used in each step, or steady-state ( $SS$ ) where only the equilibrium probabilities are used. Recall, the equilibrium probabilities are obtained by solving the equality  $\Pi = \Pi \cdot Q$ , where  $\Pi$  is the vector representing the steady-state probabilities of being in the various cells (states).

PAVAN Policy	Input information
$SS_{only}$	Steady-state mobility model
$SSL_t$	Steady-state mobility model and density of the contents in the AutoMatas within a pre-specified lookahead
$PL_t$	Predictive mobility model and density of the contents in the AutoMatas within a pre-specified lookahead
$PL_r$	Predictive mobility model and density of the AutoMatas within a pre-specified lookahead

Table 3.1: Four variants of PAVAN.

We now describe the alternative variants of PAVAN.  $SS_{only}$  provides PAVAN with both  $SS$  and the global replication table. This means the location of all AutoMatas is the same and is given by the steady-state matrix  $\Pi$ . Since no information about the contents of the AutoMatas is provided,  $SS_{only}$  uses the global replication table and assumes that an AutoMata contains the various titles governed by this global distribution. Aggregating this information for all AutoMatas yields the title location matrix  $T$ . Note that  $T$  has identical rows for each AutoMata since no information is known about their contents.

For each step along the path, the following procedure is applied:

**Algorithm 3.1.1:**  $PROCA(Steps, AutoMatas, titles)$

```

for  $step \leftarrow 1$  to  $Steps$ 
   $cell\_id \leftarrow clients\_current\_cell$ 
   $Conf \leftarrow 0$ 
  for  $i \leftarrow 1$  to  $AutoMatas$ 
    if (AutoMata  $i$  located in  $cell\_id$ )
      for  $j \leftarrow 1$  to  $titles$ 
        if (AutoMata  $i$  contains title  $j$ )
           $Conf(j, step) += \Pi(i, cell\_id) * T(i, j)$ 

```

Hence, for each title, this procedure yields the ‘Confidence’ of that particular title for that step along the journey of the client. The higher the confidence, the higher the predicted availability of the title at that step. At the end, the confidence for each title across all steps is aggregated into a metric that is then mapped into the list of available titles. This is achieved using the following procedure:

**Algorithm 3.1.2:** PROCB( $Steps, titles$ )

```

for  $i \leftarrow 1$  to  $titles$ 
  if  $Conf(i, j) \geq m$  for every step  $j, 1 \leq j \leq Steps$ 
     $Agg\_metric(i) = \sum_{j=1}^{Steps} Conf(i, j)$ 
  else  $Agg\_metric(i) = 0$ 

```

If, for title  $i$ ,  $Agg\_metric(i) > 0$ , then that title will appear in the client’s available list. In all our experiments, we choose a value of  $m = 1$ . Intuitively, if the title has a Confidence  $< 1$  at even one step, on an average less than 1 copy of that title exists at that step. Hence, the client may not find a copy of that title at that step. Such a title is not shown in the predicted list. It should be noted that values of  $m$  less than one result in optimistic predictions, while values of  $m$  greater than one result in conservative predictions.

In addition to the mobility model and the global replication table, PAVAN can be provided with the AutoMata density information and what content they carry. This can be limited to a specific area defined by a Spatio-Temporal Lookahead (STL) parameter.

### 3.1.1 Spatio-Temporal Lookahead (STL) parameter

The  $SSL_t$  variant of PAVAN consumes the global replication table,  $SS$ , and the contents of those AutoMatas within a fixed geographical area defined by STL. A STL<sup>1</sup> value of  $k$  encompasses all  $k$  adjacent cells. When STL is 0,  $SSL_t$  is similar to  $SS_{only}$ . Figure 3.2 shows an example  $5 \times 5$  map with the client occupying the shaded cell. This figure shows STL values of 1, 2, and 3. As one increases the value of STL, a client obtains

---

<sup>1</sup>We use  $k$  to denote the value of STL.

3	3	3	3	3
3	2	2	2	3
3	2	1	2	3
3	2	2	2	3
3	3	3	3	3

Figure 3.2: The numbers in the cells indicate STL value for the shaded cell numbered 1.

information about additional cells that are further away. Note that a cell is assumed to have eight adjacent neighbors.  $SSL_t$  enables an AutoMata to incorporate the content of all AutoMatas in  $k$  adjacent cells into  $T$ . The remaining AutoMata devices are assumed to contain titles as per the global replication table.

The  $PL_r$  variant of PAVAN considers the predictive mobility model ( $P$ ), the density of the AutoMatas within the STL and the global replication table to produce the title availability list. As the value of STL increases, the client obtains more information about the number of AutoMatas in the various cells. When STL spans all cells, the client obtains information about the number of AutoMatas in each cell of the entire network.

For a given client, PAVAN knows the location of AutoMatas within STL adjacent cells. For all the other AutoMatas, their location is equally likely to be a cell in the map outside those within the STL. This combined information about the initial positions of the AutoMatas yields the initial location matrix  $L$ . At each step, we compute product of  $L_i$  and  $Q_i$ , where  $i$  indicates the step under consideration,  $Q$  is the transition probability matrix defined by the mobility model, and the initial value of  $L_i = L$ . Note that the contents of the AutoMatas are not known; hence, the Title matrix  $T$  is calculated according to the global replication table. The  $L_i$  and  $T$  matrices are used with procedures PROCA (replacing  $\Pi$  by  $L$ ) and PROCB in order to obtain the predicted list of available titles.

Finally,  $PL_t$  denotes the variant of PAVAN with the following inputs: the global replication table,  $P$  and  $L_t$ . When  $k = 1$ ,  $PL_t$  is provided with the content of AutoMatas in its current cell, termed start cell.  $PL_t$  assumes the remaining AutoMatas are equally



likely to be in other cells of the network besides the ‘start cell’. This yields the initial AutoMata Location matrix  $L$ . Again, we compute product of  $L$  and  $Q_i$  at step  $i$  to obtain location matrix  $L_i$  at that step. Note that since the list of titles assigned to some of the AutoMatas is known, namely the AutoMatas present in the start cell, we incorporate that information in  $T$ .  $PL_t$  assumes other AutoMatas have the titles distributed as per the global title replication distribution. Hence, in this case, the rows of  $T$  need not all be the same. When  $k > 1$ , the client obtains precise information about the density and the contents of the AutoMatas in the cells that are reachable within a distance of  $k$  at the current instant. Using this information the client obtains the Location matrix  $L_i$  at each step  $i$  using  $L_i * Q_i$  where initially  $L_i = L$ . Similarly, the  $T$  matrix is obtained where the information of the contents of all AutoMatas within the STL is known.  $L_i$  and  $T$  can be input to PROCA (again replacing  $\Pi$  by  $L$ ) and PROCB to obtain  $Agg\_metric(i)$ , which is then converted into the client’s available titles list.

## 3.2 Simulation Study

We first describe the parameters of the simulation set-up, followed by a brief description of the simulation results.

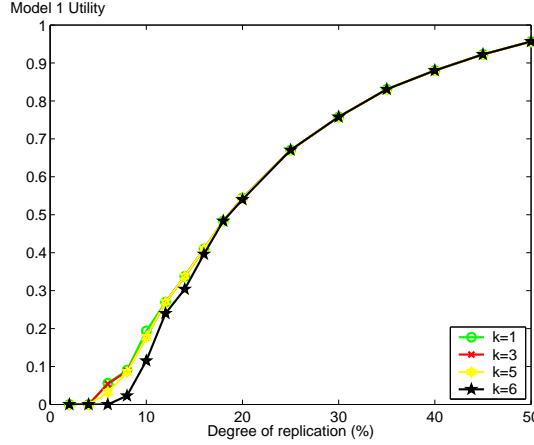
### 3.2.1 Experimental Set-up

The experimental set-up consists of a  $6 \times 6$  map as shown in Figure 3. The mobility model is weighted toward the diagonal both from left to right and vice-versa (due to gray boxes). Assume that the client starts from cell 1 and travels along the path  $\{1, 8, 15, 22, 29, 36\}$ . Numbers in the bracket indicate the sequence of visited cell IDs. At the start of a client’s journey, each variant of PAVAN retrieves its required information from the control network. Subsequently, each variant of PAVAN (see Table 3.1) produces a predicted list of available titles.

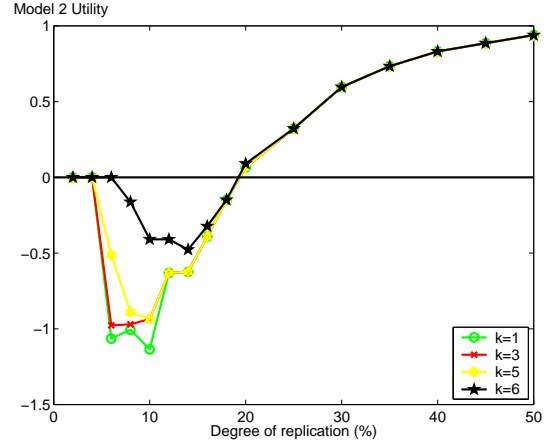
Initially, all AutoMatas are distributed uniformly across the cells in the map. This is determined by a random initial seed. The distribution of titles across AutoMatas is also chosen to be uniform. At each step, depending on the current AutoMata locations, each

Model	Weight( $w_1$ ) of $a_{10}$	Weight( $w_2$ ) of $a_{01}$	Weight( $w_3$ ) of $a_{11}$
1	0	0	1
2	0	-5	1
3	-1	0	1

Table 3.2: Three utility models to evaluate alternative variants of PAVAN.



3.3.a) Utility model 1



3.3.b) Utility model 2

Figure 3.3: A comparison of PAVAN with different inputs for utility models 1 and 2 as a function of the degree of replication of the titles.

AutoMata moves to one of its adjoining cell as governed by the mobility model. Another seed determines the choice of which cell an AutoMata moves to. Each AutoMata performs six transitions according to the mobility model. The intersection of AutoMatas with the cells along the client's path yields the actual confidence values for a particular title seen in a particular run of the simulation. For each run, a different random seed is used starting from the same initial position. For each run, at each step, the client obtains the exact distribution of titles in the network and the corresponding confidence values for each title. These values are then translated to a list of titles for this particular run (actual list) using the same procedure PROCB (see Section 3.1). For each run, the predicted list is compared with the actual list and the utility models presented later in Section 3.2.2 depict the differences. All presented results are averages across 10,000 simulation runs.

### 3.2.2 Utility Models

We use three utility models to quantify the quality of lists computed by different variants of PAVAN. These models assign a different weight to the average number of false negatives (denoted  $a_{10}$ ), false positives (denoted  $a_{01}$ ), and true positives (denoted  $a_{11}$ ). A false negative is a title present in the actual list but not in the predicted list. A false positive is a title present in the predicted list but not in the actual list. Finally, a true positive is a title present in both the actual and predicted lists.

All utility models are represented as:

$$U = w_1 \cdot a_{10} + w_2 \cdot a_{01} + w_3 \cdot a_{11}$$

We implement the alternative utility models by assigning a different weight to  $a_{10}$ ,  $a_{01}$ , and  $a_{11}$  (see Table 3.2). These models are as follows. Model 1 depends on those titles that appear correctly in both the actual and the predicted lists. So its utility value ranges from 0 to 1.

Model 2 severely penalizes those titles that appear in the predicted list but not in the actual list. It assumes that a user would be greatly dissatisfied by choosing such titles because they are not available. The utility of this model ranges in value from  $-5$  to 1.

Model 3 penalizes those titles that appear in the actual lists but not in the predicted ones. These available titles cannot be selected by a user because they are not predicated as available. The utility of this model ranges in value from  $-1$  to 1. Note that the penalty for these false negatives is not as significant as false positives.

### 3.2.3 Results

In our experiments, we used 200 AutoMatas, unless stated otherwise, and 16 titles with unique ids 1, 2,  $\dots$  16. The percentage degree of replication of a title with id  $i$  is given by:

$$Title - rep(i) = \begin{cases} 2 \cdot i & 1 \leq i \leq 10 \\ 20 + 5 \cdot (i \% 10) & 11 \leq i \leq 16 \end{cases} \quad (3.1)$$

This means title 1 has 4 copies, title 2 has 8 copies, and so on until title 10. Title 11 has 25 copies, title 12 has 30 copies, and so on until title 16. Replicas of a title are

assigned to AutoMatas randomly. An AutoMata may contain several different titles, but only one copy of a certain title.

Figure 3.3 presents a comparison of alternative variants of PAVAN. The graphs represent the utility values as a function of the different degrees of replication of the movie titles. The predicted lists generated by PAVAN in all cases (where applicable) were calculated using the largest STL value (here,  $k=6$ , the length of the path of the client). Next, we briefly describe the main lessons of this study.

**Lesson 1: As the degree of replication increases beyond a certain threshold all the variants of PAVAN start showing similar utilities.** The value of the threshold is different for different models. While for model 1, this replication threshold is 20%, it is approximately 50% with model 2. Two factors impact this observation. First is the degree of title replication. Second is the predictive nature of a specific PAVAN policy.

The general trend indicates that as the degree of replication increases, the model utility values also increase and converge toward 1 (maximum utility value for all models). With the increase in the degree of title replication, the global replication table, which is the base-line input to PAVAN, dominates the titles shown in the predicted lists yielding higher true positives ( $a_{11}$ ). Here, both false-positives and false-negatives contribute an insignificant amount toward the final observed utility for all models.

Model 2, which penalizes those titles that are present in the predicted but not in the actual list, highlights the differences between the PAVAN variants. It is seen that, in general,  $PL_t$  outperforms the others. This is because it uses information about the density and the contents of the AutoMatas within the STL. Since this utility model penalizes policies that over-predict, we see that  $SS_{only}$  performs the worst followed by  $SSL_t$ . With lower replication levels, in case of model 1, these policies were doing marginally better than  $PL_t$  because their over-predictive nature always resulted in higher  $a_{11}$  values. The performance of  $PL_r$ , which uses AutoMata density information and the predictive mobility model, lies between the two extremes.

The results above indicate that  $SS_{only}$  and  $PL_t$  represent the two extremes. Hence, we eliminate results from the other two variants for the remaining sets of experiments noting that their performance was always in between  $SS_{only}$  and  $PL_t$ .

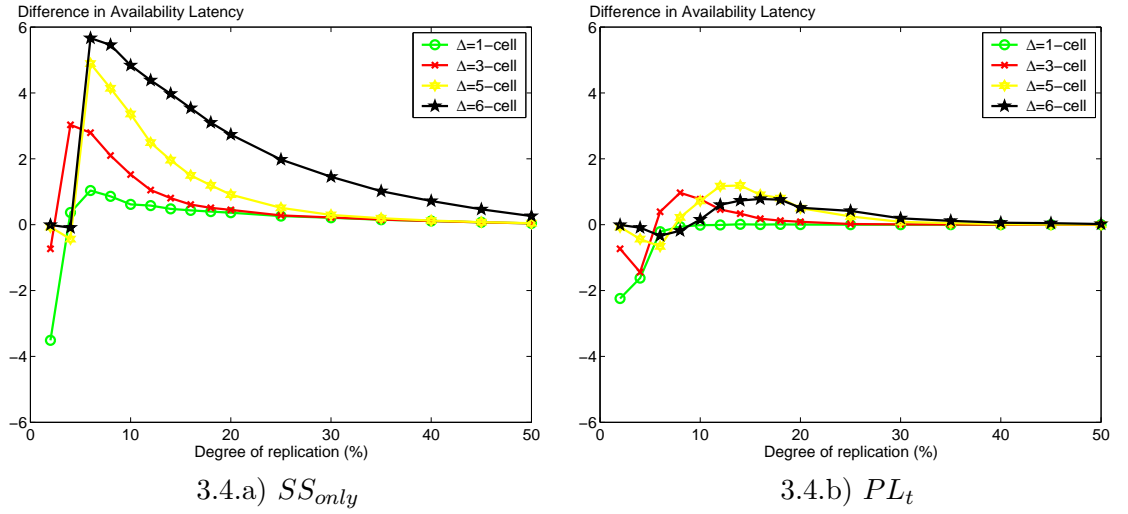


Figure 3.4: Difference in the availability latencies as a function of the degree of replication of the titles for different title display times.

**Lesson 2: The accuracy of availability latency estimated by  $PL_t$  is the best when compared with other alternatives.** Figure 3.4.a indicates the average difference between the availability latency of  $SS_{only}$  and the actual observed latency as a function of the different degrees of title replication for different title display times. The graph shows the behavior for title display times of 1, 3, 5 and 6 cells. Figure 3.4.b shows the same for  $PL_t$ . Note the lower the difference in the availability latency, the better the match between the predicted and the actual lists. We observe that the availability latency of alternative PAVAN policies is significantly different and impacted by both the display time of a title and trip duration. The main observation is that the peaks in the curves for  $SS_{only}$  (over-predictive) are much higher than that for  $PL_t$  (conservative).

With  $SS_{only}$ , the predicted list is very accurate when title display times are greater than two and the degree of replication is less than 5%. This is because the average availability latency is close to 6 in both cases. However, as the degree of replication increases beyond 5%, the predicted availability latency drops at a faster rate than what is seen with the actual availability latency. This difference is always positive because  $SS_{only}$  always over-predicts irrespective of the title display time. Note that over-prediction indicates a smaller predicted availability latency as compared to the actual one. Beyond

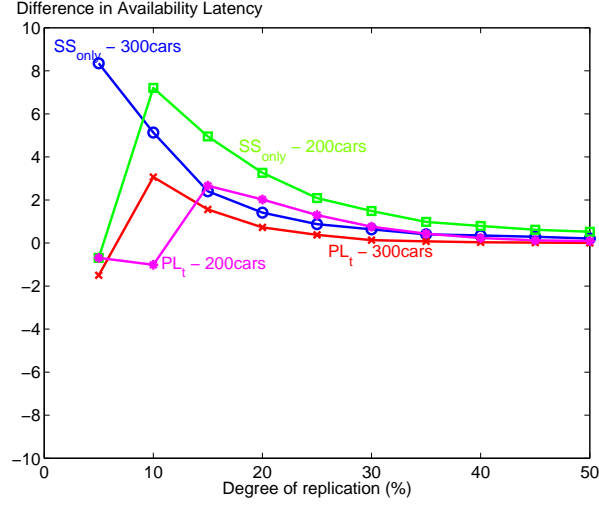


Figure 3.5: Comparison of difference in the availability latencies of  $SS_{only}$  and  $PL_t$  for different AutoMata densities in a 10x10 map as a function of the degree of title replication. The title display time under consideration spans 5 cells.

25% degree of replication, availability latency of all titles except those with display time 6 converge to zero. As the degree of replication increases, slowly the actual availability latency catches up with the predicted one thereby making the difference between them converge to zero.

With  $PL_t$ , the availability latency difference always lies between -1 and +1.5 irrespective of the title display time. For all display times, beyond 20% replication, the difference in the availability latency converges to 0. For degree of replication less than or equal to 5%, the difference in the availability latency becomes negative. This means that the predicted available latency is higher than the actual observed latency, thereby indicating that for lower degrees of replication  $PL_t$  is more conservative. The reason that the difference is not always 0 is due to the statistical variations inherent in the experiments. The mobility model is probabilistic, hence if we consider extremely large scenarios, then even for lower degrees of replication, the difference in the availability latency will converge to 0.

**Lesson 3: The accuracy of the availability latency calculations for different variants of PAVAN is sensitive to AutoMata density.** Figures 3.5 indicates the

behavior of  $SS_{only}$  and  $PL_t$  with respect to the difference in availability latency metric in a 10x10 map for AutoMata densities of 200 and 300, when considering a title with a display time of 5 cells.

When the AutoMata density increases, for a given degree of replication, now there are more number of replicas (AutoMatas) for each title. Hence, similar trends are seen as earlier but the curves for all the PAVAN variants move to the left. Moreover, the curves peak at a lower degree of replication.

Even though we have presented the results for one map, we also considered other maps in our simulations. Obtained results show the map is an important parameter that impacts the behavior of PAVAN significantly. Specifically, a lower degree of overlap between the path of an AutoMata equipped vehicle and the gray cells in the map causes the transition probabilities to rapidly diminish toward 0. In such cases, even if the degree of replication of the titles is 100% the variants of PAVAN will always under-predict. This will be the case even if the total number of gray cells increases beyond a certain threshold because then the transition probabilities diffuse quickly. So, within a few steps, the movement prediction probabilities will diminish having very little effect on the predicted lists even in the case of 100% title replication. This effect will also be seen if the map consists of entirely non-gray cells. In such cases, the trends are similar to those seen for the lower replication titles.

In conclusion,  $SS_{only}$  is appropriate for certain utility models but not all. Also, the degree of title replication has a profound impact on the availability latency metric.  $PL_t$  demonstrates a competitive performance for all utility models, all clip display times and degrees of replication.

### 3.3 Summary

PAVAN is a novel policy that computes the time when different titles are available in an ad-hoc network of AutoMata devices. It accomplishes this by employing a Markov mobility model that consumes a regional map, a mobility transition table, and a regional look ahead table. This input data is in the order of a few hundred bytes and provided by a base station. Each AutoMata device invokes PAVAN independently. Obtained results

demonstrate that one variant of PAVAN, that employs information about the density of the vehicles, the content that they carry and their mobility pattern, provides the most accurate availability latency when compared with other techniques. We quantified the quality of lists computed by PAVAN policies using different utility models.

The accuracy of the PAVAN predictions crucially depends on the transition probabilities of the Markov model. Unfortunately, with the UMassDieselNet [11] traces, the locations of the buses were not available. For a data set that has an underlying map and recorded locations of different vehicles with some fine/coarse granularity as they move about constrained by the map, one can create a Markov model for this data set. The Markov model will be employed by the PAVAN module to predict the list of available titles and the accuracy of the predictions will be measured by looking at the actual vehicular traces. This remains a future research direction as and when such a data set becomes publicly available. Synthetic data sets obtained from microscopic vehicular simulators like VISSIM/CORSIM may also be used for the above process, bringing the evaluations a step closer to reality.



## Chapter 4

### Static Replication Schemes

In this chapter, the focus is on how the degree of replication per data item affects availability latency. We consider a family of frequency-based replication strategies and study their impact on availability latency. First, a general optimization formulation is presented to determine which replication scheme minimizes the aggregate availability latency subject to a total storage constraint. Subsequently small data items and long client trip durations, we solve the optimization in the case of sparse density of vehicles. Then, we explore the latency performance in high density scenarios via simulations and present an analytical approximation that captures the trends. The results are extended to consider data items of larger size and short client trip durations. Subsequently, some of the results obtained with a 2D random walk model are evaluated on a map of the city of San Francisco with the major freeways being captured by the transition probabilities of the Markov mobility model. Finally, we explore the performance of the replication schemes on a realistic data set comprising of movement traces of buses in a small neighborhood in Amherst.

#### 4.1 Family of Replication Policies

Given a data item repository ( $T$ ), a certain vehicle density ( $N$ ), and a storage constraint per vehicle (each with storage  $\alpha$ ), we present an optimization formulation to determine the optimal number of data item replicas that minimize the average availability latency metric. We simplify the data placement issue by allocating the replicas to the vehicles

uniformly at random with the constraint that no two replicas of the same data item are placed in a vehicle.

We define the normalized frequency of access to the data item  $i$ , denoted  $R_i$ , as:

$$R_i = \frac{(f_i)^n}{\sum_{j=1}^T (f_j)^n}; 0 \leq n \leq \infty \quad (4.1)$$

$R_i$  is normalized to a value between 0 and 1. The number of replicas for data item  $i$ , denoted as  $r_i$ , is:

$$r_i = \min(N, \max(1, \lfloor \frac{R_i * N * \alpha}{S_i} \rfloor)) \quad (4.2)$$

This defines a family of replication schemes that computes the degree of replication of data item  $i$  as the  $n^{th}$  power of its frequency of access. Hence, the number of replicas for title  $i$ ,  $r_i$ , lies between 1 and  $N$ . Note that  $r_i$  includes the original copy of a data item. One may simplify Equation 4.2 by replacing the max function with  $\lfloor \frac{R_i * N * \alpha}{S_i} \rfloor$ . This would allow the value of  $r_i$  to drop to zero for a data item  $i$ . This means that there is no copy of the data item in the network. In this case, a hybrid framework might provide access to the data item  $i$ . For example, a base station employing IEEE 802.16 [22] might facilitate access to a wired infrastructure with remote servers containing the data item  $i$ .

The aggregate availability latency,  $\delta_{agg}$ , depends on the value chosen for  $n$ , since  $n$  determines the replicas per data item. Intuitively, the higher the replicas for a data item  $i$ , the lower will be the latency,  $\delta_i$ , experienced by a request for that data item. The core problem of interest here is to keep the aggregate availability latency as low as possible by tuning the data item replication levels, in the presence of storage constraints. We assume that the database size is smaller than the total storage capacity of the system,  $\sum_{i=1}^T S_i \leq S_T$ . Otherwise, data items cannot be replicated when at least one replica of a data item must be present in the system. More formally, the optimization problem can be stated as,

$$\text{Minimize } \delta_{agg}, \text{ subject to } \sum_{i=1}^T S_i \leq S_T \quad (4.3)$$

Implicit in this formulation is the design variable, namely, the desired replication for each data item. The value of  $n$  in Equation 4.3 determines a  $r_i$  value for each data item

$i$  with the objective to minimize  $\delta_{agg}$ . This minimization is a challenge when the total size of the database exceeds the storage capacity of a car,  $\sum_{i=1}^T S_i > \alpha$ . Otherwise, the problem is trivial and can be solved by replicating the data item repository on each device.

The optimization space that defines what value of  $n$  provides the best  $\delta_{agg}$  is quite large and consists of the following parameters: (i) density of cars, (ii) data item display time, (iii) size of the data item, (iv) display bandwidth per data item, (v) data item repository size, (vi) storage per car, (vii) client trip duration, (viii) frequency of access to the data items, and (ix) mobility model for the cars. We first explore this parameter space where the mobility model employed by the vehicles is a 2D random walk on the surface of a torus. Not only does this provide tractability for mathematical analysis, but it turns out that the biased Markov mobility models based on an underlying city map comprising of freeways and side-streets show performance trends similar to those observed with the simple 2D random walk based mobility model (see Section 4.6).

## 4.2 Data items with display time one and long client trip duration

In this section, we consider small data items i.e. items with a display time of one, where the client trip duration is long. We first present analytical approximations that capture the performance of availability latency for an item as a function of the number of replicas for that item for both a low and high density of replicas. Subsequently, we employ simulations to determine the optimal replication exponent that minimizes the aggregate availability latency.

### 4.2.1 Analysis

In this section, we assume data items with a display time of one cell and for a scenario with a sparse density of data item replicas, derive a closed-form expression for the aggregate availability latency. Subsequently, we use this expression to solve the optimization problem to reveal that a square-root replication scheme minimizes this latency. Then,

we derive an expression that approximates the aggregate availability latency in case of a high density of data item replicas.

#### 4.2.1.1 Sparse Scenario

In this section, we provide a formulation that captures scenarios with a low density of vehicles. One can obtain the relationship between the  $\delta_i$  and  $r_i$  under a given storage constraint. In general, the relationship is a function of the mobility model of the vehicles. For illustration, we have considered that vehicles follow a random walk-based mobility model on a 2D-torus. Aldous *et al.* [2] show that the mean of the hitting time for a symmetric random walk on the surface of a 2D-torus is  $\Theta(G \log G)$  where  $G$  is the number of cells in the torus. Moreover, the mean of the meeting time for 2 random walks is half of the mean hitting time. Furthermore, the distribution of the meeting times for an ergodic Markov chain can be approximated by an exponential distribution of the same mean [2]. Hence,

$$P(\delta_i > t) = \exp\left(\frac{-t}{c \cdot G \cdot \log G}\right) \quad (4.4)$$

where the constant  $c \simeq 0.34$  for  $G \geq 25$ . Now since there are  $r_i$  replicas, there are  $r_i$  potential servers. Hence, the the meeting time, or equivalently the availability latency for the data item  $i$  is the time till it encounters any of these  $r_i$  replicas for the first time. This can be modelled as a minimum of  $r_i$  exponentials. Hence,

$$P(\delta_i > t) = \exp\left(\frac{-t}{c \cdot \frac{G}{r_i} \cdot \log G}\right) \quad (4.5)$$

Note, however that this formulation is valid only for the cases when  $G \gg r_i$ , which is the case for sparse scenarios. The expected value of  $\delta_i$  is given by:

$$\bar{\delta}_i = \frac{c \cdot G \cdot \log G}{r_i} \quad (4.6)$$

For a given 2D-torus,  $G$  is constant, hence we have  $\bar{\delta}_i \propto \frac{1}{r_i}$  or equivalently,  $\bar{\delta}_i = \frac{C}{r_i}$  where  $C = c \cdot G \cdot \log G$ .

Hence, we have the following optimization formulation,

$$\text{Min } \left[ \sum_{i=1}^T f_i \cdot \frac{C}{r_i} \right] \quad (4.7)$$

Subject to:

$$\sum_{i=1}^T r_i = N \cdot \alpha \quad (4.8)$$

$$r_i \leq N ; \forall i = 1 \text{ to } T \quad (4.9)$$

$$r_i \geq 1 ; \forall i = 1 \text{ to } T \quad (4.10)$$

**Theorem 1.** *In case of a sparse density of vehicles, a replication scheme that allocates data item replicas as a function of the square-root of the frequency of access to data items minimizes the aggregate availability latency.*

$$r_i = \begin{cases} \frac{\sqrt{f_i} \cdot N \cdot \alpha}{\sum_{j=1}^T \sqrt{f_j}} & \frac{1}{N \cdot \alpha} \leq \frac{\sqrt{f_i}}{\sum_{j=1}^T \sqrt{f_j}} \leq \frac{1}{\alpha} \\ \max \left( 1, \min \left( \sqrt{\frac{f_i \cdot C}{\gamma_0}}, N \right) \right) & \text{in the general case} \\ & \text{where } \gamma_0 \text{ is s.t. } \sum_{i=1}^T r_i = N \cdot \alpha \end{cases} \quad (4.11)$$

*Proof.* We solve the above optimization using the method of Lagrange multipliers. First, we prove part(i) of the theorem.

The Lagrangian for the optimization can be written as:

$$H = \sum_{i=1}^T \frac{f_i \cdot C}{r_i} + \varphi \left[ \sum_{i=1}^T r_i - N \cdot \alpha \right] \quad (4.12)$$

We solve for  $r_i$  as follows:

$$\frac{\partial H}{\partial r_i} = -f_i \cdot \frac{C}{r_i^2} + \varphi = 0 \quad (4.13)$$

$$r_i = \sqrt{\frac{C \cdot f_i}{\varphi}} \quad (4.14)$$

Substituting  $r_i$  in the constraint, we get:

$$\varphi = \left( \frac{\sum_{i=1}^T \sqrt{C \cdot f_i}}{N \cdot \alpha} \right)^2 \quad (4.15)$$

Finally, we get the optimal value of  $r_i$  as,

$$r_i = \frac{\sqrt{f_i} \cdot N \cdot \alpha}{\sum_{j=1}^T \sqrt{f_j}} \quad (4.16)$$

The constraints are satisfied if  $\frac{1}{N \cdot \alpha} \leq \frac{\sqrt{f_i}}{\sum_{j=1}^T \sqrt{f_j}} \leq \frac{1}{\alpha}$  which proves part (i) of the theorem.

Without this condition on  $f_i$ , the above optimization can be re-written as the following Lagrangian taking all the constraints into account as:

$$G = \sum_{i=1}^T \frac{f_i \cdot C}{r_i} + \gamma_0 \left[ \sum_{i=1}^T r_i - N \cdot \alpha \right] - \sum_{i=1}^T \gamma_i \cdot (r_i - N \cdot \alpha) - \sum_{i=1}^T \beta_i \cdot (-r_i + 1) \quad (4.17)$$

The Kuhn Tucker Conditions for the modified Lagrangian are:

$$-f_i \cdot \frac{C}{r_i^2} + \gamma_0 - \gamma_i + \beta_i = 0; \forall i = 1 \text{ to } T \quad (4.18)$$

$$\sum_{i=1}^T r_i \leq N \cdot \alpha, \gamma_0 \geq 0, \text{ and } \gamma_0 \left[ \sum_{i=1}^T r_i - N \cdot \alpha \right] = 0 \quad (4.19)$$

$$r_i \leq N, \gamma_i \geq 0, \text{ and } \gamma_i \cdot (r_i - N) = 0; \forall i = 1 \text{ to } T \quad (4.20)$$

$$-r_i \leq -1, \beta_i \geq 0, \text{ and } \beta_i \cdot (-r_i + 1) = 0; \forall i = 1 \text{ to } T \quad (4.21)$$

Solving Equation 4.18, we get,

$$r_i = \sqrt{\frac{f_i \cdot C}{\gamma_0 - \gamma_i + \beta_i}} \quad (4.22)$$

Equations 4.20 and 4.21 imply that either  $\gamma_i = 0$  or  $r_i = N$  and also either  $\beta_i = 0$  or  $r_i = 1$  respectively. Therefore, the optimum solution for  $r_i$  is given by,

$$r_i = \max \left( 1, \min \left( \sqrt{\frac{f_i \cdot C}{\gamma_0}}, N \right) \right) \quad (4.23)$$

where  $\gamma_0$  is such that  $\sum_{i=1}^T r_i = N \cdot \alpha$  proving part (ii) of the theorem.  $\square$

Hence, in a sparse network, the optimal replication that minimizes the aggregate availability latency is obtained if the number of replicas for a data item is proportional to the square root of the frequency of access for that data item. Cohen *et al.* [13] proved that for unstructured peer-to-peer networks the expected search size is minimized using a square-root replication strategy which is shown to be optimal. The aggregate availability latency metric in wireless mobile ad-hoc networks is analogous to the expected search size used in peer-to-peer networks.

It should be noted that, in general, the optimal replication depends on how  $\delta_i$  is related to  $r_i$  i.e.  $\delta_i = F(r_i)$  and  $F(\cdot)$  is the function that will determine the optimal replication strategy. The above methodology can be used to obtain the optimal number of replicas as long as  $F(\cdot)$  is differentiable.

Figure 4.1 shows the typical trend shown by  $\delta_i$  for a  $10 \times 10$  torus, where  $r_i$  is increased from 1 to  $N$  where  $N = 100$ . In other words, in a  $G = 100$  cell torus,  $N = 100$  cars are deployed, with  $r_i$  of them having a replica for the data item. We only consider a single data item, a request for that item can be issued at any vehicle chosen uniformly at random among all the cars. If the item is stored locally, the latency is 0. The figure indicates that when  $r_i$  is small, ( $r_i \leq 20$ ) the analytical approximation in Equation 4.6 is valid. Subsequently, latency reduces at a much faster rate when compared to that predicted by the sparse approximation. This is because for a given  $G$ , as  $r_i$  increases, the latency till any one of the  $r_i$  replicas is encountered can no longer be modeled as the minimum

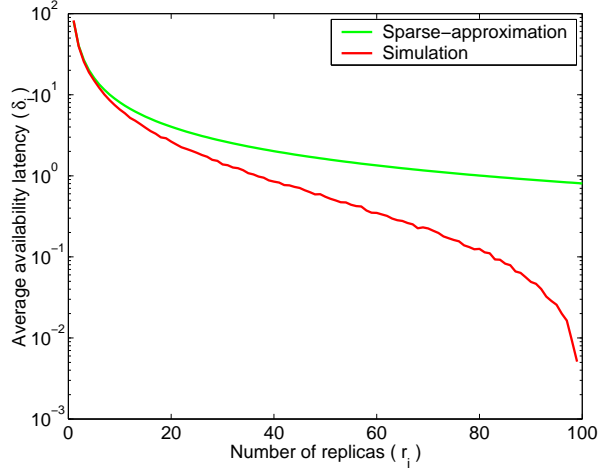


Figure 4.1: Sparse analysis ( Equation 4.6) versus simulation obtained average availability latency for a data item as a function of its replicas for a  $10 \times 10$  torus, when the number of cars is set to 100.

of  $r_i$  independent exponentials. In the next section, we provide an approximation that captures the high density case.

#### 4.2.1.2 Dense scenario

In this section, we provide an analytical formulation that captures the trends shown by the availability latency in the presence of a high density of replicas. Recall that  $N$  cars are distributed uniformly at random across  $G$  cells,  $r_i$  off the  $N$  cars carry a copy of the data item of interest. Here, we use the traditional definition of the expected availability latency for title  $i$ , namely,

$$\bar{\delta}_i = \sum_{k=0}^{\infty} k \cdot P(\delta_i = k) \quad (4.24)$$

We first determine an expression for the case when the latency is 0. This occurs if the data item is locally stored at a client or a data item replica is located in the same cell as the client at which the request is issued. Hence, the probability that the latency experienced by a client is zero is given by the following expression:

$$P(\delta_i = 0) = \frac{r_i}{N} + \left(1 - \frac{r_i}{N}\right) \cdot \left(1 - \left(1 - \frac{1}{G}\right)^{r_i}\right) \quad (4.25)$$



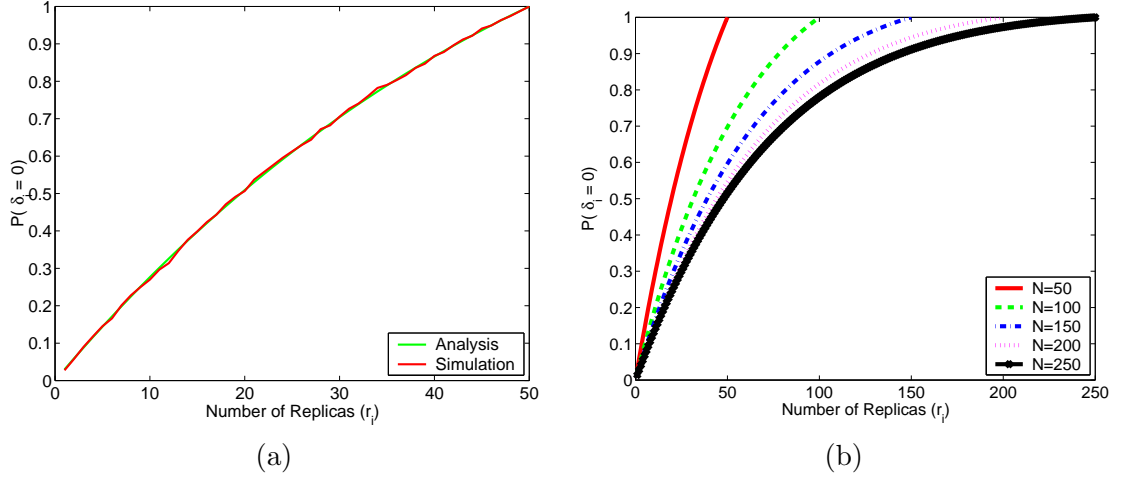


Figure 4.2: Figure 4.2(a) shows the validation of the analytical expression in Equation 4.25 for the probability that availability latency is zero. Figure 4.2(b) shows the probability that the availability latency is zero as a function of the replicas for the data item for 5 different car densities  $\{50, 100, 150, 200, 250\}$ .

Figure 4.2(a) indicates that the analytical expression above matches the simulation results quite well. For a given car density  $N$ , as the density of replicas increases, the probability that the availability latency experienced by a client is zero also increases. Figure 4.2(b) shows how this probability varies with increasing car density. Given a torus comprising  $G$  cells, increase in  $P(\delta_i = 0)$  shows a decreasing steepness as  $N$  increases. This is because  $r_i$  varies from 1 to  $N$ , and only when  $r_i = N$ ,  $P(\delta_i = 0) = 1$  since the data item is locally stored by every car.

Define  $A_k$  as the event that a data item  $i$  is encountered by the client for the *first* time in the  $k^{th}$  cell. Let  $P(A_k)$  denote the probability of event  $A_k$  occurring. Let  $p_k$  denote the probability of encountering data item  $i$  in the  $k^{th}$  cell, given that it was not encountered in the previous  $k - 1$  cells. Note that  $p_k$  is a conditional probability. Also,  $p_1 = P(\delta_i = 0)$  as defined by Equation 4.25. Then,

$$p_k = 1 - \left(1 - \frac{1}{G - k + 1}\right)^{r_i} ; 2 \leq k \leq G \quad (4.26)$$

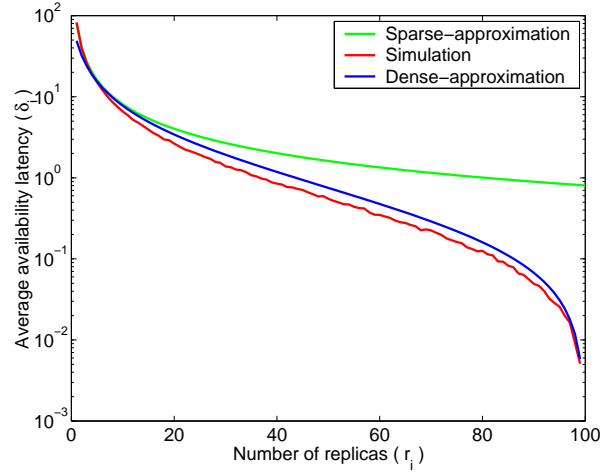


Figure 4.3: The complete picture depicting the availability latency for a data item obtained via simulations as compared with its sparse and dense approximation as a function of its replicas for a  $10 \times 10$  torus, when the number of cars is set to 100.

Note that the model assumes that not encountering the data item in the  $(k - 1)^{th}$  cell increases the probability of encountering it in the  $k^{th}$  cell. Moreover, when  $k = G$ ,  $p_k = 1$  no matter what the value of  $r_i$ , meaning that the maximum latency that a client will encounter will always be no more than  $G$ . Although this is true for a high density of replicas, this approximation is not valid for a sparse replica density, where  $p_k$  may not increase as  $k$  increases, especially for the first few steps of the client.

Note that,  $P(A_k)$  is a joint probability since encountering a data item for first time in the  $k^{th}$  cell indicates that it was not encountered in any of the previous  $k - 1$  cells. Clearly,  $p_k$  and  $p_{k-1}$  are not independent, hence, we use the multiplication rule to obtain the value of  $P(A_k)$  as,

$$P(A_k) = p_k \prod_{j=1}^{k-1} (1 - p_j) ; 2 \leq k \leq G \quad (4.27)$$

Then, the average availability latency  $(\bar{\delta}_i)$  for data item  $i$  is given by,

$$\bar{\delta}_i = \sum_{k=1}^G (k - 1) P(A_k) \quad (4.28)$$

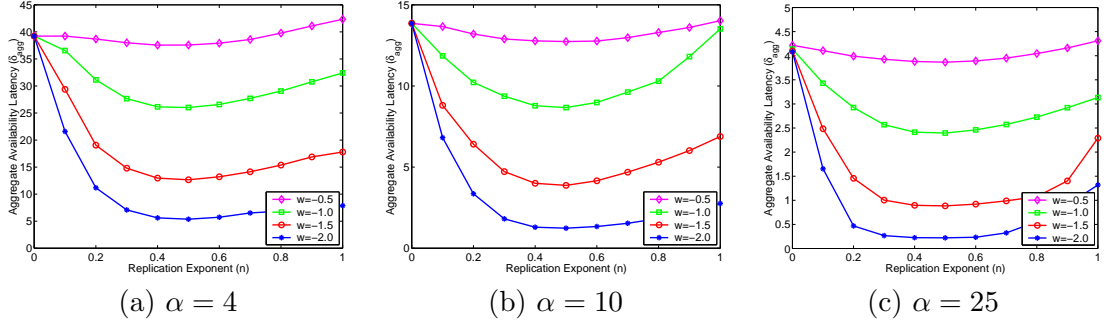


Figure 4.4: Aggregate availability latency for different replication strategies for a  $10 \times 10$  torus when  $T = 100$  and  $N = 50$ . Figures (a), (b), and (c) depict three different storage values per car:  $\{4, 10, 25\}$ .

Figure 4.3 shows that the above equation captures the trend depicted by the average availability latency for higher replica densities where the sparse and dense approximations are plotted together with the latency obtained via simulations.

#### 4.2.2 Simulation results

In this section, we present how the aggregate availability latency realized by the different replication schemes is affected individually by the different parameters in the optimization space.

In all our experiments, we assume that the various data item popularities are distributed as per the Zipf's law [65]. This means that the frequency of the  $r^{th}$  popular data item is inversely proportional to its rank i.e.

$$f_i = \frac{\frac{1}{i^v}}{\sum_{j=1}^T \frac{1}{j^v}}; 1 \leq i \leq T \quad (4.29)$$

Here, the exponent  $v$  controls the skewness in the popularity distribution of the data items. We denote  $w = -v$  as the skewness parameter. A higher value of  $w$  indicates that most of the popularity weight is spread across the first few popular titles. Note that the data item repository size is  $T$  and the denominator is simply a normalization constant.

Figure 4.4 depicts the latency performance for different replication schemes when storage per car is increased from 4 to 25 slots. The title repository size is  $T = 100$

and the car density is  $N = 50$  which implies that the total storage  $S_T$  is increased from 200 to 1250 slots. As expected the latency decreases as storage is increased. The replication schemes with exponent values 0, 0.5, and 1 have been popularly studied in the literature [13, 42, 17, 60, 58] and are labelled **random**, **square-root**, and **linear** respectively. Below, we describe the main observations from this figure.

The random scheme allocates the same number of replicas per data item irrespective of their popularity. Hence, in all cases, it yields the same aggregate availability latency irrespective of the value of  $w$ . As the replication exponent increases from 0 to 1 progressively more replicas are allocated for the popular data items. This increase in the replicas is accelerated for higher values of  $w$  that provide a bias for the popular titles. Hence, we see a sharp decrease in the availability latency from  $n = 0$  to  $n = 0.3$  for  $w = -1.5$  and  $w = -2$ . However, the maximum number of replicas per data item can never exceed  $N$ . For a value of  $w = -0.5$  in which case the popularity weight is spread more evenly among all the data items, it almost doesn't matter what the replication scheme is as seen by the flat latency curves for  $w = -0.5$ .

When storage per car is low,  $\alpha = 4$ , this represents a scenario with a sparse density of data item replicas. In this case, the square-root replication scheme provides the minimum latency. Also, the range where the replication exponent  $n$  varies from 0.4 to 0.6 shows a latency very close to the square-root scheme. This is true even when the data item popularities are skewed. Moreover, the range  $0.4 \leq n \leq 0.6$  shows near optimal latency performance even when the storage is increased (see Figures 4.4(b) and (c)). In other words, through the entire spectrum of the replica density, a replication scheme defined by an exponent in this range will provide near optimal performance. For the rest of this chapter, we will consider the square-root ( $n = 0.5$ ) scheme as representative of this range and compare its performance to the two extremes namely, random ( $n = 0$ ) and linear ( $n = 1$ ).

#### 4.2.2.1 Scale-up experiments

In these set of experiments, we maintain a constant ratio of the total storage to the data item repository size ( $S_T : T$ ). Figure 4.5 presents the performance of the three

replication schemes when  $S_T = 3000$  and  $T = 600$ . Since  $S_T = N \cdot \alpha$ , we vary the values of  $(N, \alpha)$  as  $\{(50, 60), (100, 30), (150, 20), (200, 15), (250, 12)\}$ . Since the total storage in the system  $S_T$  remains the same the number of replicas allocated per data item also remains. As  $N$  increases, the number of potential clients increases, this accounts for the slight upward trend in the latency curves for the different replication schemes. With increasing skewness, for the same total storage, the latency realized by the square-root and linear schemes reduces. This is because replicas assigned to the more popular data items result in lower latency for those items because as the skewness parameter  $w$  increases, a higher popularity weight assigned to these data items. The random replication scheme is blind to the popularity of the data items and hence shows similar latency performance independent of the value of  $w$ . For all but  $w = -0.5$ , it performs an order of magnitude worse as compared to the square-root scheme.

#### 4.2.2.2 Variation in car density

Next, we study the effect of car density on the performance of the replication schemes. Figure 4.6 presents the performance of the three replication schemes as a function of the car density when the storage per car is held constant at 3 for  $T = 100$ . Increase in the car density increases the total storage in the system. Hence, more replicas per data item can be allocated resulting in an overall decrease in the aggregate availability latency. This is true for all replication schemes. However, here for  $w = -0.5$  and  $w = -1$  (beyond  $N = 100$ ), the random scheme shows slightly better performance than the linear scheme. This is because for a lower skew in title popularities, assigning equal number of replicas per data item is better than providing higher replicas for the popular data items which do not have a sufficiently high popularity weight. However, for higher skew in popularity, the behavior of the linear scheme starts paying richer dividends in reducing the overall latency, hence, it outperforms the random scheme. In all cases, the square-root scheme always yields the lowest aggregate availability latency.

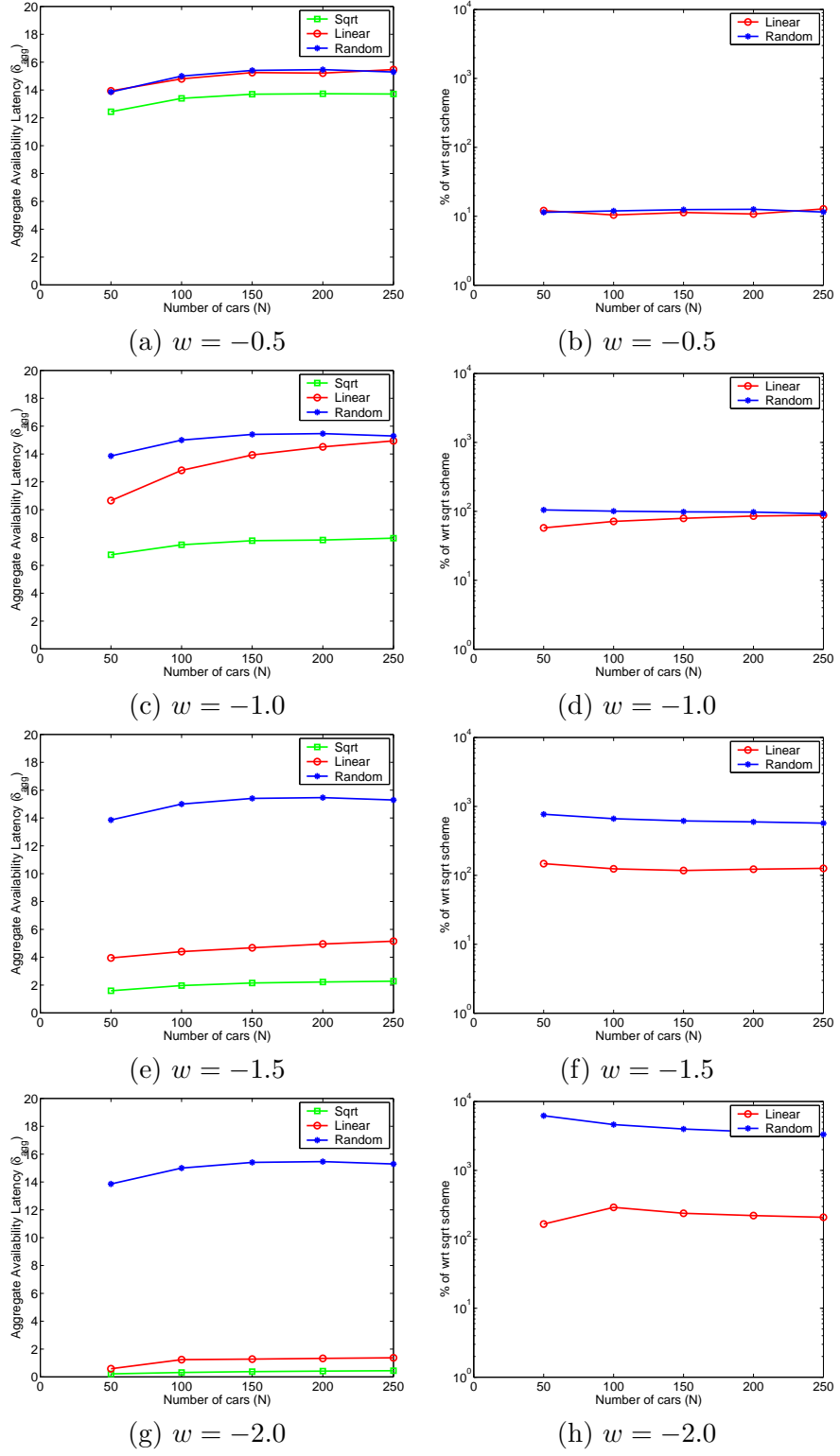


Figure 4.5: Scale-up experiments where the total storage to the data item repository size is held constant at  $\frac{S_T}{T} = \frac{3000}{600}$ . The number of cars and the storage per car are varied to realize  $S_T = 3000$ .

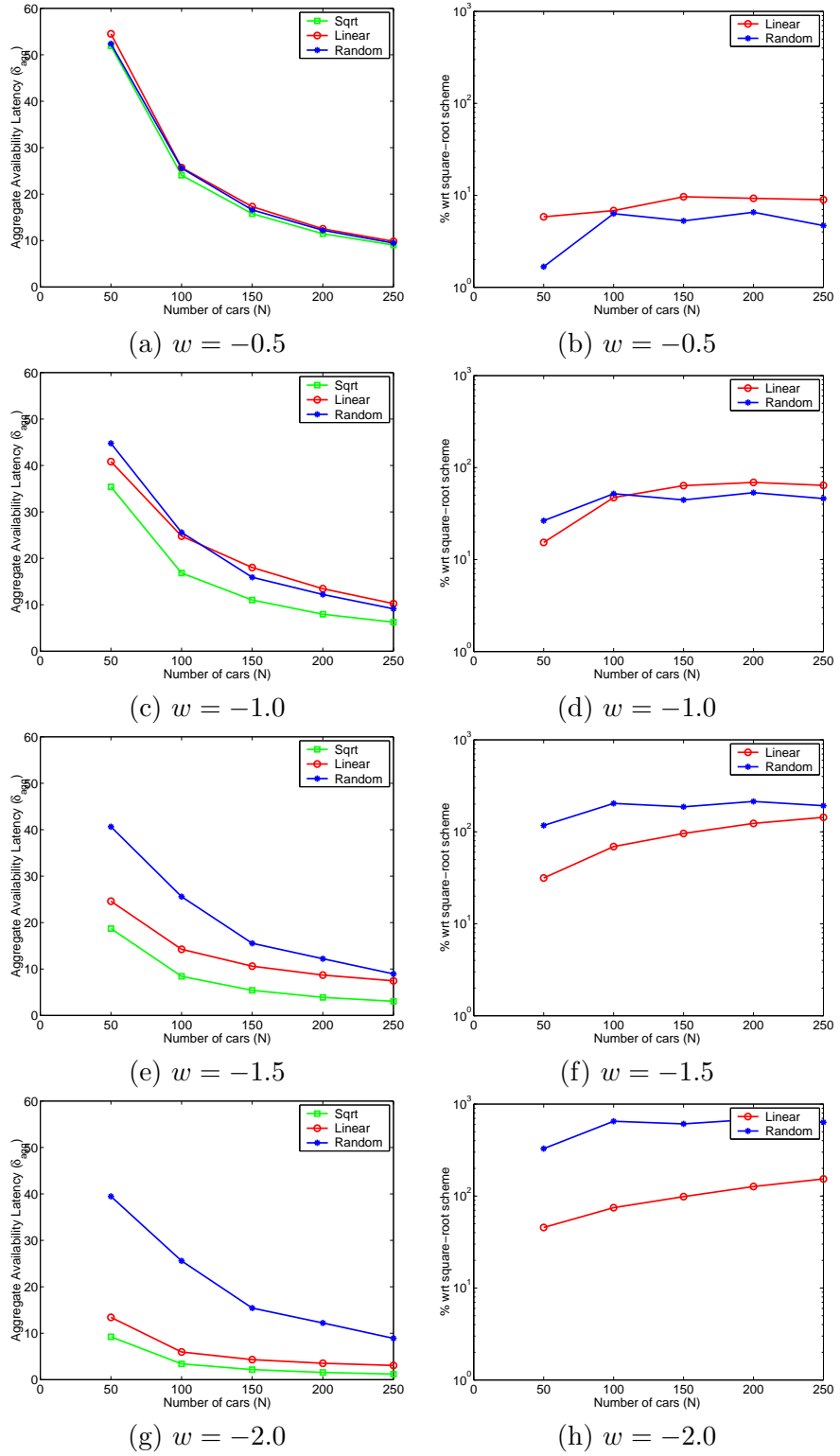


Figure 4.6: Aggregate availability latency for the three replication schemes as a function of the car density when the storage per car is fixed at 3. Here  $T = 100$ .

#### 4.2.2.3 Variation in storage per car

The total storage in the system can also be increased by keeping the car density constant and increasing the storage per car. Figure 4.7 shows the performance of the three replication schemes as a function of the storage per car when the car density is held constant at 50 for  $T = 50$ . As expected increasing storage reduces the latency for all the schemes. In case of  $w = -0.5$ , the random and linear scheme show a latency performance within 10 – 20% of the square-root scheme. However, with higher  $w$  values this difference blows up with the square-root scheme providing a much lower latency as compared to the other two.

#### 4.2.2.4 Variation in data item repository size

Finally, we consider the effect of increasing the data item repository size for a given value of car density and storage per car. Figure 4.8 depicts the latency performance of the replication schemes as a function of  $T$  when  $N = 50$  and  $\alpha = 15$  giving  $S_T = 750$ . For the same total storage, as the data item repository size increases, lesser replicas are assigned per data item, resulting in an increase in the overall availability latency. With  $w = -0.5$ , all the schemes show an almost linear increase in the latency as  $T$  increases. The increase in the latency becomes less significant with increasing skewness because enough replicas can still be assigned to the popular data items which have a major contribution to the aggregate availability latency. With  $w = -1.5$  and  $w = -2$ , the random scheme shows a step function like behavior because increase in data item repository size from 100 to 400 first causes a reduction in the replicas for the popular data items. However, further increase in  $T$  from 400 to 800 does not change the number of replicas for the popular data items causing minimal change in the aggregate availability latency values.

### 4.3 Data items with display time one and short trip duration

The analysis and simulation results presented so far assumed that once a request is issued at a client, it is willing to wait as long as it takes for its request to be satisfied. In other words, the client trip duration was assumed to be unbounded. For the specific mobility



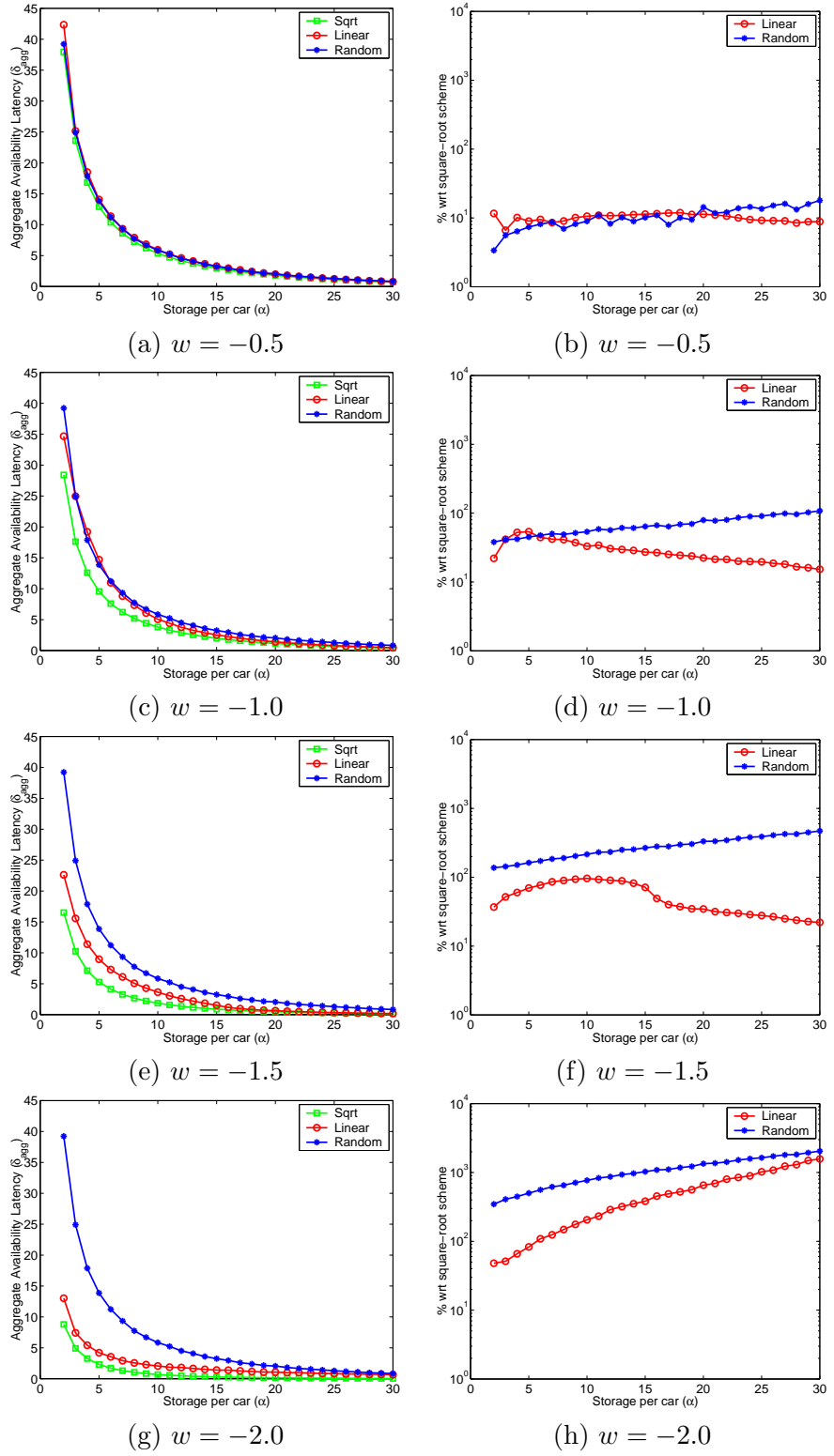
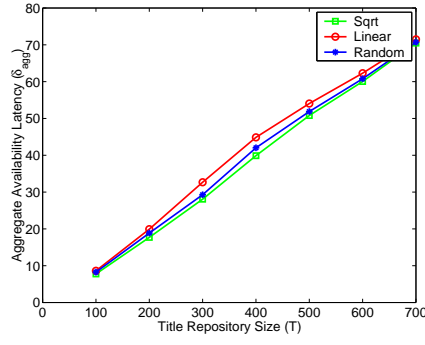
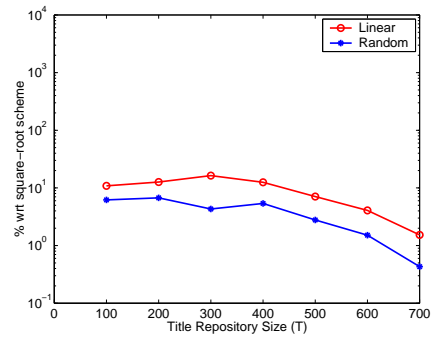


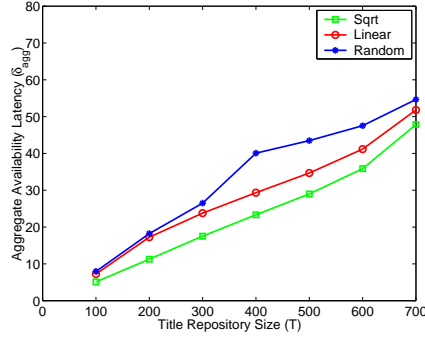
Figure 4.7: Aggregate availability latency for the three replication schemes as a function of the storage per car when data item repository size is 50 and car density is 50.



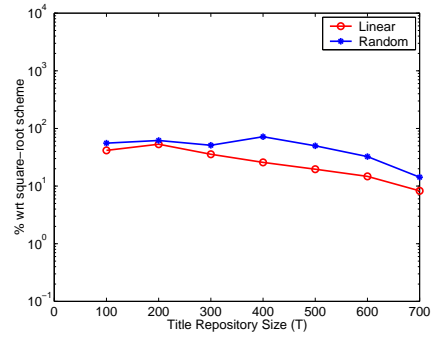
(a)  $w = -0.5$



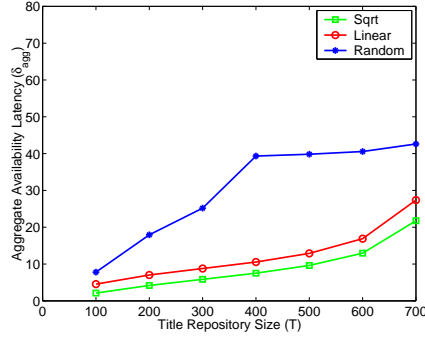
(b)  $w = -0.5$



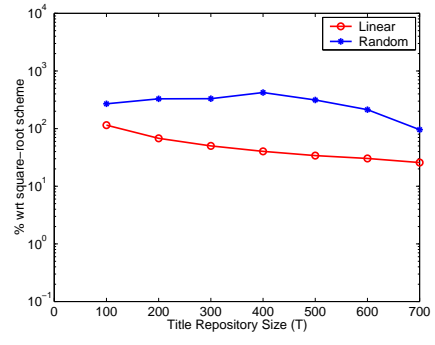
(c)  $w = -1.0$



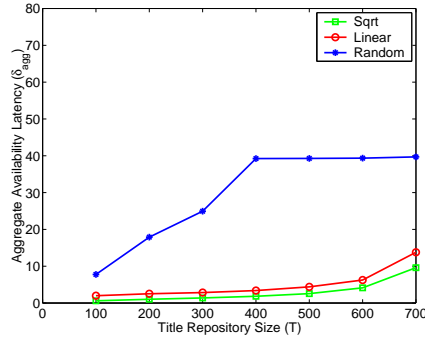
(d)  $w = -1.0$



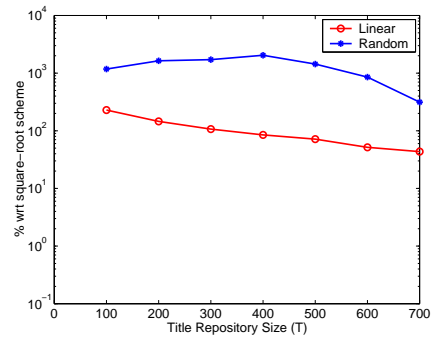
(e)  $w = -1.5$



(f)  $w = -1.5$



(g)  $w = -2.0$



(h)  $w = -2.0$

Figure 4.8: Aggregate availability latency for the three replication schemes as a function of the data item repository size for a car density of 50 and storage per car of 15.

model under consideration, namely 2D random walk on a torus, the maximum latency experienced by a client is bounded [2] as long as at least one replica of every time is present in the system at all times. However, in more practical scenarios, the client may have a certain maximum time it is willing to wait for request resolution. This is captured by considering a finite trip duration,  $\gamma$ , for the client. The availability latency for item  $i$ ,  $\delta_i$ , can be any value between 0 and  $\gamma - 1$ . If the client's request is not satisfied, we set  $\delta_i = \gamma$  indicating that the client's request for item  $i$  was not satisfied<sup>1</sup>.

### 4.3.1 Analysis

As before with Section 4.2.1, here, we present approximations for the average availability latency in the presence of a low and high density of replicas for small data items in the presence of short client trip durations.

#### 4.3.1.1 Sparse Approximation

Recall that latency in the case of a 2D-random walk on a torus can be modeled as an exponential distribution as:

$$P(\delta_i > t) = \lambda \cdot \exp(-\lambda \cdot t) \quad (4.30)$$

where  $\lambda = \frac{r_i}{c \cdot G \cdot \log G}$ . The average availability latency with finite trip duration  $\gamma$  is then given by,

$$\bar{\delta}_i = \int_0^\gamma x \cdot \lambda \cdot \exp(-\lambda \cdot t) dx + \int_\gamma^\infty \gamma \cdot \lambda \cdot \exp(-\lambda \cdot t) dx \quad (4.31)$$

Hence, we get

$$\bar{\delta}_i = \frac{c \cdot G \cdot \log G}{r_i} \cdot [1 - \exp(\frac{-\gamma \cdot r_i}{c \cdot G \cdot \log G})] \quad (4.32)$$

---

<sup>1</sup>Another way of handling finite trip durations is to divide the input requests into satisfied and unsatisfied respectively, and only calculating the expected latency for the satisfied requests. We adopt such an approach while evaluating the performance of different replication schemes using mobility derived from traces of bus movements from a real test-bed, see Section 4.7.

#### 4.3.1.2 Dense Approximation

Recall that as defined in Section 4.2.1.2,  $A_k$  is the event that a data item  $i$  is encountered by the client for the *first* time in the  $k^{th}$  cell and  $P(A_k)$  is the probability that event  $A_k$  occurs. Also,  $p_k$  is the probability of encountering data item  $i$  in the  $k^{th}$  cell, given that it was not encountered in the previous  $k - 1$  cells. Then,

$$p_k = 1 - \left(1 - \frac{1}{G - k + 1}\right)^{r_i} ; 2 \leq k \leq \gamma \quad (4.33)$$

Also, we rewrite  $P(A_k)$  incorporating the finite trip duration constraint as,

$$P(A_k) = p_k \prod_{j=1}^{k-1} (1 - p_j) ; 2 \leq k \leq \gamma \quad (4.34)$$

Let  $P(A_{\gamma+1})$  denote the probability of not encountering the data item  $i$  during the entire trip duration  $\gamma$ . Hence,

$$P(A_{\gamma+1}) = \prod_{j=1}^{\gamma} (1 - p_j) \quad (4.35)$$

Then, the availability latency ( $\delta_i$ ) for data item  $i$  is given by,

$$\bar{\delta}_i = \sum_{k=1}^{\gamma+1} (k - 1) P(A_k) \quad (4.36)$$

Figure 4.9 shows that the above approximations for low and high density of replicas matches the latency obtained by simulations. In this case, the dense approximation is also valid for a low density of replicas because the finite trip duration  $\gamma$  limits the maximum value of the availability latency. For a low density of replicas in most cases the latency will be higher than  $\gamma$  and hence it will be bounded by  $\gamma$ . For a higher replica density, the value of  $\gamma$  is not as significant since the latency for that item will be much lower than  $\gamma$ .

#### 4.3.2 Simulation Results

Figure 4.10 depicts the latency performance for different replication schemes when storage per car is increased from 4 to 25 slots when the trip duration is set as 10. When storage

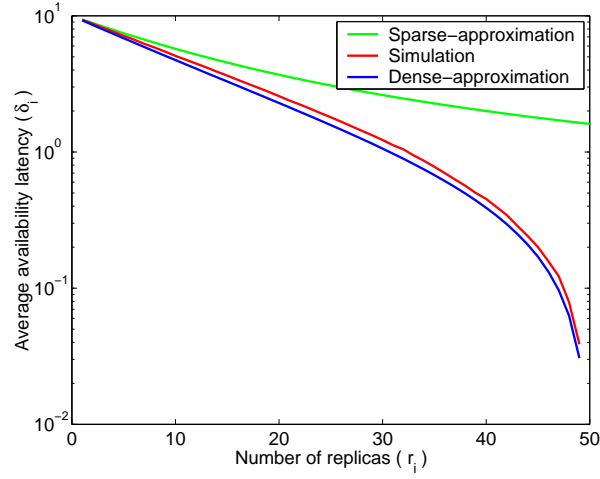


Figure 4.9: Average availability latency for a data item as a function of its replicas for a finite trip duration  $\gamma$  of 10. The simulation curves are plotted along with the sparse and dense approximations for finite trip duration for a  $10 \times 10$  torus, when the number of cars is set to 50.

per car is low,  $\alpha = 4$ , this represents a constrained storage scenario. The linear scheme that allocates more replicas to the popular data items shows superior performance as compared to the square-root scheme. This is because in such scenarios the replicas per data item is small, hence, only data items having a larger number of replicas will provide a latency less than  $\gamma$ . Since the popular data items are the ones that requested more often allocating more replicas for these items lowers the aggregate availability latency. Contrast this scenario with the case of unbounded trip duration where a square-root replication scheme always provided the minimum latency (see Figure 4.4).

The optimal scheme here is a super-linear one which allocates most of the replicas to the first few popular data items after satisfying the constraint that at least one copy of every data item must be present in the network. For a highly skewed scenario,  $w = -2$ , allocating all the remaining storage for the most popular data item minimizes the latency. This is because most of the popularity weight is associated with the most popular data item which is requested very often.

As the storage per car is increased further the curves start becoming flatter and at  $\alpha = 25$ , see Figure 4.10(c), a replication scheme characterized by an exponent in the

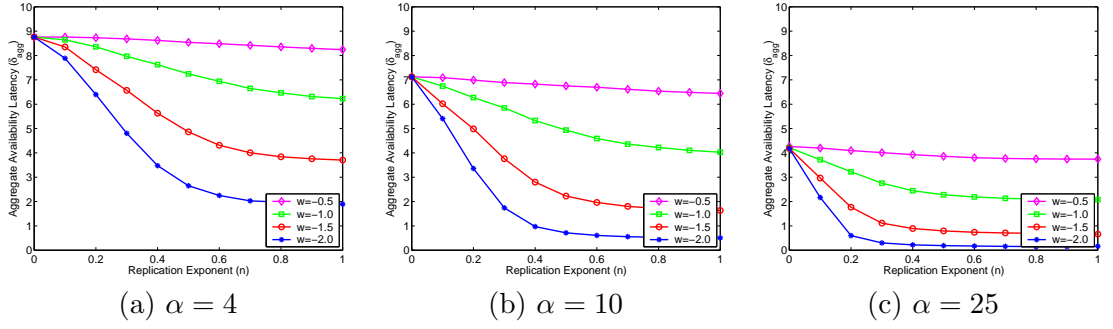


Figure 4.10: Aggregate availability latency for different replication strategies for a  $10 \times 10$  torus for a finite trip duration of 10 when  $T = 100$  and  $N = 50$ . Figures (a), (b), and (c) depict three different storage values per car:  $\{4, 10, 25\}$ .

range,  $0.3 \leq n \leq 1.0$ , shows near optimal performance. This is because the storage is abundant enough for all these schemes to allocate a copy of the popular data items to every car bringing the latency for these items to 0. The difference in the replicas allocated for the lesser popular data items has minimal effect on the aggregate availability latency on account of their lower request rate. Recall, the frequency of access to the data items follows a Zipf distribution that depicts a heavy-tailed behavior.

#### 4.4 Data items with higher display time and long client trip duration

All the results so far considered a homogeneous repository of data items with a display time ( $\Delta$ ) of one. In this section, we consider data items with a higher  $\Delta$ . In such cases, the latency encountered by a client request is given by the earliest time when a contiguous block of  $\Delta$  cells containing at least one replica of the request item is encountered. Figure 4.11 depicts the average availability latency for a data item with a higher display time ( $\Delta = \{2, 3, 4, 5\}$ ) as a function of the replicas for that item. Here, again we consider a sparse density of replicas. For a given data item replica density, the latency increases with the display time. As expected the latency reduces with increase in the replicas. A simple curve-fit on the latency curves yields a close-match with the expression of the form

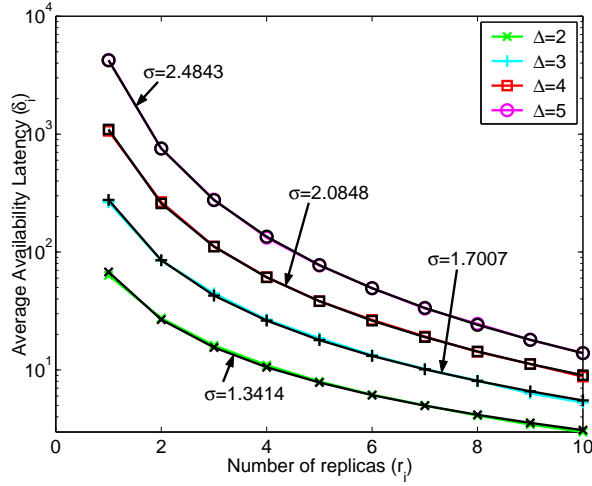


Figure 4.11: Average availability latency for a data item as a function of its replicas for different data item display times for a  $10 \times 10$  torus. The latency is given by  $\frac{C}{r_i^\sigma}$  where the exponent  $\sigma$  increases with data item display time.

$$\bar{\delta}_i = \frac{C}{r_i^\sigma} \quad (4.37)$$

where  $\sigma$  represents the exponent for a given data item display time and  $C$  is a constant that is a function of the size of the torus. Note that the value of  $\sigma$  for data items with a display time of one is one (see Equation 4.6). The values of  $\sigma$  increases with data item display time. This indicates that an increase in the replicas provides a larger drop in the latency for a data item with a higher display time. Intuitively, encountering a replica in a contiguous block of  $\Delta$  cells becomes more and more difficult as  $\Delta$  increases. Hence, an increase in the replica density provides a faster reduction in the latency for the higher  $\Delta$  items. This is captured by the increasing value of  $\sigma$  with  $\Delta$ .

The specific formulation of Equation 4.37 has special significance. Equation 4.37 can be plugged in directly into the optimization formulation in Section 4.2.1.1 to determine the optimum replication scheme that minimizes the availability latency in case of data items with higher display times. Following a similar procedure as stated in Theorem 1, we get the following result.

$\Delta$	$\sigma$	$n = \frac{1}{\sigma+1}$
2	1.3414	0.4271
3	1.7007	0.3703
4	2.0848	0.3242
5	2.4843	0.287

Table 4.1: Approximate optimal replication exponents for data items with higher data item display times.

**Corollary 1.** *In case of a sparse density of vehicles, with a repository of data items with higher display times ( $\Delta > 1$ ), replication exponent  $n$ , such that  $n < 0.5$ , minimize the aggregate availability latency metric.*

*Proof.* Following a similar procedure as the proof listed in Theorem 1, we obtain the optimal replication exponents for the  $\sigma$  values capturing higher data item display times in Figure 4.11. Table 4.1 lists the display times and the corresponding approximate optimal exponent values.  $\square$

## 4.5 Data items with higher display time and short client trip duration

In this section, we consider scenarios with short client trip durations with data items having higher display times. This implies that the client is only willing to wait for a short period for its request to be satisfied (denoted by  $\gamma$ ). Otherwise the request is tagged with a latency equal to  $\gamma$ .

Figure 2.2 shows a  $6 \times 6$  grid used as the map for our experimental study. Assume that the client starts from cell 1 and travels along the path  $\{1, 8, 15, 22, 29, 36\}$ . Numbers in the bracket indicate the sequence of visited cell IDs. Hence,  $\gamma = 6$ . For our experiments, we chose  $N = 200$  and  $T = 100$ . We simulated a skewed distribution of access to the  $T$  data items using a Zipf distribution with a mean of 0.27. The distribution is shown to correspond to sale of movie theater tickets in the United States [15].

Initially, all cars are distributed across the cells of the map as per the steady state distribution which is determined by a random number generator initialized with a seed.



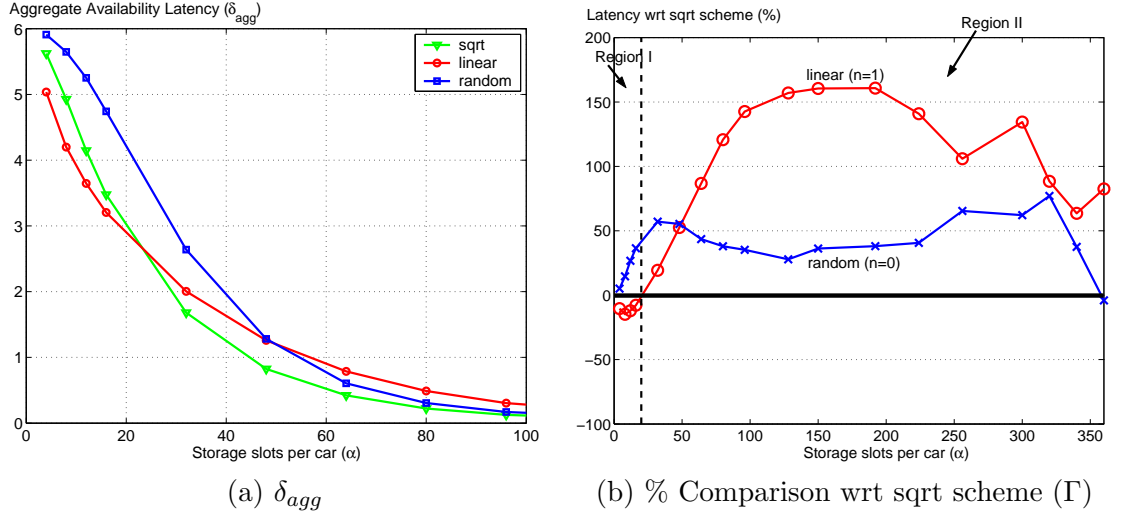


Figure 4.12: Figure 4.12(a) shows  $\delta_{agg}$  of the sqrt, linear and random replication schemes versus  $\alpha$  for  $\Delta = 4$  and  $N = 200$ . Figure 4.12(b) shows the % comparison of the linear and random schemes wrt the sqrt scheme for this scenario. Region I and Region II, respectively, indicate the parameter space where  $n = 1$  and  $n = 0.5$  perform the best.

Depending on the particular replication technique, the replicas for each data item are calculated using Equation 4.2 and then distributed across the car. A car only contains a maximum of one replica for a particular data item. The distribution of data item replicas across the cars is uniform. At each step, depending on the current car location, it moves to one of its adjoining cell (including itself) as governed by the mobility model. Another seed determines the choice of which cell a car moves to. Since  $\gamma = 6$ , each car performs six transitions according to the mobility model. We performed the comparisons for several different data item distribution seeds starting from the same initial car positions. Next, we varied the initial car positions by changing the initial seed. Specifically, we chose 50 different initial seeds and for each of these we used 50 seeds that decide the distribution of the data item replicas among the cars. Thus, each point in all the presented results is an average of 2500 simulations.

Below is an overview of the key lessons learnt from these experiments with higher data item display times and a short trip duration:

- The optimal value of  $n$  varies as a function of the scarcity of the network storage

- When storage is scarce, the optimal aggregate availability latency is realized by using a higher value of  $n$ .
- Even a random scheme with  $n = 0$  is good enough when storage is abundant relative to the repository size.

When storage is extremely scarce, with larger data item sizes ( $\Delta > 1$ ), linear ( $n = 1$ ) scheme provides the best performance. This is because it allocates more replicas for the popular data items at the cost of assigning very few for the remaining data items. In this case, the contribution to  $\delta_{agg}$  is a function of the  $\delta$  for the more popular data items since for the less popular data items there will be insufficient replicas to reduce their  $\delta$ . On the other hand, since the random scheme is blind to the data item access frequencies, on an average, it assigns equal number of replicas for each data item thereby providing the worst performance.

The square root ( $n = 0.5$ ) scheme assigns fewer replicas for the popular data items than the linear scheme. As we increase the amount of storage, there is a cut-off point along the storage axis, where allocating more replicas for the popular data items provides negligible improvement in  $\delta_{agg}$ . It is beyond this point that the square root scheme starts outperforming the linear scheme. This is because the square root scheme can use the extra storage savings for allocating replicas for the less popular data items, thereby reducing their  $\delta$ .

To illustrate, Figure 4.12 shows the variation of  $\delta_{agg}$  as a function of  $\alpha$  for  $\Delta = 4$ . Since  $\delta_{agg}$  is a function of the value of  $n$ , hence, here we denote it as  $\delta_{agg}(n = i)$ . For Figure 4.12(b), the y-axis represents the percentage comparison of the linear ( $n = 1$ ) and the random ( $n = 0$ ) schemes with respect to the square root ( $n = 0.5$ ) scheme calculated as,

$$\Gamma = \left( \frac{\delta_{agg}(n = i) - \delta_{agg}(n = 0.5)}{\delta_{agg}(n = 0.5)} \right) \times 100; \text{ where } i = \{0, 1\} \quad (4.38)$$

Figure 4.12(b) shows two distinct regions in which the schemes with  $n = 0.5$  and  $n = 1$  perform well under certain parameter settings within the design space. For  $\alpha \leq 20$ , the linear scheme ( $n = 1$ ) performs the best. For  $20 \leq \alpha \leq 360$ , the square root scheme

( $n = 0.5$ ) performs the best. Beyond this value even a random scheme ( $n = 0$ ) provides a competitive latency performance.

With  $\Delta = x$  and  $T = y$ , the value of  $\alpha$  needed to replicate the entire database on each car is  $\alpha_{db} = x \cdot y$ . At a certain storage threshold (earlier than  $\alpha_{db}$ ), the random scheme assigns enough replicas to the popular data items to bring their  $\delta$  down. In this case, all the data items have the same number of replicas, thereby producing a low  $\delta$  for every data item. Hence, from this point onward, even a random scheme provides a good performance. However, this point requires sufficient storage per car and hence a random scheme may be appropriate only for over-provisioned scenarios. As illustrated in Figure 4.12(b) with  $N = 200$ ,  $T = 100$ , and  $\Delta = 4$ , the storage threshold is around 360 slots per car. For  $\Delta = 5$ , and 6, this threshold is approximately 450 and 540, respectively. These are loose upper bounds.

#### 4.5.1 Aggregate availability latency as a function of car density ( $N$ )

Car density, which in turn affects the available storage in the system, has a major impact on the performance of  $\delta_{agg}$  for all the schemes. With the decrease in the car density to  $N = 100$ , the number of replicas allocated by the schemes is reduced thereby giving comparatively larger values of  $\delta_{agg}$  across the same storage axis. As  $\alpha$  is increased, the drop in the  $\delta_{agg}$  curves for all the schemes is not as steep as seen in the case with  $N = 200$ . Again, this is because the number of replicas is not increasing at such a high rate. The storage is reduced by an order of 2, hence a higher value of  $\alpha$  is needed to produce the same drop in  $\delta_{agg}$  as was seen in the case with  $N = 200$  cars. This is observed across all values of  $\Delta$ .

For all experiments, we also calculated the standard deviations (SD) and the standard error of the mean (SEM). The 95% confidence intervals determined as  $1.96 * SEM$  are quite small and accordingly the curves are quite smooth. However, the standard deviation is quite large, especially for the cases when  $\delta_{agg}$  is low for high values of  $\Delta$  and  $\alpha$ . This is because a low latency requires the data item to be present in every cell along the journey depending on the value of  $\Delta$ . As  $\Delta$  increases, it becomes increasingly difficult to meet

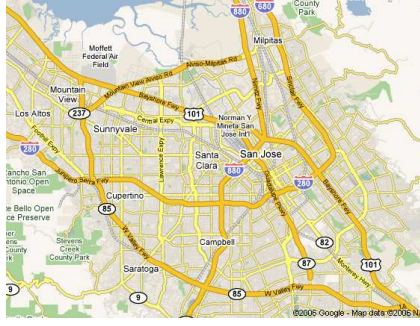
this condition thereby showing a high variance in  $\delta_{agg}$ . The large SD value is an empirical observation about the nature of the random process.

Here, we summarize the main results obtained so far with the 2D random walk based mobility model. For data items with a display time of one and long client trip duration, we have the following: in case of sparse density of data item replicas, we analytically solved storage constrained optimization to minimize the availability latency, yielding that a replication scheme that allocates replicas for a data item as per the square-root of its frequency of access provides the optimal aggregate availability latency. With a higher density of replicas, a scheme that allocates replicas as the  $n^{th}$  root of the data item access frequencies where  $0.4 \leq n \leq 0.6$  provides near optimal latency. In all cases, we presented the relative performance of three well-known replication schemes, namely, linear (allocates replicas in direct proportion to the data items' frequencies), square-root, and random (allocates equal number of replicas per data item).

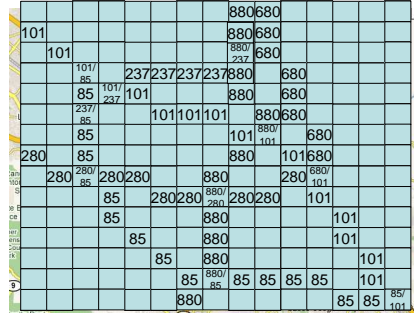
For data items with a display time of one and short client trip duration, we found that a linear replication scheme provides a superior performance. In such cases, a super-linear replication scheme that allocates a large number of replicas to the popular data items provides the optimal latency. With data items with larger display time and long client trip duration, we found that higher  $n^{th}$  root replication schemes where  $n < 0.5$ , show optimal latency performance. Finally, with a short client trip duration, we found that in extremely low storage scenarios, a linear scheme shows superior performance, in moderate to high storage scenarios a square-root replication scheme provides superior performance while in abundant storage scenarios even a random replication scheme is good enough to provide a low latency.

## 4.6 Evaluation with a real map

In this section, we describe the performance of the various replication schemes where the vehicular movements are dictated by an underlying map of the San Francisco Bay Area. Figure 4.13(a) depicts a section of the San Francisco Bay area with the major freeways and their intersections. We superimpose a 2D-grid on this map and the individual cells are labelled with the respective freeway id that they cover as shown in Figure 4.13 (b).



(a)



(b)

Figure 4.13: A map of the San Francisco Bay Area obtained from <http://maps.google.com> is shown in Figure 4.13(a). Figure 4.13(b) superimposes a  $15 \times 15$  grid on this map and labels the cells appropriately with the freeway IDs that they overlap with.

This 2D-grid serves to capture the underlying map at a coarse granularity. Most of the probability mass is concentrated on the cells that represent the major freeways. The non-labelled cells have equal transition probabilities to each of its neighboring eight cells.

The outgoing transition probabilities at a cell that represents an intersection between two freeways are calculated as follows. As an example, consider the intersection of the freeways 880 and 85 as shown in Figure 4.14. We obtained the traffic density seen on the freeways before and after the intersection from Caltrans data provided by the California department of transportation [44]. The website allowed real time gathering of vehicle traffic data. We considered a time window between 7-8 pm for a particular week and averaged the vehicular density seen during this period. The day-to-day statistics were quite similar, here, we show an example of how the actual data was converted into the probability transition values that formed the basis of the Markov mobility model. Similar calculations were employed to populate the entire transition probability matrix. Finally, to eliminate finite edge effects, we converted the  $15 \times 15$  grid into a torus by allowing cars at the boundaries to appear at the opposite ends with equal transition probabilities.

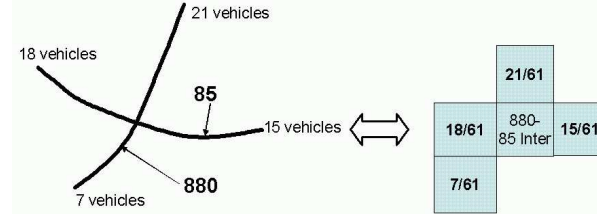


Figure 4.14: The intersection between freeways 880 and 85 is captured in the figure along with the equivalent probability transitions in the Markov model based on data obtained from Caltrans regarding the vehicular densities.

The transition matrix was used to generate the car movements. We provide a notion of directionality to the car movements by ensuring that the next step for a cars movement takes into account both the current cell as well as the previous cell which a car traversed. This is done by storing both the cell IDs as part of the state of the Markov chain. Consequently, the flip-flop movements of the cars is avoided thereby ensuring that car movements are constrained by the underlying freeway structure of the map and are not entirely random. We used these car movements to investigate the relative performance of the various replication schemes under such a scenario.

#### 4.6.1 Results with replication schemes

In this section, we present some representative results for the various replication schemes obtained by employing the Markov mobility model previously derived from a map of the San Francisco Bay area. Below we describe the three sets of experiments used in our evaluation for comparison of the linear, square-root, and random replication schemes. As before, requests are issued, one at a time at each time-step at vehicles in a round-robin manner, as per a Zipf distribution with a mean of 0.27. The three sets of experiments were:

- For data item repository size  $T$  set as 25, client trip duration,  $\gamma$ , set as 10, storage per car,  $\alpha$ , set as 2, the latency performance with the various replication schemes is studied as a function of increasing car density  $N$  (see Figure 4.15).

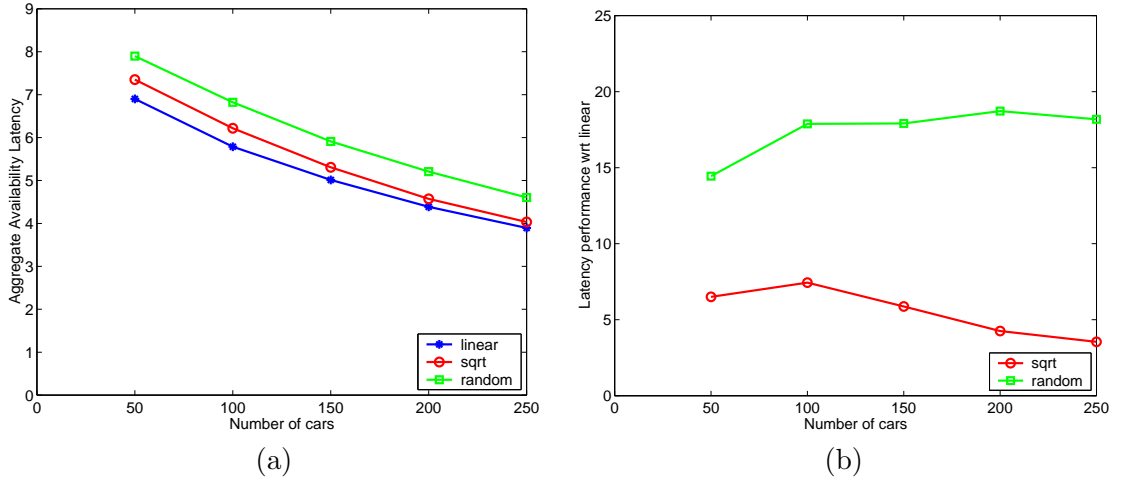


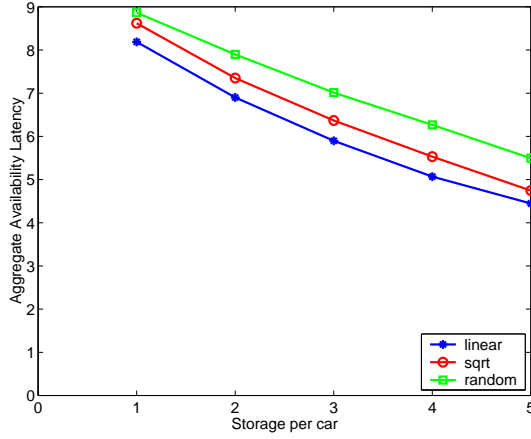
Figure 4.15: Performance of various replication schemes as a function of car density when  $T = 25$ ,  $\alpha = 2$ , and  $\gamma = 10$ . Figure 4.15(b) shows the performance wrt the linear scheme.

- For data item repository size  $T$  set as 25, client trip duration,  $\gamma$ , set as 10, car density,  $N$ , set as 50, the latency performance with the various replication schemes is studied as a function of increasing storage per car  $\alpha$  (see Figure 4.16).
- For car density,  $N$ , set as 50, client trip duration,  $\gamma$ , set as 10, storage per car,  $\alpha$ , set as 2, the latency performance with the various replication schemes is studied as a function of increasing data item repository size  $T$  (see Figure 4.17).

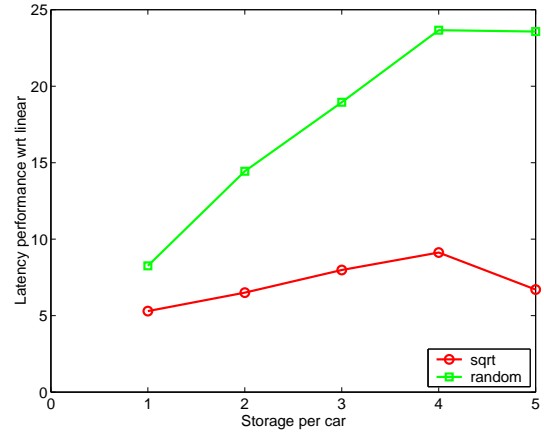
In all cases, the main conclusion is that the linear replication scheme shows superior performance as seen in Section 4.3 for the case with data item size equal to 1 and finite client trip duration. The trends seen with this model are similar to those seen with a uniform Markov mobility model with equal transition probabilities. This result suggests that the uniform probability transition matrix based Markov model may be a good indicator of the performance that may be seen with a model derived from real maps.

## 4.7 Evaluation with real movement traces

We now evaluate the latency performance of the static replication schemes using traces obtained from a bus-based DTN test-bed called UMassDieselNet [11]. First, we briefly

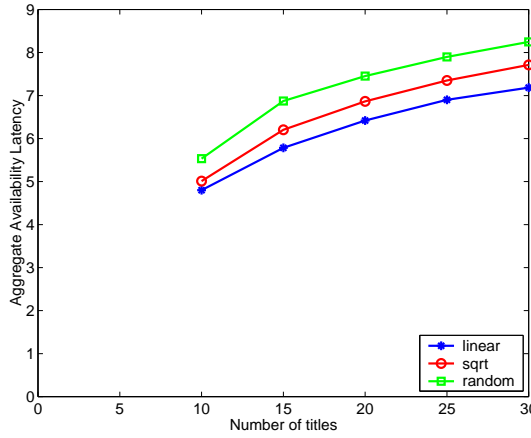


(a)

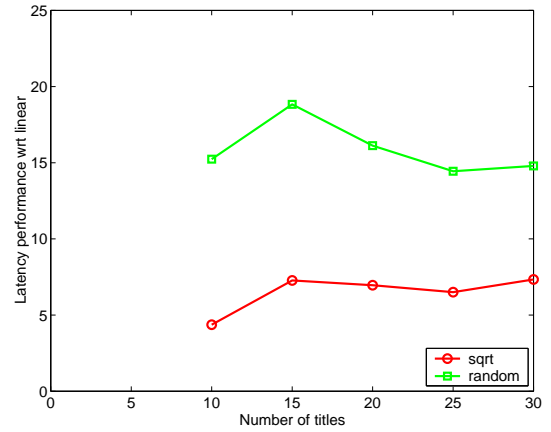


(b)

Figure 4.16: Performance of various replication schemes as a function of storage per car when  $T = 25$ ,  $N = 50$ , and  $\gamma = 10$ . Figure 4.16(b) shows the performance wrt the linear scheme.



(a)



(b)

Figure 4.17: Performance of various replication schemes as a function of data item repository size when  $N = 50$ ,  $\alpha = 2$ , and  $\gamma = 10$ . Figure 4.17(b) shows the performance wrt the linear scheme.



describe the test-bed and present some properties of the mobility model followed by the buses. Then, we describe the details of the experimental set-up and the results comparing the square-root, linear, and random replication schemes under different parameter settings using these traces.

#### 4.7.1 UMassDieselNet Traces

Here, we briefly describe the details of the UMassDieselNet test-bed and present some properties of the mobility model that characterizes the movement of the vehicles that are part of the test-bed. The UMassDieselNet network operates daily around the UMass campus and the surrounding county. It comprises of 30 buses equipped with a Linux based computer coupled with a IEEE 802.11b wireless interface that permits ad-hoc communication between the buses when they are in radio range. An IEEE 802.11b access point is also connected to the brick computer that allows DHCP access to passengers within the bus. The traces are available for a period of 60 days, the logs describe every encounter between every pair of buses that occurred during the day. These traces do not contain the logs for accesses made between the passengers and the access point within the bus. The identity of the buses involved in the encounter, the time of encounter and amount of data transmitted during the encounter are logged in the trace files. Certain buses had long routes while others had short ones. Unfortunately, due to technical difficulties, the GPS device on the buses were unable to provide details about the bus locations during the encounter.

Figure 4.18 shows the number of buses that were active on each day of the 60 day period during which the traces were collected. We only considered traces where the number of active buses was greater than 15. This accounted for 52 traces. In general, the traces indicated a sparse density of buses where there was a high degree of locality in the encounters. In other words, if 2 buses encounter each other at the beginning then they will continue to encounter each other more frequently than other buses. This is captured in Figure 4.19 where we set a minimum separation time between two consecutive encounters of the same pair of buses to consider it a different encounter. In Figure 4.19(b), the separation time is set as 20 seconds as compared to 0 seconds set for Figure 4.19(a).

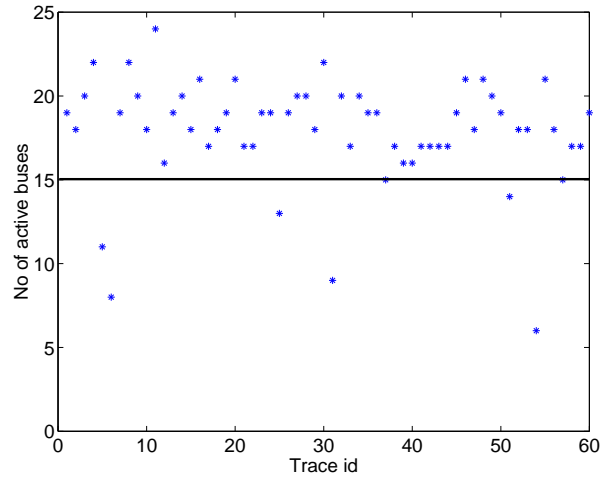


Figure 4.18: The number of active buses for each trace representing the bus encounters for each day of a 60-day period. The buses operated from 7am to 5pm.

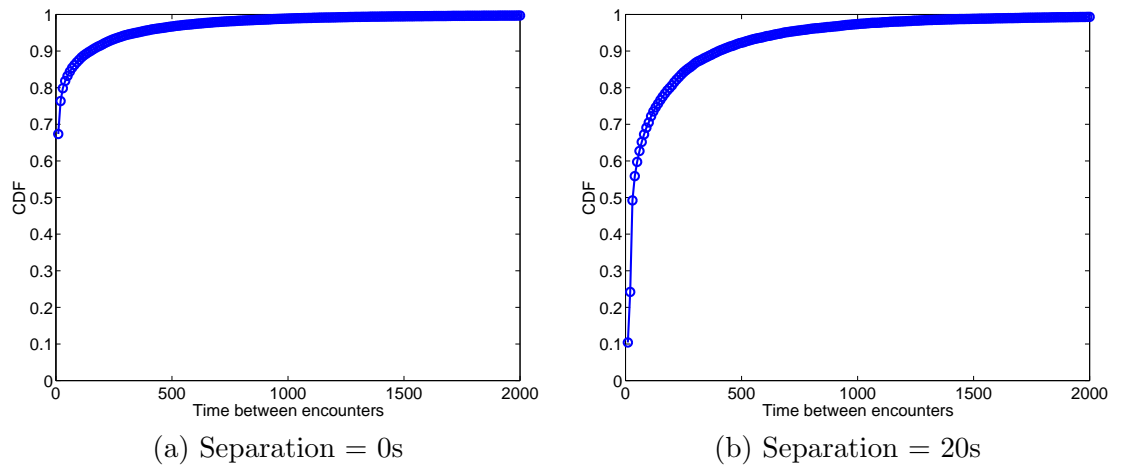


Figure 4.19: The CDF of the time between encounters averaged across all the traces for 2 different separation times 0s (Figure 4.19(a)) and 20s (Figure 4.19(b)).

### 4.7.2 Experimental Set-up

In this section, we describe the details of the simulation set-up used for evaluation of the replication schemes employing the UMassDieselNet traces. Each trace represents the movements of buses during that particular day. There is no correlation between trace movements across days. Hence, we process each trace one at a time and then average the results observed across all the days noting that the average is indicative of the performance seen on most days. However, certain days do appear as outliers since the number of active buses differs from day-to-day.

As before we consider a finite data item repository of size  $T$ . Each bus is assumed to carry  $\alpha$  storage slots. Replicas for each data item are determined based on a replication scheme and then allocated across the buses uniformly at random. The constraint is that at least one copy of every data item must be present in the network at all times. We consider the three representative replication schemes: random, square-root, and linear and study the relative performance of the schemes.

Since the buses only operate for a finite amount of time we consider two separate metrics (i) Average availability latency for satisfied requests (ii) Normalized unsatisfied request rate. Requests for the  $T$  titles are generated as per a Zipf distribution with an exponent  $w = -0.73$ . The duration during which the buses were active during a day is determined apriori and subject to this duration requests are issued at equal inter-arrival times. A generated request is assigned to a bus chosen uniformly at random. A request is assumed to be satisfied either if the data item requested is locally stored or another bus carrying the requested item is encountered at some point after the request is issued. Those requests that are not satisfied at the end of the day are tagged as unsatisfied requests.

### 4.7.3 Results

In this section, we briefly describe the main results from evaluation of the performance of the replication schemes using the UMassDieselNet traces. For the first set of experiments we vary the values of  $(T, \alpha)$  as  $\{(5,1), (10,2), (15,3), (20,4), (25,5)\}$  (see Figure 4.20). The linear replication scheme provides the lowest average availability latency for satisfied

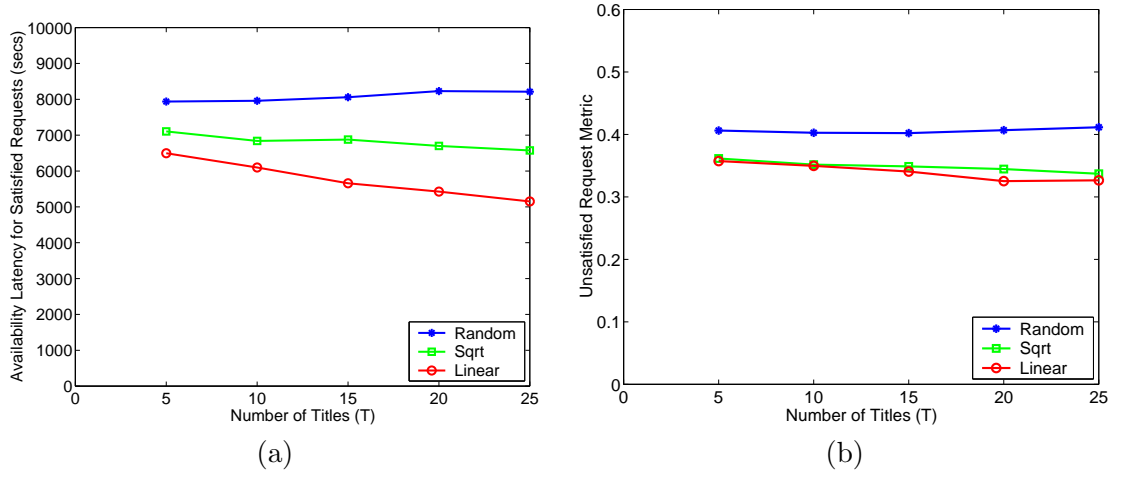


Figure 4.20: Aggregate availability latency for satisfied requests and the aggregate unsatisfied request metric for the random, square-root, and linear replication schemes are shown in Figure 4.20(a) and (b) respectively. The ratio of the storage per car to the data item repository size,  $\frac{\alpha}{T}$  is maintained as 1:5.

requests (about 10 – 25% better than the square-root scheme). The linear and square-root scheme show similar performance in terms of the normalized unsatisfied requests. The random scheme shows poor performance both in terms of latency as well as the normalized unsatisfied requests.

Figure 4.21 shows the performance of the replication schemes when the data item repository size is fixed at 25 and the storage per bus is increased. Increase in storage leads to increase in the replicas per data item, hence, as expected for all schemes, the latency and the normalized unsatisfied requests go down. The linear scheme continues to show superior performance with respect to both metrics.

Figure 4.22 shows the performance of the replication schemes when the storage per bus is fixed at 3 and the data item repository size is increased. As the data item repository size is increased, lesser replicas are allocated per data item resulting in an increase in the latency as well as the unsatisfied requests. Initially the increase in latency is linear but slowly becomes sub-linear. Note that the data item repository size cannot be made arbitrarily large since the number of active buses is constrained. As before the linear scheme always shows the best performance.

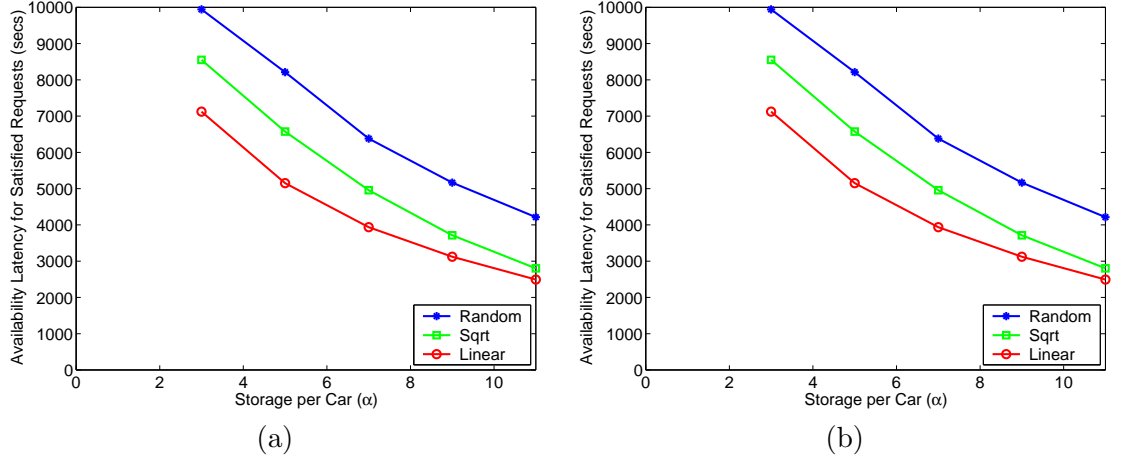


Figure 4.21: Aggregate availability latency for satisfied requests (Figure 4.21(a)) and the aggregate unsatisfied request metric (Figure 4.21(b)) as a function of the storage per car for a data item repository size of 25.

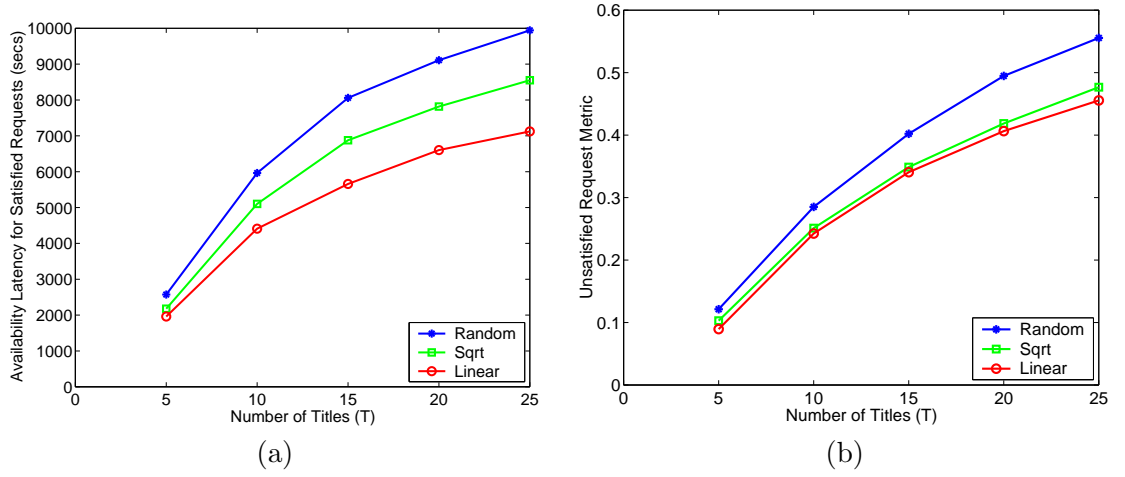


Figure 4.22: Aggregate availability latency for satisfied requests (Figure 4.22(a)) and the aggregate unsatisfied request metric (Figure 4.22(b)) as a function of the data item repository size when storage per car  $\alpha$  is fixed at 3.

The mobility model provided by the traces represents an extremely sparse density of buses where inherently there is a limit to the maximum amount of time for request satisfaction (namely the last encounter time on the trace). The finite trip duration in conjunction with the low density and encounter model favors a linear scheme which allocates more replicas for the popular data items. The popular data items are requested more frequently, and within the finite time for request satisfaction, have a higher probability of being satisfied on account of the larger number of replicas. The square-root scheme tries to allocate replicas less aggressively to the more popular data items in favor of the less popular ones. This hurts its performance since the less popular data items have a very low probability of being satisfied. Nevertheless, in such scenarios, a random scheme that allocates replicas equally across the data items shows the worst performance.

We now consider an equivalent scenario with the Markov mobility and study its properties in terms of the time between encounters (see Figure 4.23). The aim is to capture a similar scenario as depicted by the UMassDieselNet traces (compare with Figure 4.19(a)). We consider a similar set-up to experimental scenario described in Figure 4.20. Similar to the trends seen with the traces, Figure 4.24 shows that the linear replication scheme outperforms the square-root and the random schemes in terms of the latency for satisfied requests. The performance in terms of the normalized unsatisfied requests is quite similar for the three schemes. These results suggest that the results obtained from the Markov mobility model may be applicable across a vast range of scenarios comprising different mobility models. Adequate adjustment to the transition probabilities of the Markov model may enable this model to suitably capture the mobility trends of other models such as Manhattan, Highway, Random Way-Point etc.

## 4.8 Summary

In this chapter, we have investigated the performance of various replication schemes for a mobile ad hoc network of AutoMata devices. These schemes compute the degree of replication for each data item as a power law function of its popularity, i.e., frequency of access. We propose a general optimization formulation to minimize the average availability latency subject to a total constraint enforced by the car density and the storage per

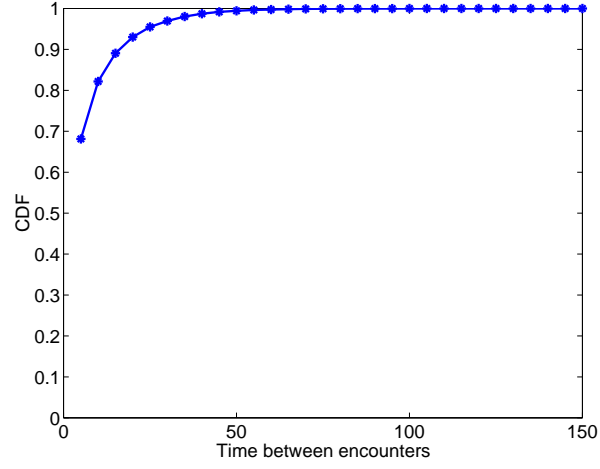


Figure 4.23: CDF of the time between encounters from the Markov model for a  $25 \times 25$  torus with a car density of 15.

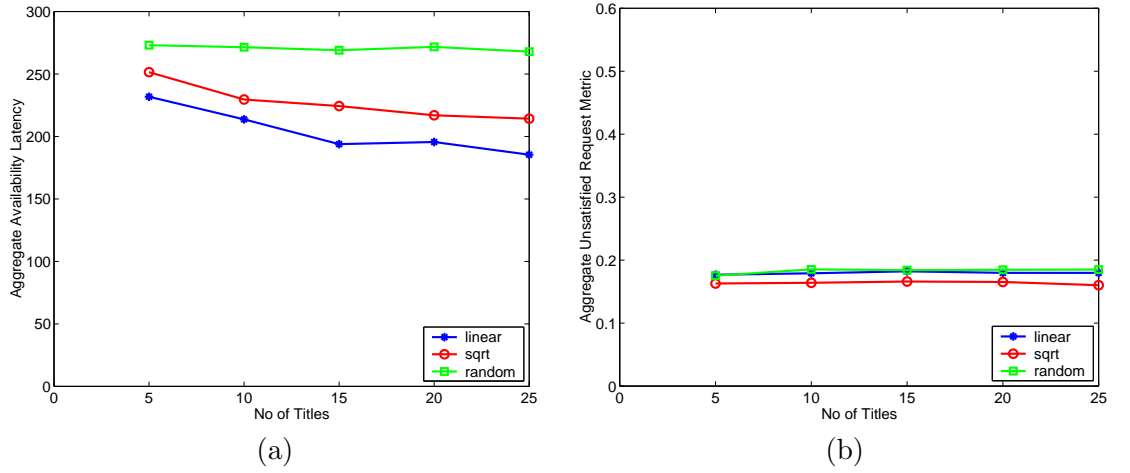


Figure 4.24: Aggregate availability latency for satisfied requests and the aggregate unsatisfied request metric as obtained from an equivalent scenario employing the Markov model. The ratio of the storage per car to the data item repository size,  $\frac{\alpha}{T}$  is maintained as 1:5.

car. While it may be possible to solve this optimization under some scenarios, because of the large parameter design space we solve the problem indirectly using simulations. Obtained results indicate the following key lessons:

- For a repository with data items having size 1 and with unbounded client trip duration, a square-root replication scheme minimizes availability latency
- For a repository with data items having size 1 and with finite client trip duration, linear/super-linear replication schemes show superior performance
- For a repository with data items having size  $> 1$  and with unbounded client trip duration, higher  $n^{th}$  order replication schemes minimize the aggregate availability latency where  $n > 2$  (see Equation 4.1)
- For a repository with data items having size  $> 1$  and with finite client trip duration, a linear scheme shows superior performance for highly constrained storage scenarios while a square-root scheme shows superior performance over a large storage space.

Obtained results are validated with Markov models derived from different maps of cities such as San Francisco that constraint the movements of the vehicles on the basis of the underlying freeway structure. We also investigate the performance of the various replication schemes with traces obtained from a bus-based DTN test-bed called UMass-DieselNet, also providing the equivalence between the trace-based mobility model and the Markov model.

A promising future research direction that will complement the above results is to explore heterogeneity. While this study has considered a homogeneous repository of data items, it may be useful to extend our evaluation to a heterogeneous repository of items with different display times and sizes. Moreover, the vehicles themselves may have different quantities of available storage and different trip durations.



## Chapter 5

### Zebroids

The choice of a replication scheme has a significant impact on the availability latency experienced by the client requests. In particular, we saw that a square-root replication scheme provides the minimum aggregate availability latency in case of unbounded trip duration, while a linear replication scheme provides superior performance for finite trip durations. We now try to answer the following question: Can the availability latency be improved further after an appropriate static replication scheme has computed the replicas per data item? Recall that the replication schemes mentioned earlier do not consider placement of the replicas. Hence, a system may dynamically reorganize the replicas across the vehicles on the basis of the currently active requests to make better use of the available system storage. Specifically, appropriate vehicles may be scheduled to carry a data item on behalf of a server (vehicle containing the requested item) to a client requesting it, thereby reducing its latency. These data carriers are labeled **zebroids**. In this chapter, we quantify the improvements in latency that can be obtained by employing these data carriers along with the incurred overhead.

The organization of this chapter is as follows. Section 5.1 formally defines a zebroid and gives a brief overview. Section 5.2 describes how the zebroids may be employed. Section 5.3 briefly describes the replacement policies that may be employed by a zebroid. Section 5.4 provides details of the analysis methodology employed to capture the performance with zebroids. Section 5.5 describes the details of the simulation environment used for evaluation. Section 5.6 enlists the key questions examined in this study and answers them via analysis and simulations. Section 5.7 presents some representative results

with zebroids when employing a Markov mobility model for the vehicles that adheres to the location of major freeways in the San Francisco Bay area. Section 5.8 provides an evaluation of the performance of zebroids on traces obtained from a real DTN test-bed comprising 30 – 40 buses. Finally, Section 5.9 provides brief conclusions of the study.

## 5.1 Overview of Zebroids

In this section, we provide a brief overview of the use of zebroids as data carriers. Selection of zebroids is facilitated by the two-tiered architecture. The control plane enables centralized information gathering at a dispatcher present at a base-station<sup>1</sup>. Some examples of control information are currently active requests, travel path of the clients and their destinations, and paths of the other cars. For each client request, the dispatcher may choose a set of  $z$  carriers that collaborate to transfer a data item from a server to a client (**z-relay zebroids**). Here,  $z$  is the number of zebroids such that  $0 \leq z < N$ , where  $N$  is the total number of cars. When  $z = 0$  there are no carriers, requiring a server to deliver the data item directly to the client. Otherwise, the chosen relay team of  $z$  zebroids hand over the data item transitively to one another to arrive at the client, thereby reducing availability latency (see Section 5.2 for details). To increase robustness, the dispatcher may employ multiple relay teams of  $z$ -carriers for every request. This may be useful in scenarios where the dispatcher has lower prediction accuracy in the information about the routes of the cars. Finally, storage constraints may require a zebroid to evict existing data items from its local storage to accommodate the client requested item. Hence, suitable replacement policies may be employed by a zebroid.

In this chapter, we quantify the following main factors that affect availability latency in the presence of zebroids: (i) data item repository size, (ii) car density, (iii) storage capacity per car, (iv) client trip duration, (v) replacement scheme employed by the zebroids, and (vi) accuracy of the car route predictions. For a significant subset of these factors, we address some key questions pertaining to use of zebroids both via analysis and extensive simulations.

---

<sup>1</sup>There may be dispatchers deployed at a subset of the base-stations for fault-tolerance and robustness. Dispatchers between base-stations may communicate via the wired infrastructure.

## 5.2 Solution Approach

When a client references a data item missing from its local storage, the dispatcher identifies all cars with a copy of the data item as servers. Next, the dispatcher obtains the future routes of all cars for a finite time duration equivalent to the maximum time the client is willing to wait for its request to be serviced. Using this information, the dispatcher schedules the quickest delivery path from any of the servers to the client using any other cars as intermediate carriers namely zebroids. Hence, it determines the optimal set of forwarding decisions that will enable the data item to be delivered to the client in the minimum amount of time. Note that the latency along the quickest delivery path that employs a relay team of  $z$  zebroids is similar to that obtained with epidemic routing [59] under the assumptions of infinite storage and no interference.

A modified version of the Bellman Ford's shortest path algorithm is employed to determine this path along the encounter graph of the cars. In the following, we give a brief description of this algorithm.

A client waits for a maximum of  $\gamma$  time steps for each issued request to be satisfied. Hence, we construct graph  $G = (V, E)$ , the encounter graph of the cars for a given request where  $V$  is the set of all cars ( $|V| = N$ ) and  $E$  is the set of edges  $\{(u, v, t) \text{ such that } 0 \leq t \leq \gamma\}$ . An edge  $(u, v, t)$  exists in  $G$  only if cars  $u$  and  $v$  encounter each other at time step  $t$  and  $u$  and  $v$  are non-servers. For a given data item request, the dispatcher identifies the set of cars that have a copy of this item as servers. Now the quickest path from each of these servers to the client is determined individually using the following algorithm listed below:

Here  $s$  is the client car,  $\text{parent}[v]$  stores the car id that handed a copy of the requested data item to car  $v$ , and  $d[s]$  stores the length of the quickest path to the client. Finally, the quickest path from all of these is selected as the minimum delay path to the client. Each car along this path except the server and the client represent a  $z$ -relay zebroid. The complexity of this modified Bellman Ford algorithm is  $O(N^3 \cdot \gamma)$ . However, in actual practice the complexity is much lower since on an average not all  $N$  cars encounter each other. Moreover, with a random walk like mobility model, cars that meet once may meet quite often due to the locality property.

**Algorithm 5.2.1:**  $\text{QUICKESTPATH}(V, E, s)$ 

```

for each  $v$  in  $V[G]$ 
     $d[v] = \infty$ 
     $parent[v] = NIL$ 
     $prevEncounterTime[v] = -1$ 
 $d[s] = 0$ 
for  $i \leftarrow 1$  to  $|V[G]| - 1$ 
    for each edge  $(u, v)$  in  $E[G]$ 
        for  $t \leftarrow 1$  to  $\gamma$ 
            if  $(d[v] > t) \& (prevEncounterTime[v] \leq t)$ 
                 $d[v] = t$ 
                 $parent[v] = u$ 
                 $prevEncounterTime[v] = t$ 

```

We use the above algorithm in sparse scenarios when  $N$  is small ( $N \leq 100$ ). In other cases, we simulate an epidemic routing kind of dissemination for the trip duration number of steps. Then we determine the earliest time the client car was infected, if at all, this establishes the minimum latency encountered by a client. Since each car stores the parent car that handed over the requested item to it, the sequence of cars involved in carrying the data item from a server to the client along the minimum latency path can be determined as a simple look-up operation.

If we restrict the length of the quickest path to a maximum of 2 hops, with the additional restriction that the first edge  $(u, v, t)$  along this path should have  $t = 1$ , then the zebroid used becomes a **one-instantaneous zebroid** with  $z = 1$ . Recall that  $u$  will be a server and  $t = 1$  signifies the instantaneous transfer of the data item from  $u$  to  $v$ , the zebroid. The zebroid then meets the client directly thereby satisfying the request.

In some cases, the dispatcher might have inaccurate information about the routes of the cars. Hence, a zebroid scheduled on the basis of this inaccurate information may not rendezvous with its target client. To minimize the likelihood of such scenarios, the

dispatcher may schedule multiple zebroids. This may incur additional overhead due to redundant resource utilization to obtain the same latency improvements.

The time required to transfer a data item from a server to a zebroid depends on its size and the available link bandwidth. With small data items, it is reasonable to assume that this transfer time is small, especially in the presence of the high bandwidth data plane. Large data items may be divided into smaller chunks enabling the dispatcher to schedule one or more zebroids to deliver each chunk to a client in a timely manner. This remains a future research direction.

Initially, number of replicas for each data item are computed as per a static replication scheme (see Chapter 4). This scheme computes the number of data item replicas as a function of their popularity. It is static because number of replicas in the system do not change and no replacements are performed. Hence, this is referred to as the ‘no-zebroids’ environment. We quantify the performance of the various replacement policies with reference to this base-line that does not employ zebroids.

One may assume a cold start phase, where initially only one or few copies of every data item exist in the system. Many storage slots of the cars may be unoccupied. When the cars encounter one another they construct new replicas of some selected data items to occupy the empty slots. The selection procedure may be to choose the data items uniformly at random. New replicas are created as long as a car has a certain threshold of its storage unoccupied. Eventually, a majority of the storage capacity of a car will be exhausted.

### 5.3 Carrier-based Replacement policies

In this section, we briefly describe the different replacement schemes that may be employed by a zebroid. The replacement policies considered here are reactive since a replacement occurs only in response to a request issued for a certain data item. When the local storage of a zebroid is completely occupied, it needs to replace one of its existing items to carry the client requested data item. For this purpose, the zebroid must select an appropriate candidate for eviction. This decision process is analogous to that encountered in operating system paging where the goal is to maximize the cache hit ratio to

prevent disk access delay. We present below a list of carrier-based replacement policies employed in our study which are adapted from different page replacement policies used in operating systems [57].

1. **Least recently used (LRU)** LRU-K [45] maintains a sliding window containing the time stamps of the  $K^{th}$  most recent references to data items. During eviction, the data item whose  $K^{th}$  most recent reference is furthest in the past is evicted. Here, we consider the case with  $K = 1$ . Depending on whether the evictions are based on the least recently used data item across all client requests (**lru-global**) or only the individual client's requests (**lru-local**), we consider global or local variants of the LRU policy.
2. **Least frequently used (LFU)** (a) **Local (lfu-local)**: Each AutoMata keeps track of the least frequently used data item within its local repository. During eviction<sup>2</sup>, this is the candidate replica that is replaced. (b) **Global (lfu-global)**: The dispatcher maintains the frequency of access to the data items based on requests from all clients. When a zebroid contacts the dispatcher for a victim data item, the dispatcher chooses the data item with the lowest frequency of access.
3. **Random policy (random)** In this case, the chosen zebroid evicts a data item replica from its local storage chosen uniformly at random.

The replacement policies incur the following overheads. First, the complexity associated with the implementation of a policy. Second, the bandwidth used to transfer a copy of a data item from a server to the zebroid. Third, the average number of replacements incurred by the zebroids. Note that in the no-zebroids case neither overhead is incurred.

The metrics considered in this study are aggregate availability latency,  $\delta_{agg}$ , percentage improvement in  $\delta_{agg}$  with zebroids as compared to the no-zebroids case and average number of replacements incurred per client request which is an indicator of the overhead incurred by zebroids.

---

<sup>2</sup>The terms eviction and replacement are used interchangeably.

Note that the dispatchers with the help of the control plane may ensure that no data item is lost from the system. In other words, at least one replica of every data item is maintained in the ad-hoc network at all times. In such cases, even though a car may meet a requesting client earlier than other servers, if its local storage contains data items with only a single copy in the system, then such a car is not chosen as a zebroid.

## 5.4 Analysis Methodology

Here, we present the analytical evaluation methodology and some approximations as closed-form equations that capture the improvements in availability latency that can be obtained with both one-instantaneous and z-relay zebroids. First, we present some preliminaries of our analysis methodology.

- Let  $N$  be the number of cars in the network performing a 2D random walk on a  $\sqrt{G} \times \sqrt{G}$  torus. An additional car serves as a client yielding a total of  $N + 1$  cars. Such a mobility model has been used widely in the literature [55, 53] chiefly because it is amenable to analysis and provides a baseline against which performance of other mobility models can be compared. Moreover, this class of Markovian mobility models has been used to model the movements of vehicles [6, 50, 66].
- We assume that all cars start from the stationary distribution and perform independent random walks. Although for sparse density scenarios, the independence assumption does hold, it is no longer valid when  $N$  approaches  $G$ .
- Let the size of data item repository of interest be  $T$ . Also, data item  $i$  has  $r_i$  replicas. This implies  $r_i$  cars, identified as servers, have a copy of this data item when the client requests item  $i$ .

All analysis results presented in this section are obtained assuming that the client is willing to wait as long as it takes for its request to be satisfied (unbounded trip duration  $\gamma = \infty$ ). With the random walk mobility model on a 2D-torus, there is a guarantee that as long as there is at least one replica of the requested data item in the network, the

client will eventually encounter this replica [2]. Later, we extend our analysis to consider finite trip duration  $\gamma$ .

Consider a scenario where no zebroids are employed. In this case, the expected availability latency for the data item is the expected meeting time of the random walk undertaken by the client with any of the random walks performed by the servers. Aldous *et al.* [2] show that the the meeting time of two random walks in such a setting can be modelled as an exponential distribution with the mean  $C = c \cdot G \cdot \log G$ , where the constant  $c \simeq 0.17$  for  $G \geq 25$ . The meeting time, or equivalently the availability latency  $\delta_i$ , for the client requesting data item  $i$  is the time till it encounters any of these  $r_i$  replicas for the first time. This is also an exponential distribution with the following expected value (note that this formulation is valid only for sparse cases when  $G \gg r_i$ ):  $\bar{\delta}_i = \frac{cG \log G}{r_i}$

The aggregate availability latency without employing zebroids is then this expression averaged over all data items, weighted by their frequency of access:

$$\delta_{agg}(no - zeb) = \sum_{i=1}^T \frac{f_i \cdot c \cdot G \cdot \log G}{r_i} = \sum_{i=1}^T \frac{f_i \cdot C}{r_i} \quad (5.1)$$

#### 5.4.1 One-instantaneous zebroids

Recall that with one-instantaneous zebroids, for a given request, a new replica is created on a car in the vicinity of the server, provided this car meets the client earlier than any of the  $r_i$  servers. Moreover, this replica is spawned at the time step when the client issues the request. Let  $\bar{N}_i^c$  be the expected total number of nodes that are in the same cell as any of the  $r_i$  servers. Then, we have

$$\bar{N}_i^c = (N - r_i) \cdot \left(1 - \left(1 - \frac{1}{G}\right)^{r_i}\right) \quad (5.2)$$

In the analytical model, we assume that  $\bar{N}_i^c$  new replicas are created, so that the total number of replicas is increased to  $r_i + \bar{N}_i^c$ . The availability latency is reduced since the client is more likely to meet a replica earlier. The aggregated expected availability latency in the case of one-instantaneous zebroids is then given by,



$$\delta_{agg}(zeb) = \sum_{i=1}^T \frac{f_i \cdot c \cdot G \cdot \log G}{r_i + \overline{N}_i^c} = \sum_{i=1}^T \frac{f_i \cdot C}{r_i + \overline{N}_i^c} \quad (5.3)$$

Note that in obtaining this expression, for ease of analysis, we have assumed that the new replicas start from random locations in the torus (not necessarily from the same cell as the original  $r_i$  servers). It thus treats all the  $\overline{N}_i^c$  carriers independently, just like the  $r_i$  original servers. As we shall show below by comparison with simulations, this approximation provides an upper-bound on the improvements that can be obtained because it results in a lower expected latency at the client.

It should be noted that the procedure listed above will yield a similar latency to that employed by a dispatcher employing one-instantaneous zebroids (see Section 5.2). Since the dispatcher is aware of all future car movements it would only transfer the requested data item on a single zebroid, if it determines that the zebroid will meet the client earlier than any other server. This selected zebroid is included in the  $\overline{N}_i^c$  new replicas.

#### 5.4.2 z-relay zebroids

The expected availability latency with z-relay zebroids can be calculated using a coloring problem analog similar to an approach used by Spyropoulos *et al.* [55]. Consider a data item  $i$  requested by the client. Recall that, there are  $N$  total cars and  $r_i$  replicas for data item  $i$ . Assume that each of these  $r_i$  replicas is colored red, while the other cars including the client are colored blue. Whenever a red car encounters a blue car, the latter is colored red.

The expected number of steps until the client is colored red then gives the average availability latency with z-relay zebroids. If at a given step, there are  $k$  red cars ( $k \geq r_i$ ), then there will be  $N - k$  blue cars. Recall that meeting time between cars can be modelled as an exponential distribution. Hence, by the property of exponential distribution, the average time until any of the  $k$  red cars meets any of the  $N + 1 - k$  blue cars is  $\frac{C}{k \cdot (N + 1 - k)}$ . Now, the expected time until all the cars are colored red is  $\sum_{k=r_i}^N \frac{C}{k \cdot (N + 1 - k)}$ .

Note that the client may be colored red in any one of these steps with equal probability. Consequently, the expected time till the client is colored red is given by,

$$\bar{\delta}_i = \frac{C}{N+1-r_i} \sum_{m=r_i}^N \sum_{k=r_i}^m \frac{1}{k \cdot (N+1-k)} \quad (5.4)$$

Evaluating the above expression, we get,

$$\bar{\delta}_i = \frac{C}{N+1} \cdot \frac{1}{N+1-r_i} \cdot [N \cdot \log \frac{N}{r_i} - \log (N+1-r_i)] \quad (5.5)$$

Now, the aggregate availability latency ( $\delta_{agg}$ ) with z-relay zebroids is obtained by definition,

$$\begin{aligned} \delta_{agg}(zeb) = & \sum_{i=1}^T [f_i \cdot \frac{C}{N+1} \cdot \frac{1}{N+1-r_i} \cdot \\ & (N \cdot \log \frac{N}{r_i} - \log (N+1-r_i))] \end{aligned} \quad (5.6)$$

## 5.5 Simulation Methodology

The simulation environment considered in this study comprises of vehicles such as cars that carry a fraction of the data item repository. A prediction accuracy parameter inherently provides a certain probabilistic guarantee on the confidence of the car route predictions known at the dispatcher. A value of 100% implies that the exact routes of all cars are known at all times. A 70% value for this parameter indicates that the routes predicted for the cars will match the actual ones with probability 0.7. Note that this probability is spread across the car routes for the entire trip duration. We now provide the preliminaries of the simulation study and then describe the parameter settings used in our experiments.

- Similar to the analysis methodology, the map used is a 2D torus. A Markov mobility model representing a unbiased 2D random walk on the surface of the torus describes the movement of the cars across this torus.
- Each grid/cell is a unique state of this Markov chain. In each time slot, every car makes a transition from a cell to any of its neighboring 8 cells. The transition is

a function of the current location of the car and a probability transition matrix  $Q = [q_{ij}]$  where  $q_{ij}$  is the probability of transition from state  $i$  to state  $j$ . Only AutoMata equipped cars within the same cell may communicate with each other.

- The parameters  $\gamma, \delta$  have been discretized and expressed in terms of the number of time slots.
- An AutoMata device does not maintain more than one replica of a data item. This is because additional replicas occupy storage without providing benefits.
- Either one-instantaneous or z-relay zebroids may be employed per client request for latency improvement.
- Unless otherwise mentioned, the prediction accuracy parameter is assumed to be 100%. This is because this study aims to quantify the effect of a large number of parameters individually on availability latency.

Here, we set the size of every data item,  $S_i$ , to be 1.  $\alpha$  represents the number of storage slots per AutoMata. Each storage slot stores one data item.  $\gamma$  represents the duration of the client's journey in terms of the number of time slots. Hence the possible values of availability latency are between 0 and  $\gamma$ .  $\delta$  is defined as the number of time slots after which a client AutoMata device will encounter a replica of the data item for the first time. If a replica for the data item requested was encountered by the client in the first cell then we set  $\delta = 0$ . If  $\delta > \gamma$  then we set  $\delta = \gamma$  indicating that no copy of the requested data item was encountered by the client during its entire journey. In all our simulations, for illustration we consider a  $5 \times 5$  2D-torus with  $\gamma$  set to 10. Our experiments indicate that the trends in the results scale to maps of larger size.

We simulated a skewed distribution of access to the  $T$  data items that obeys Zipf's law [65] with a mean of 0.27. This distribution is shown to correspond to sale of movie theater tickets in the United States [15]. For the sake of completeness, we provide a brief overview of a Zipf distribution. This means that the frequency of the  $r^{th}$  popular data item is inversely proportional to its rank i.e.

$$f_i = \frac{\frac{1}{i^v}}{\sum_{j=1}^T \frac{1}{j^v}}; 1 \leq i \leq T \quad (5.7)$$

Here, the exponent  $v$  controls the skewness in the popularity distribution of the data items. We denote  $w = -v$  as the skewness parameter. A higher absolute value of  $w$  indicates that most of the popularity weight is spread across the first few popular titles. Note that the data item repository size is  $T$  and the denominator is simply a normalization constant.

We employ a replication scheme that allocates replicas for a data item as a function of the square-root of the frequency of access of that item. The square-root replication scheme is shown to have competitive latency performance over a large parameter space [19]. The data item replicas are distributed uniformly across the AutoMata devices. This serves as the base-line no-zebroids case. The square-root scheme also provides the initial replica distribution when zebroids are employed. Note that the replacements performed by the zebroids will cause changes to the data item replica distribution. Requests generated as per the Zipf distribution are issued one at a time. The client car that issues the request is chosen in a round-robin manner. After a maximum period of  $\gamma$ , the latency encountered by this request is recorded.

Initially, all cars are distributed across the map as per the steady-state distribution governed by  $Q$ . This initial placement of cars across the map is determined by a random number generator initialized with a seed. All results presented in this section are averages over 10 such seeds each invoking 20,000 requests. Hence, each point in all the presented results is an average of 200,000 requests.

The 95% confidence intervals are determined for all sets of results. These intervals are quite tight for the metrics of latency and replacement overhead, hence, we only present them for the metric that captures the percentage improvement in latency with respect to the no-zebroids case.

## 5.6 Results

In this section, we describe our evaluation results where the following key questions are addressed. With a wide choice of replacement schemes available for a zebroid, what is their effect on availability latency? A more central question may be: Do zebroids provide significant improvements in availability latency? What is the associated overhead incurred in employing these zebroids? What happens to these improvements in scenarios where a dispatcher may have imperfect information about the car routes? What inherent trade-offs exist between car density and storage per car with regards to their combined as well as individual effect on availability latency in the presence of zebroids? We present both simple analysis and detailed simulations to provide answers to these questions as well as gain insights into design of carrier-based systems.

### 5.6.1 Zebroid replacement schemes

In this section, we describe how replacement policies employed by zebroids impact availability latency. For illustration, we present ‘scale-up’ experiments where one-instantaneous zebroids are employed (see Figure 5.1). By scale-up, we mean that  $\alpha$  and  $N$  are changed proportionally to keep the total system storage,  $S_T$ , constant. Here, we set  $T = 50$  and  $S_T = 200$ . We choose the following values of  $(N, \alpha) = \{(20, 10), (25, 8), (50, 4), (100, 2)\}$ . The figure indicates that a random replacement scheme shows a competitive performance. This is because of several reasons.

Recall that the initial replica distribution is set as per the square-root replication scheme. The random replacement scheme does not alter this distribution since it makes replacements blind to the popularity of a data item. However, the replacements cause dynamic data re-organization so as to better serve the currently active request. Moreover, the mobility pattern of the cars is random, hence, the locations from which the requests are issued by clients are also random and not known a priori at the dispatcher. These findings are significant because a random replacement policy can be implemented in a simple decentralized manner.

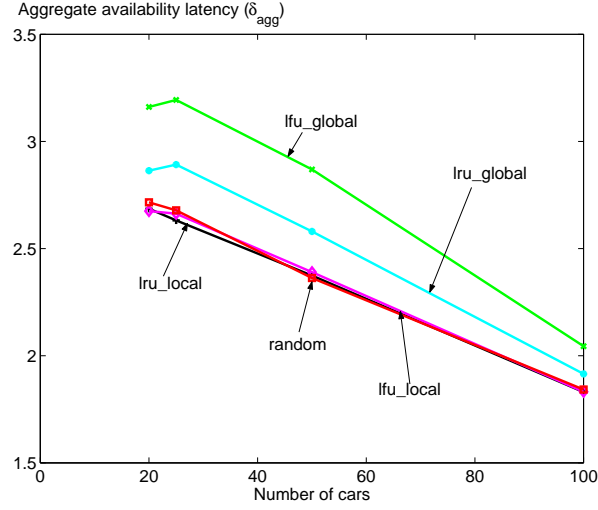


Figure 5.1: Availability latency when employing one-instantaneous zebroids as a function of  $(N, \alpha)$  values, when the total storage in the system is kept fixed,  $S_T = 200$ .

The lru-global and lfu-global schemes provide a latency performance that is worse than random. This is because these policies rapidly develop a preference for the more popular data items thereby creating a larger number of replicas for them. During eviction, the more popular data items are almost never selected as a replacement candidate. Consequently, there are fewer replicas for the less popular items. Hence, the initial distribution of the data item replicas changes from square-root to that resembling linear replication. The higher number of replicas for the popular data items provide marginal additional benefits, while the lower number of replicas for the other data items hurts the latency performance of these global policies. The lfu-local and lru-local schemes have similar performance to random since they do not have enough history of local data item requests. We speculate that the performance of these local policies will approach that of their global variants for a large enough history of data item requests. On account of the competitive performance shown by a random policy, for the remainder of the paper, we present the performance of zebroids that employ a random replacement policy.

As part of our future work, it remains to be seen if there are other more sophisticated replacement schemes that may have a performance better than random. Moreover, the distribution of replicas seems to have a profound impact on the availability latency in the

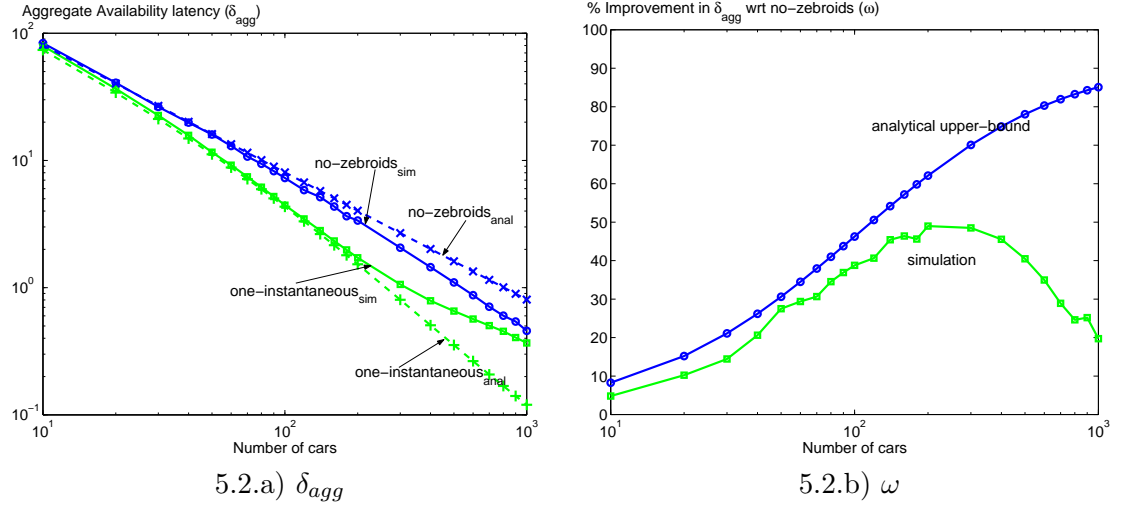


Figure 5.2: Latency performance with one-instantaneous zebroids via simulations along with the analytical approximation for a  $10 \times 10$  torus with  $T = 10$ .

presence of zebroids. Exploring and analyzing the cumulative effect of different replica distributions with zebroids presents a promising future research direction. A concrete goal is the investigation of a replacement scheme that over time converges to a replica distribution resembling that provided by a square-root replication scheme.

### 5.6.2 Zebroids performance improvement

We find that in many scenarios employing zebroids provides substantial improvements in availability latency.

#### 5.6.2.1 Analysis

We first consider the case of one-instantaneous zebroids. Figure 5.2.a shows the variation in  $\delta_{agg}$  as a function of  $N$  for  $T = 10$  and  $\alpha = 1$  with a  $10 \times 10$  torus using Equation 5.3. Both the x and y axes are drawn to a log-scale. Figure 5.2.b show the % improvement in  $\delta_{agg}$  obtained with one-instantaneous zebroids. In this case, only the x-axis is drawn to a log-scale. For illustration, we assume that the  $T$  data items are requested uniformly.

Initially, when the network is sparse the analytical approximation for improvements in latency with zebroids, obtained from Equations 5.1 and 5.3, closely matches the simulation

results. However, as  $N$  increases, the sparseness assumption for which the analysis is valid, namely  $N \ll G$ , is no longer true. Hence, the two curves rapidly diverge. The point at which the two curves move away from each other corresponds to a value of  $\delta_{agg} \leq 1$ . Moreover, as mentioned earlier, the analysis provides an upper bound on the latency improvements, as it treats the newly created replicas given by  $\overline{N}_i^c$  independently. However, these  $\overline{N}_i^c$  replicas start from the same cell as one of the server replicas  $r_i$ . Finally, the analysis captures a one-shot scenario where given an initial data item replica distribution, the availability latency is computed. The new replicas created do not affect future requests from the client.

Next, we consider the case where z-relay zebroids are employed (see Figure 5.3). Similar observations, like the one-instantaneous zebroid case, apply since the simulation and analysis curves again start diverging when the analysis assumptions are no longer valid. However, the key observation is that the latency improvement with z-relay zebroids is significantly better than the one-instantaneous zebroids case, especially for lower storage scenarios. This is because in sparse scenarios, the transitive hand-offs between the zebroids creates higher number of replicas for the requested data item, yielding lower availability latency. Moreover, it is also seen that the simulation validation curve for the improvements in  $\delta_{agg}$  with z-relay zebroids approaches that of the one-instantaneous zebroid case for higher storage (higher  $N$  values). This is because one-instantaneous zebroids are a special case of z-relay zebroids.

### 5.6.2.2 Simulation

We conduct simulations to examine the entire storage spectrum obtained by changing car density  $N$  or storage per car  $\alpha$  in order to also capture scenarios where the sparseness assumptions for which the analysis is valid do not hold. We separate the effect of  $N$  and  $\alpha$  by capturing the variation of  $N$  while keeping  $\alpha$  constant (case 1) and vice-versa (case 2) both with z-relay and one-instantaneous zebroids. Here, we set the repository size as  $T = 25$ . Figure 5.4 and 5.5 respectively capture the two cases mentioned above. With Figure 5.4.b, keeping  $\alpha$  constant, initially increasing car density has higher latency benefits because increasing  $N$  introduces more zebroids in the system. As  $N$  is further



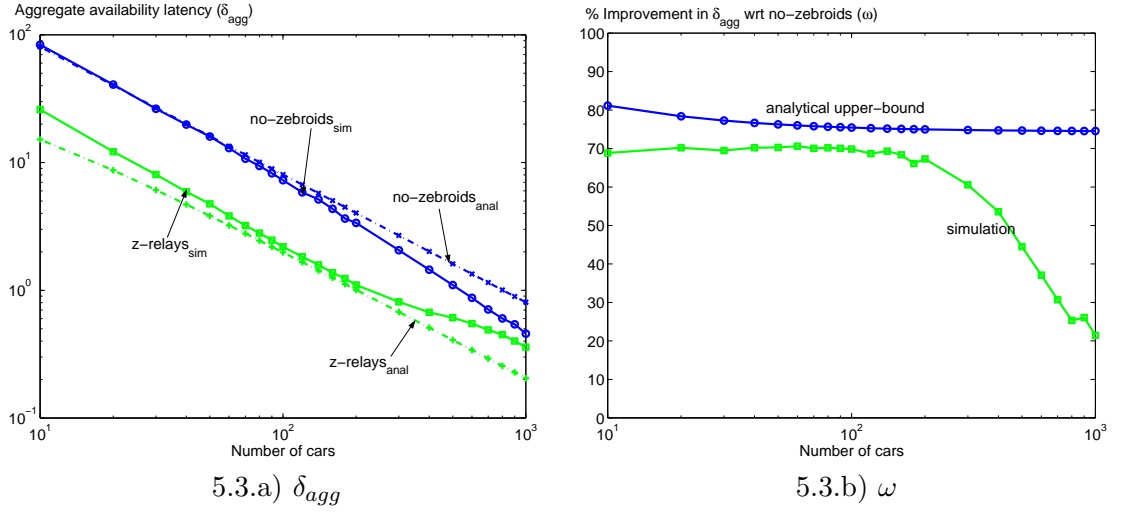


Figure 5.3: Latency performance with z-relay zebroids via analysis and simulations for a  $10 \times 10$  torus with  $T = 10$ .

increased,  $\omega$  reduces because the total storage in the system goes up. Consequently, the number of replicas per data item goes up thereby increasing the number of servers. Hence, the replacement policy cannot find a zebroid as often to transport the requested data item to the client earlier than any of the servers. On the other hand, the increased number of servers benefits the no-zebroids case in bringing  $\delta_{agg}$  down. The net effect results in reduction in  $\omega$  for larger values of  $N$ . Similar trends are seen by keeping  $N$  constant and increasing  $\alpha$  (see Figure 5.5.b).

The trends mentioned above are similar to that obtained from the analysis. However, somewhat counter-intuitively with relatively higher system storage, z-relay zebroids provide slightly lower improvements in latency as compared to one-instantaneous zebroids. We speculate that this is due to the different data item replica distributions enforced by them. Note that replacements performed by the zebroids cause fluctuations in these replica distributions which may effect future client requests. Exploring other suitable choices of parameters that can capture these changing replica distributions may be a useful future research direction.

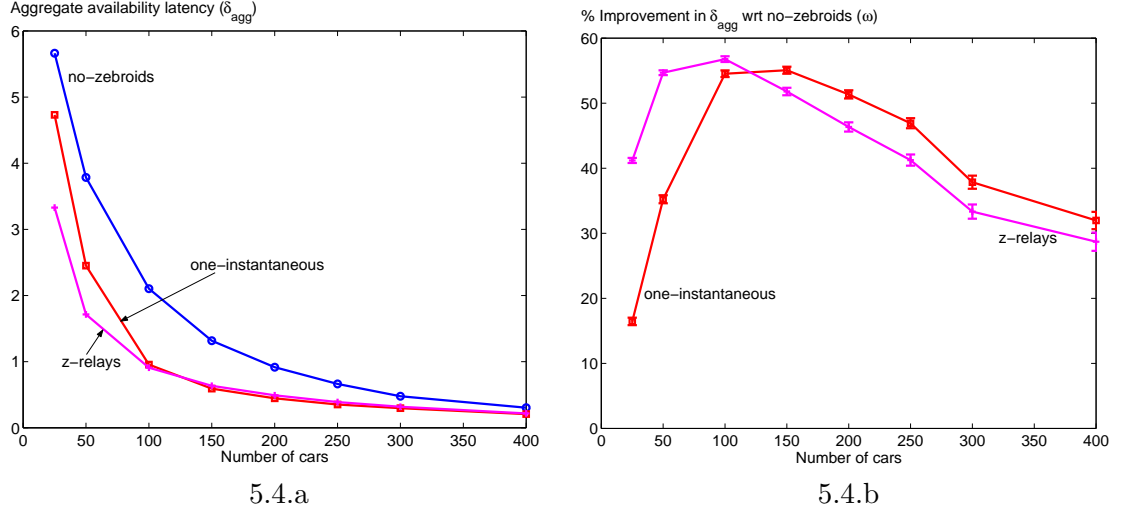


Figure 5.4: Latency performance with both one-instantaneous and z-relay zebroids as a function of the car density when  $\alpha = 2$  and  $T = 25$ .

### 5.6.3 Zebroid overhead

We find that the improvements in latency with zebroids are obtained at a minimal replacement overhead ( $< 1$  per client request).

#### 5.6.3.1 Analysis

With one-instantaneous zebroids, for each client request a maximum of one zebroid is employed for latency improvement. Hence, the replacement overhead per client request can amount to a maximum of one. Recall that to calculate the latency with one-instantaneous zebroids,  $\overline{N}_i^c$  new replicas are created in the same cell as the servers. Now a replacement is only incurred if one of these  $\overline{N}_i^c$  newly created replicas meets the client earlier than any of the  $r_i$  servers.

Let  $X_{r_i}$  and  $X_{\overline{N}_i^c}$  respectively be random variables that capture the minimum time till any of the  $r_i$  and  $\overline{N}_i^c$  replicas meet the client. Since  $X_{r_i}$  and  $X_{\overline{N}_i^c}$  are assumed to be independent, by the property of exponentially distributed random variables we have,

$$Overhead/request = 1 \cdot P(X_{\overline{N}_i^c} < X_{r_i}) + 0 \cdot P(X_{r_i} \leq X_{\overline{N}_i^c}) \quad (5.8)$$

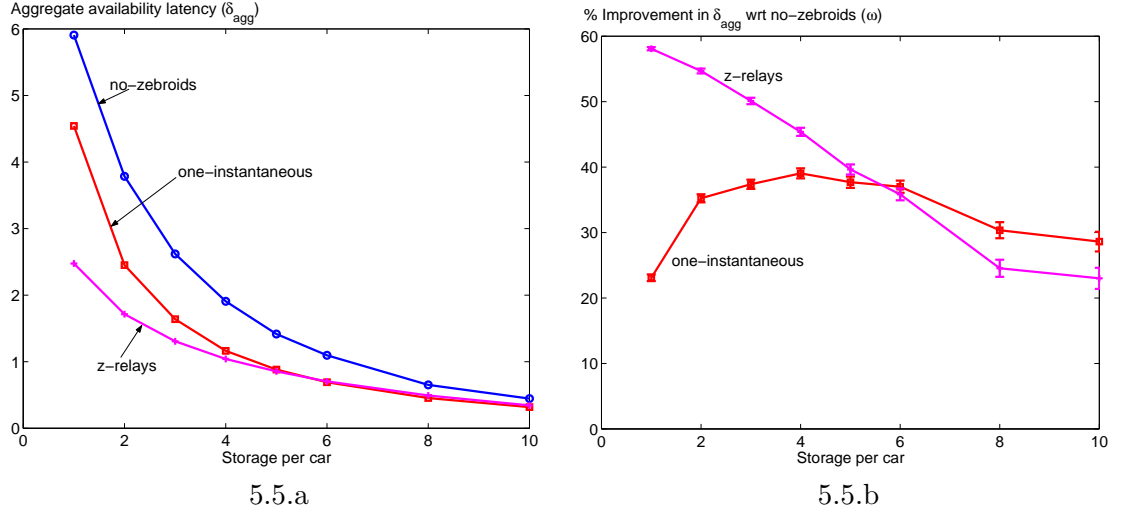


Figure 5.5: Latency performance with both one-instantaneous and z-relay zebroids as a function of  $\alpha$  when  $N = 50$  and  $T = 25$ .

$$Overhead/request = \frac{\frac{r_i}{C}}{\frac{r_i}{C} + \frac{N_i^c}{C}} = \frac{r_i}{r_i + N_i^c} \quad (5.9)$$

Recall that the number of replicas for data item  $i$ ,  $r_i$ , is a function of the total storage in the system i.e.,  $r_i = k \cdot N \cdot \alpha$  where  $k$  satisfies the constraint  $1 \leq r_i \leq N$ . Using this along with Equation 5.1, we get

$$Overhead/request = 1 - \frac{G}{G + N \cdot (1 - k \cdot \alpha)} \quad (5.10)$$

Now if we keep the total system storage  $N \cdot \alpha$  constant since  $G$  and  $T$  are also constant, increasing  $N$  increases the replacement overhead. However, if  $N \cdot \alpha$  is constant then increasing  $N$  causes  $\alpha$  to go down. This implies that a higher replacement overhead is incurred for higher  $N$  and lower  $\alpha$  values. Moreover, when  $r_i = N$ , this means that every car has a replica of data item  $i$ . Hence, no zebroids are employed when this item is requested, yielding an overhead/request for this item as zero. Next, we present simulation results that validate our analysis hypothesis for the overhead associated with deployment of one-instantaneous zebroids.

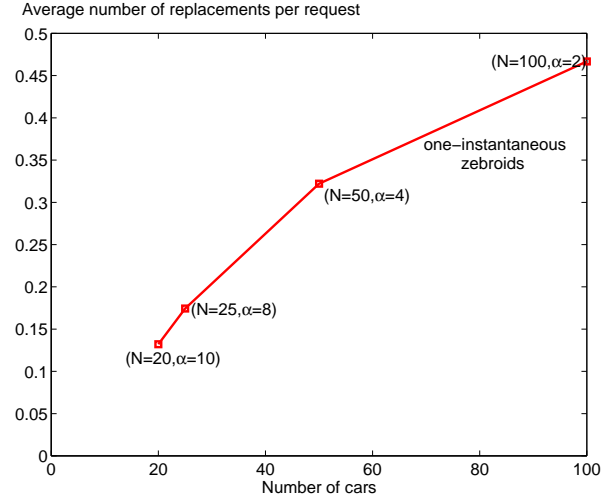


Figure 5.6: Replacement overhead when employing one-instantaneous zebroids as a function of  $(N, \alpha)$  values, when the total storage in the system is kept fixed,  $S_T = 200$ .

### 5.6.3.2 Simulation

Figure 5.6 shows the replacement overhead with one-instantaneous zebroids when  $(N, \alpha)$  are varied while keeping the total system storage constant. The trends shown by the simulation are in agreement with those predicted by the analysis above. However, the total system storage can be changed either by varying car density ( $N$ ) or storage per car ( $\alpha$ ). Figures 5.7.a and Figure 5.7.b respectively indicate the replacement overhead incurred with both one-instantaneous and z-relay zebroids when  $\alpha$  is kept constant and  $N$  is varied and vice-versa.

We present an intuitive argument for the behavior of the per-request replacement overhead curves. When the storage is extremely scarce so that only one replica per data item exists in the AutoMata network, the number of replacements performed by the zebroids is zero since any replacement will cause a data item to be lost from the system. The dispatcher ensures that no data item is lost from the system. At the other end of the spectrum, if storage becomes so abundant that  $\alpha = T$  then the entire data item repository can be replicated on every car. The number of replacements is again zero since each request can be satisfied locally. A similar scenario occurs if  $N$  is increased

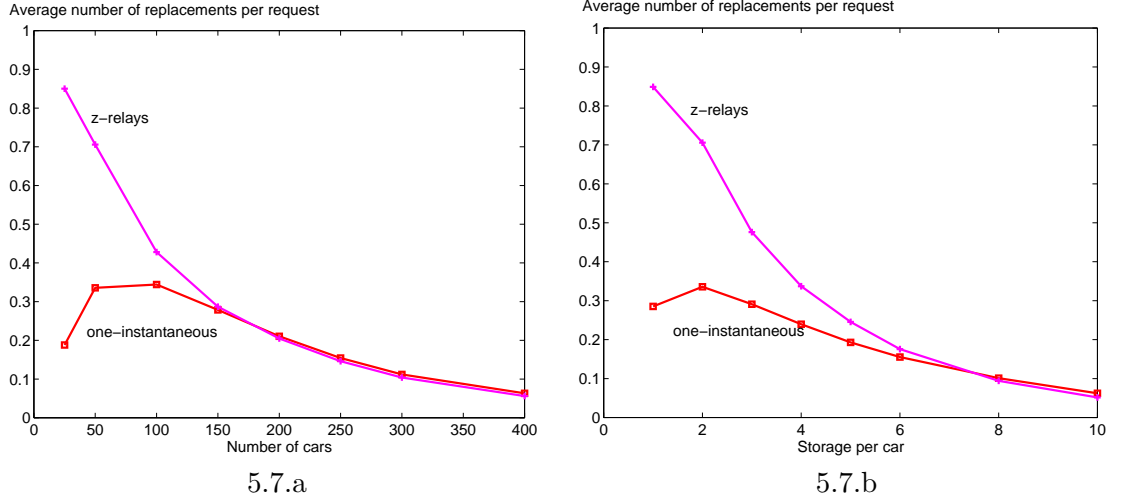


Figure 5.7: Replacement overhead with zebroids for the cases when  $N$  is varied keeping  $\alpha = 2$  (figure 5.7.a) and  $\alpha$  is varied keeping  $N = 50$  (figure 5.7.b).

to such a large value that another car with the requested data item is always available in the vicinity of the client. However, there is a storage spectrum in the middle where replacements by the scheduled zebroids result in improvements in  $\delta_{agg}$  (see Figures 5.4.b and 5.5.b).

Moreover, we observe that for sparse storage scenarios, the higher improvements with z-relay zebroids are obtained at the cost of a higher replacement overhead when compared to the one-instantaneous zebroids case. This is because in the former case, each of these  $z$  zebroids selected along the lowest latency path to the client needs to perform a replacement. However, the replacement overhead is still less than 1 indicating that on an average less than one replacement per client request is needed even when z-relay zebroids are employed.

Note that the average replacement per request metric does not explicitly capture the bandwidth overhead associated with the transfer of items to the zebroids. This bandwidth overhead may be significant in the case of multiple simultaneous active requests. We intend to explicitly incorporate these bandwidth considerations in our model as part of our future research.

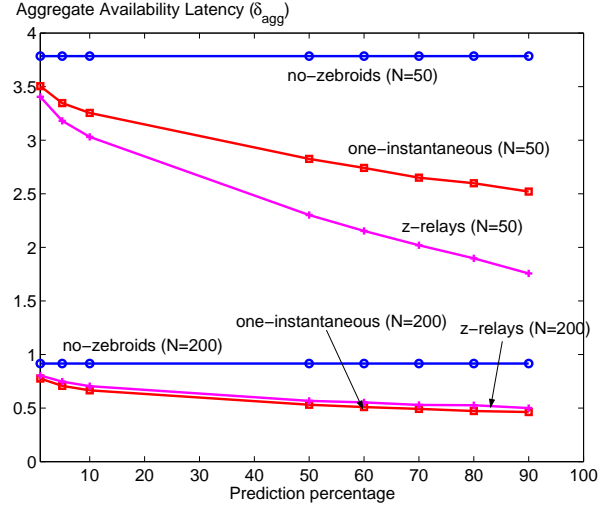


Figure 5.8: Availability latency,  $\delta_{agg}$ , for different car densities as a function of the prediction accuracy metric with  $\alpha = 2$  and  $T = 25$ .

#### 5.6.4 Zebroids with inaccurate route predictability

We find that zebroids continue to provide improvements in availability latency even with lower accuracy in the car route predictions. We use a single parameter  $p$  to quantify the accuracy of the car route predictions. This parameter inherently provides a certain probabilistic guarantee on the confidence of the car route predictions for the entire trip duration.

##### 5.6.4.1 Analysis

Since  $p$  represents the probability that a car route predicted by the dispatcher matches the actual one, hence, the latency with zebroids can be approximated by,

$$\delta_{agg}^{err} = p \cdot \delta_{agg}(zeb) + (1 - p) \cdot \delta_{agg}(no - zeb) \quad (5.11)$$

$$\delta_{agg}^{err} = p \cdot \delta_{agg}(zeb) + (1 - p) \cdot \frac{C}{r_i} \quad (5.12)$$

Expressions for  $\delta_{agg}(zeb)$  can be obtained from Equations 5.3 (one-instantaneous) or 5.6 (z-relay zebroids).

#### 5.6.4.2 Simulation

Figure 5.8 shows the variation in  $\delta_{agg}$  as a function of this route prediction accuracy metric. We observe a smooth reduction in the improvement in  $\delta_{agg}$  as the prediction accuracy metric reduces. For zebroids that are scheduled but fail to rendezvous with the client due to the prediction error, we tag any such replacements made by the zebroids as failed. It is seen that failed replacements gradually increase as the prediction accuracy reduces.

In this study, we have considered a metric that probabilistically governs errors in the car route predictions. Another possible choice for the metric is similar to that used by Jun *et al.* [36] where the car routes are assumed to follow a Gaussian distribution defined by a mean and a variance. The estimates about the mean and the variance can be built at the dispatcher based on the history of the individual car movements. Exploring such alternate choices of prediction control metrics presents a promising future research direction.

#### 5.6.5 Maximum improvement with zebroids

Surprisingly, we find that the improvements in latency obtained with one-instantaneous zebroids are independent of the input distribution of the popularity of the data items.

##### 5.6.5.1 Analysis

The fractional difference (labelled  $\omega$ ) in the latency between the no-zebroids and one-instantaneous zebroids is obtained from equations 5.1, 5.2, and 5.3 as

$$\omega = \frac{\sum_{i=1}^T \frac{f_i \cdot C}{r_i} - \sum_{i=1}^T \frac{f_i \cdot C}{r_i + (N - r_i) \cdot \left(1 - \left(1 - \frac{1}{G}\right)^{r_i}\right)}}{\sum_{i=1}^T \frac{f_i \cdot C}{r_i}} \quad (5.13)$$

Here  $C = c \cdot G \cdot \log G$ . This captures the fractional improvement in the availability latency obtained by employing one-instantaneous zebroids. Let  $\alpha = 1$ , making the total

storage in the system  $S_T = N$ . Assuming the initial replica distribution is as per the square-root replication scheme, we have,  $r_i = \frac{\sqrt{f_i} \cdot N}{\sum_{j=1}^T \sqrt{f_j}}$ . Hence, we get  $f_i = \frac{K^2 \cdot r_i^2}{N^2}$ , where  $K = \sum_{j=1}^T \sqrt{f_j}$ . Using this, along with the approximation  $(1 - x)^n \simeq 1 - n \cdot x$  for small  $x$ , we simplify the above equation to get,

$$\omega = 1 - \frac{\sum_{i=1}^T \frac{r_i}{1 + \frac{N - r_i}{G}}}{\sum_{i=1}^T r_i} \quad (5.14)$$

In order to determine when the gains with one-instantaneous zebroids are maximized, we can frame an optimization problem as follows:

$$\text{Maximize } \omega, \text{ subject to } \sum_{i=1}^T r_i = S_T \quad (5.15)$$

**Theorem 2.** *With a square-root replication scheme, improvements obtained with one-instantaneous zebroids are maximized with a uniform input popularity distribution of the data items.*

*Proof.* Using the Lagrangian multipliers method, the optimization can be expressed as:

$$\text{Max} \left\{ 1 - \sum_{i=1}^T \frac{G \cdot r_i}{N \cdot (G + N - r_i)} + \lambda \cdot \left[ \sum_{i=1}^T r_i - N \right] \right\} \quad (5.16)$$

We solve for  $r_i$  to obtain:

$$r_i = G + N - G \cdot \sqrt{\frac{G + N}{G \cdot N \cdot \lambda}} \quad (5.17)$$

Note that in Equation 5.17, while  $r_i$  is independent of  $i$ , it is the same for all titles  $i$  and is given by  $\frac{N}{T}$ . It can be verified that the maximum  $\omega$  occurs at this value of  $r_i$  since  $\frac{\partial^2 \omega}{\partial r_i^2} < 0$ . This implies that  $f_i = \frac{1}{T}$ , in other words, all the data items must have the same popularity.  $\square$

### 5.6.5.2 Simulation

We perform simulations with two different frequency distribution of data items: Uniform and Zipf (with mean=0.27). Similar latency improvements with one-instantaneous



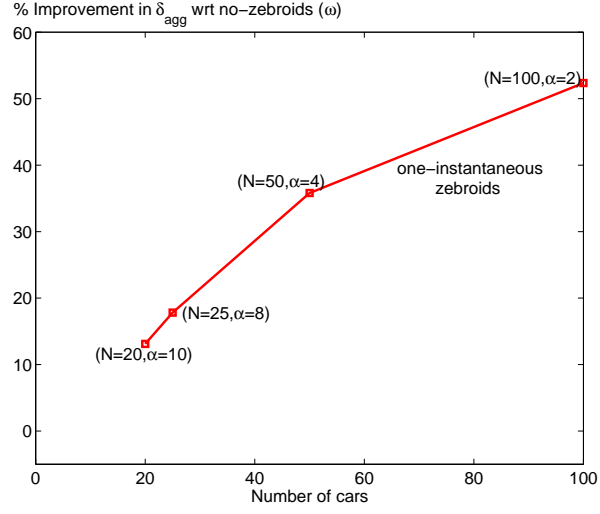


Figure 5.9: Improvement in availability latency with one-instantaneous zebroids as a function of  $(N, \alpha)$  values, when the total storage in the system is kept fixed,  $S_T = 200$ .

zebroids are obtained in both cases. This result has important implications. In cases with biased popularity toward certain data items, the aggregate improvements in latency across all data item requests still remain the same. Even in scenarios where the frequency of access to the data items changes dynamically, zebroids will continue to provide similar latency improvements.

### 5.6.6 Zebroid trade-offs with car density and storage per car

Our findings indicate that higher latency improvements can be obtained with zebroids when there are more cars with lower storage than fewer ones with higher storage.

#### 5.6.6.1 Analysis

Consider the case of one-instantaneous zebroids. The fractional difference (labelled  $\omega$ ) in  $\delta_{agg}$  between the no-zebroids and one-instantaneous zebroids cases is obtained in Equation 5.13. Using the approximation  $(1 - x)^n \simeq 1 - n \cdot x$  for small  $x$ , we simplify the above equation to get,

$$\omega = 1 - \frac{\sum_{i=1}^T \frac{G \cdot f_i}{r_i \cdot (G + N - r_i)}}{\sum_{i=1}^T \frac{f_i}{r_i}} \quad (5.18)$$

Recall that the number of replicas for data item  $i$ ,  $r_i$ , is a function of the total storage in the system i.e.,  $r_i = k \cdot N \cdot \alpha$  where  $k$  has to satisfy the constraint that  $1 \leq r_i \leq N$ . Given a total system storage  $N \cdot \alpha$ , we find that except for  $N$  all other terms in Equation 5.18 are constant. Also, with increasing  $N$ ,  $\omega$  increases. However, for a constant  $N \cdot \alpha$ , if  $N$  increases,  $\alpha$  has to reduce. This indicates for a given system storage, higher improvements in latency with zebroids are obtained with higher car density and lower storage per car.

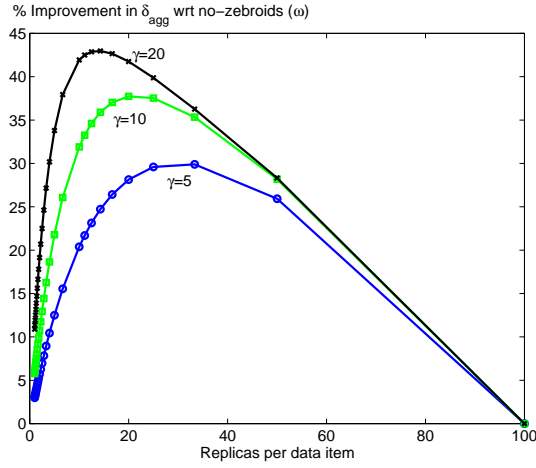
#### 5.6.6.2 Simulation

Figure 5.9 validates the insight obtained from the analysis in that the improvements in latency go up with higher  $N$  and lower  $\alpha$  values when  $N \cdot \alpha = 200$ . The increase in  $N$  increases the zebroid density enabling the dispatcher to almost always find a zebroid that can deliver the requested data item to the client earlier than any of the potential servers. This trade-off between the two system parameters of number of cars and storage per car may have important implications in the design of carrier-based networks that improve availability latency.

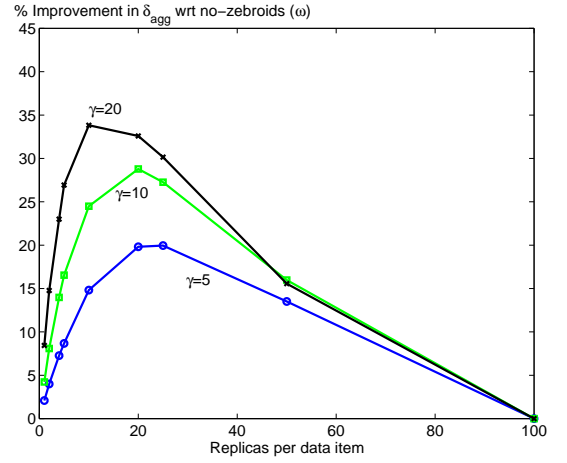
Although we have assumed a constant storage per car for all cars, in practical scenarios, there may be variable storage per car. Part of the AutoMata device storage space may be reserved by a user for his preferred titles which the user may not seek to erase/evict. These considerations may create a more heterogeneous environment with variable storage per car. In addition, zebroids may need to take into account user preferences prior to making these replacements. Incorporating all these considerations into the model is likely to pay richer dividends in estimating latency in real deployments.

#### 5.6.7 Impact of different trip durations and repository sizes

In this section, we describe the impact of different trip durations and repository sizes on availability latency in the presence of zebroids.



5.10.a) Analysis



5.10.b) Simulation

Figure 5.10: Improvement in  $\delta_{agg}$  with one-instantaneous zebroids for different client trip durations in case of  $10 \times 10$  torus with a fixed car density,  $N = 100$ .

#### 5.6.7.1 Analysis

In most practical scenarios, the client will wait for a maximum duration within which it expects its issued request to be satisfied. Here, we consider the case where the client has a finite trip duration  $\gamma$ , similar to that considered in the simulation environment ( $\gamma = 10$ ). The availability latency,  $\delta_i$ , can be any value between 0 and  $\gamma - 1$ . If the client's request is not satisfied, we set  $\delta_i = \gamma$  indicating that the client's request for item  $i$  was not satisfied.

Recall that latency in the case of a 2D-random walk on a torus can be modelled as an exponential distribution as:

$$P(\delta_i > t) = \lambda \cdot \exp(-\lambda \cdot t) \quad (5.19)$$

where  $\lambda = \frac{r_i}{c \cdot G \cdot \log G}$ . The average availability latency with finite trip duration  $\gamma$  is then given by,

$$\bar{\delta}_i = \int_0^\gamma x \cdot \lambda \cdot \exp(-\lambda \cdot t) dx + \int_\gamma^\infty \gamma \cdot \lambda \cdot \exp(-\lambda \cdot t) dx \quad (5.20)$$

Hence, we get

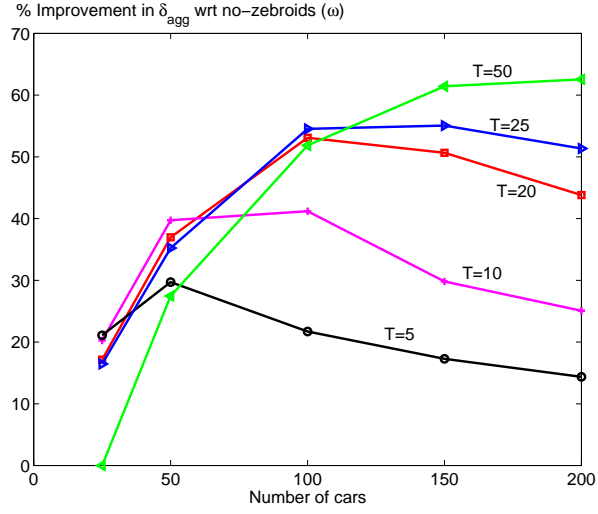


Figure 5.11: Shows improvement in availability latency as a function of the car density for different repository sizes with  $\alpha = 2$  and  $\gamma = 10$ .

$$\bar{\delta}_i = \frac{c \cdot G \cdot \log G}{r_i} \cdot [1 - \exp(\frac{-\gamma \cdot r_i}{c \cdot G \cdot \log G})] \quad (5.21)$$

The aggregate availability latency with finite trip duration is then given by,

$$\delta_{agg}(no - zeb) = \sum_{i=1}^T f_i \cdot \frac{c \cdot G \cdot \log G}{r_i} \cdot [1 - \exp(\frac{-\gamma \cdot r_i}{c \cdot G \cdot \log G})] \quad (5.22)$$

In the presence of one-instantaneous zebroids, the aggregate availability latency can be obtained using a procedure similar to that used in Section 5.4.1, giving

$$\delta_{agg}(zeb) = \sum_{i=1}^T f_i \cdot \frac{c \cdot G \cdot \log G}{(r_i + \bar{N}_i^c)} \cdot [1 - \exp(\frac{-\gamma \cdot (r_i + \bar{N}_i^c)}{c \cdot G \cdot \log G})] \quad (5.23)$$

The above equations yield the improvements in latency with finite trip duration. We consider a  $10 \times 10$  torus with  $N = 100$  cars each with one storage slot ( $\alpha = 1$ ). The distribution of data item replicas is assumed to be uniform. Hence, as we increase the size of the data item repository ( $T$ ) the number of replicas per data item decreases. Specifically, the value of  $T$  is varied as  $\{1, 2, 4, 10, 20, 25, 50, 100\}$ . Then, the number of replicas per data item changes as  $\{100, 50, 25, 10, 5, 4, 2, 1\}$ . Figure 5.10.a and 5.10.b

capture the latency performance obtained via analysis and simulations respectively for different trip durations.

We now describe the behavior of the curves for a given finite trip duration  $\gamma$ . When  $T = 1$ , every car has a copy of the item, hence, no car can serve as a zebroid. As  $T$  increases, replicas per item go down. Hence, some cars can potentially serve as zebroids, providing some improvements in availability latency. As  $T$  is further increased, a peak is reached beyond which the improvements start diminishing. This is because if  $T$  is large then the number of server replicas per item becomes small. Hence, the likelihood of finding another car in the vicinity of such a server that will meet the client earlier also reduces. Moreover, as  $\gamma$  increases, peak improvements in latency are obtained with higher  $T$ .

#### 5.6.7.2 Simulation

Here, we present simulation results that capture the effect of different repository sizes on the availability latency when the trip duration  $\gamma = 10$  (see Figure 5.11). For a fixed storage per car, sufficient car density is needed to provide higher improvements in latency for a given repository size. This implies that from a system designer's point of view, if an estimate of the total car density is known, then sufficient gains in latency with zebroids can be realized by adjusting the repository size of titles presented to the users.

While a homogeneous repository of data items has been assumed throughout this study, sizes of data items such as audio clips are typically smaller than video clips. One way in which our model can be extended to consider such a heterogeneous repository is to assume that every data item can be divided into a set of constant-sized blocks. Different blocks of an item may be stored across different cars. During data delivery, zebroids will be scheduled to ensure timely delivery of the various blocks of a requested data item to a client. Next, we validate a subset of the observations with zebroids with traces collected from a small vehicular test-bed.

## 5.7 Evaluation with a real map

In this section, we describe the performance improvements obtained with zebroids in a scenario where the vehicle movements are dictated by an underlying map of the San Francisco Bay Area. The details about how the Markov model was derived from the underlying map are described in Chapter 4, Section 4.6. Below we describe briefly the experimental set-up and corresponding results obtained with zebroids when such a Markov model is employed.

### 5.7.1 Results with zebroids

In this section, we present some representative results for the improvements in latency that are obtained with both one-instantaneous and z-relay zebroids when employing the Markov mobility model previously derived from a map of the San Francisco Bay area. As before, requests are issued, one at a time at each time-step at vehicles in a round-robin manner, as per a Zipf distribution with a mean of 0.27. At a time only one request is active, each request is active for a maximum of client trip duration number of steps (set as  $\gamma = 10$ ). In the following, we describe briefly the experiment set-up followed by a brief description of the main observations.

- The total storage in the system is held constant as  $S_T = 200$ . The values of  $(N, \alpha)$  are varied as  $\{(20, 10), (25, 8), (50, 4), (100, 2), (200, 1)\}$  to realize this value of  $S_T$ . For data item repository size  $T$  set as 25, client trip duration,  $\gamma$ , set as 10, the latency performance with zebroids is studied as a function of the different  $(N, \alpha)$  values (see Figure 5.12). Similar trends are obtained as were seen in Section 5.6.6. In other words, having more cars with lower storage provides higher latency improvements as opposed to having fewer cars with higher storage.
- For data item repository size  $T$  set as 25, client trip duration,  $\gamma$ , set as 10, storage per car,  $\alpha$ , set as 2, the latency performance with zebroids is studied as a function of increasing car density  $N$  (see Figure 5.13).

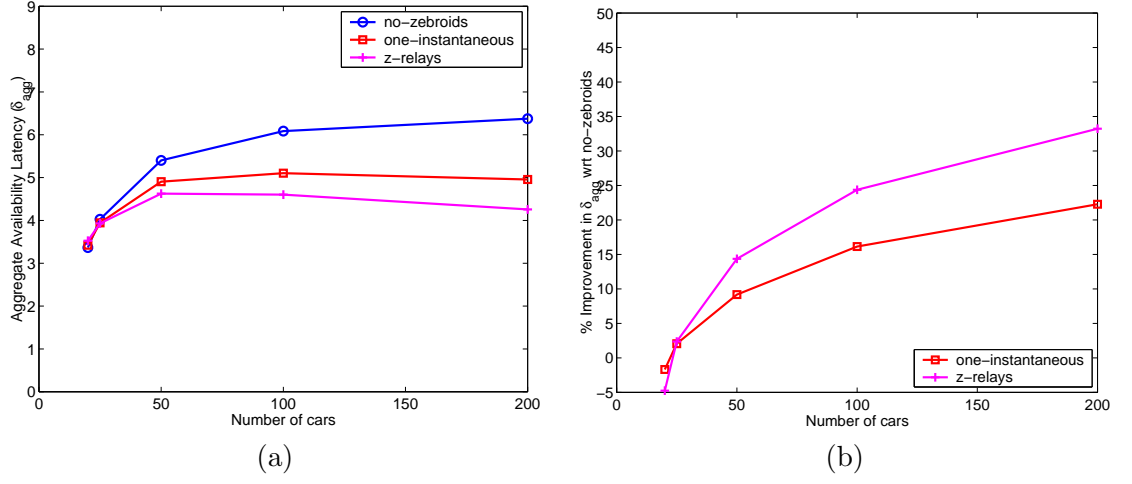


Figure 5.12: Performance with zebroids as a function of different  $(N, \alpha)$  values when the total storage in the system is held constant at  $S_T = 200$ ,  $\gamma = 10$ . Figure (b) shows the percentage improvement with zebroids when compared to the no-zebroids case.

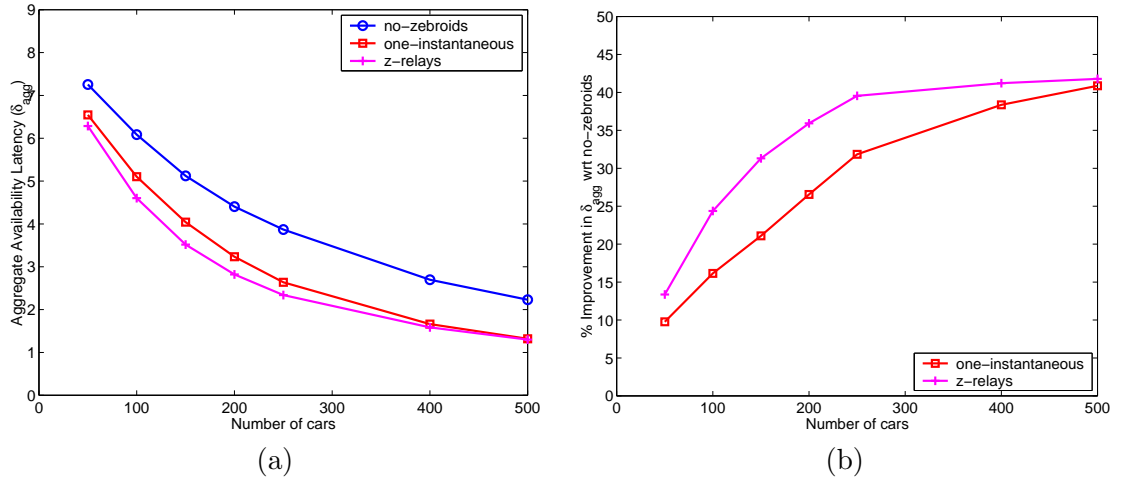


Figure 5.13: Performance with zebroids as a function of car density when  $T = 25$ ,  $\alpha = 2$ , and  $\gamma = 10$ . Figure (b) shows the percentage improvement with zebroids when compared to the no-zebroids case.

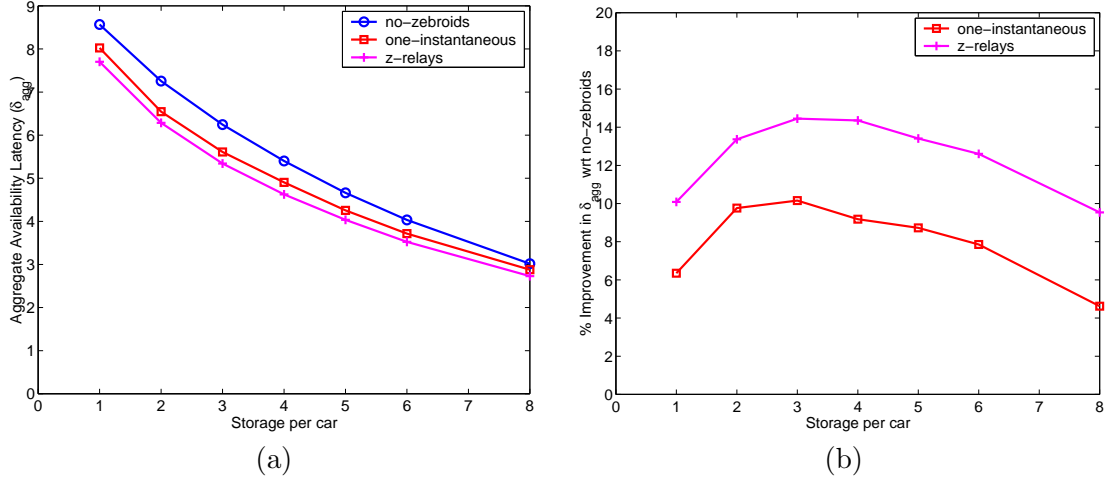


Figure 5.14: Performance with zebroids as a function of storage per car when  $T = 25$ ,  $N = 50$ , and  $\gamma = 10$ . Figure (b) shows the percentage improvement with zebroids when compared to the no-zebroids case.

- For data item repository size  $T$  set as 25, client trip duration,  $\gamma$ , set as 10, car density,  $N$ , set as 50, the latency performance with zebroids is studied as a function of increasing storage per car  $\alpha$  (see Figure 5.14). As seen in Section 5.6.2, we observe that there exists a storage density range that provides the highest improvements with zebroids.
- For car density,  $N$ , set as 50, client trip duration,  $\gamma$ , set as 10, storage per car,  $\alpha$ , set as 2, the latency performance with zebroids is studied as a function of increasing data item repository size  $T$  (see Figure 5.15). Again, the trends seen are similar to that observed in Section 5.6.7. For a small data item repository size, employing zebroids has almost no benefits over simply using the square-root static replication scheme. However, as the data item repository size increases, for the same total storage in the system ( $S_T = N \cdot \alpha$ ), the replicas per data item reduces, thereby increasing the utility of zebroids. However, if the data item repository is too large then the utility of zebroids is reduced because of the constraint that at least one replica of every data item must be present in the system at all times. Moreover,



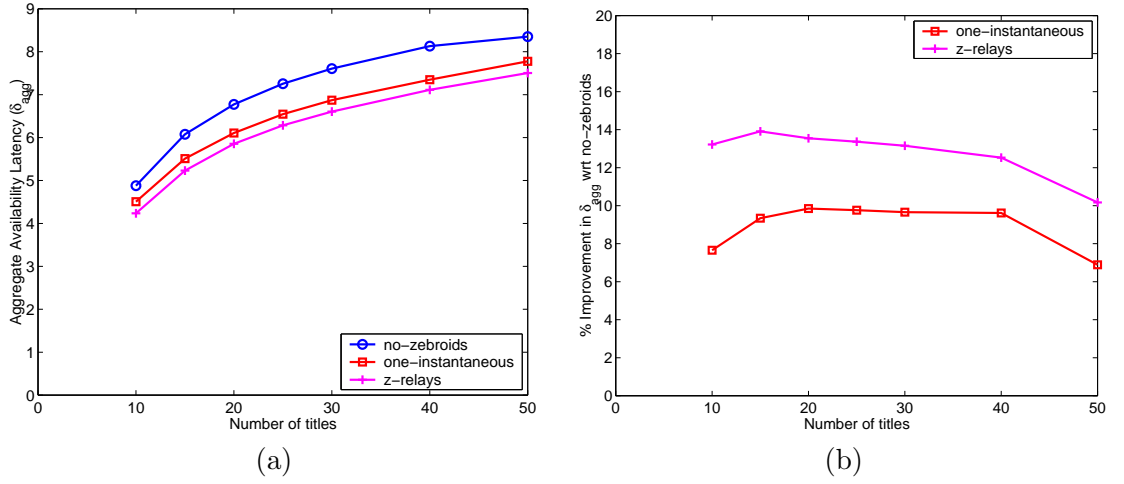


Figure 5.15: Performance with zebroids as a function of data item repository size when  $N = 50$ ,  $\alpha = 2$ , and  $\gamma = 10$ . Figure (b) shows the percentage improvement with zebroids when compared to the no-zebroids case.

the replicas per data item are further reduced thereby reducing the probability of finding a lower latency path from any server to the requesting client.

In all cases, it is seen that z-relay zebroids provide a higher latency improvement as compared to one-instantaneous zebroids albeit at a higher replacement overhead. Again, these results suggest that the uniform probability transition matrix based Markov model may be a good indicator of the performance that may be seen with a model derived from real maps.

## 5.8 Evaluation with Real Traces

In this section, we evaluate the performance of employing zebroids using traces obtained from the UMassDieselNet bus-based DTN test-bed [11]. The details of the test-bed and the properties of the traces have been explained in Chapter Section. Next, we describe the simulation set-up for the evaluation of zebroids followed by a brief explanation of the main results.

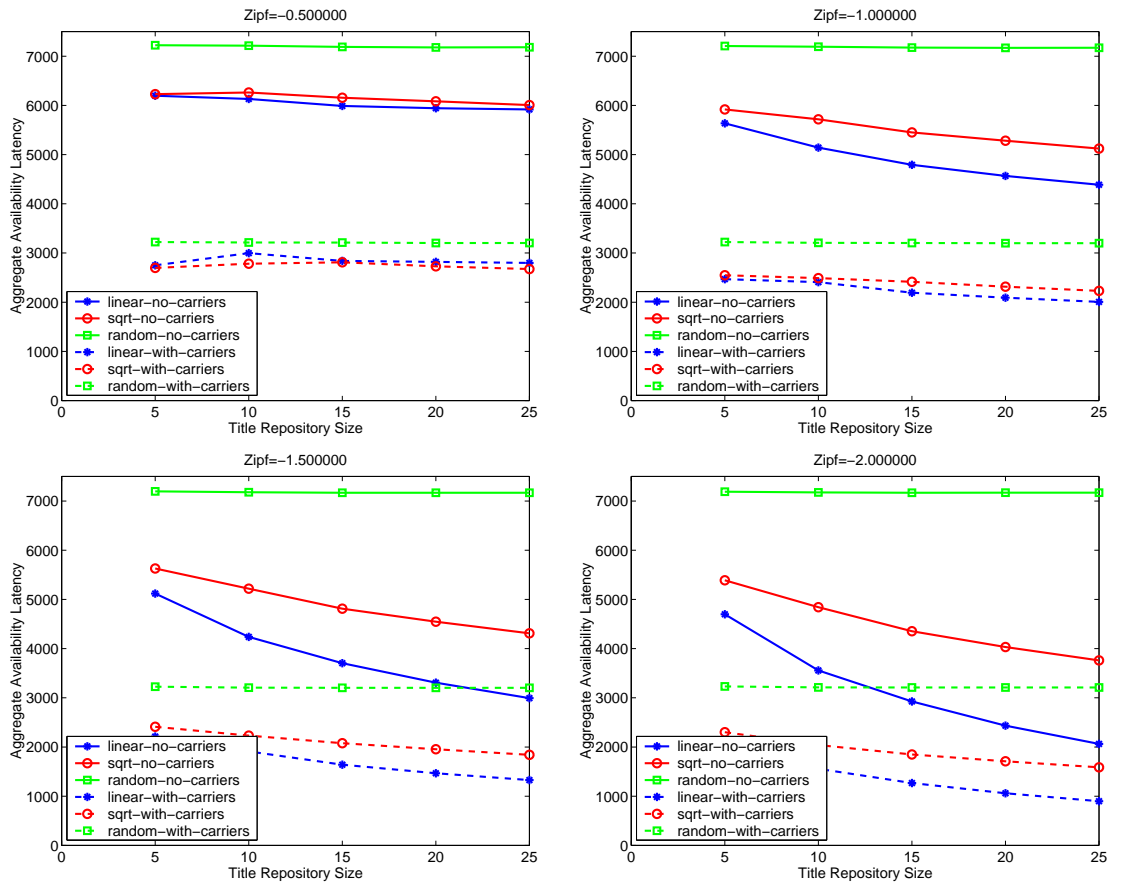


Figure 5.16: Aggregate Availability Latency for Linear, Sqrt, and Random Replication Schemes for Zipf values -0.5, -1.0, -1.5, and -2.0

### 5.8.1 Experimental Set-up

The UMassDieselNet traces represent the movements of buses during a particular day. There is no correlation between trace movements across days. Hence, we process each trace at a time and then average the results observed across all the days noting that the average is indicative of the performance seen on most days. However, certain days do appear as outliers since the number of active buses differs from day-to-day.

As before we consider a finite data item repository of size  $T$ . Each bus is assumed to carry  $\alpha$  storage slots. Initially, replicas for each data item are determined based on a square-root replication scheme then allocated across the buses uniformly at random. The constraint is that at least one copy of every data item must be present in the network at all times. Requests for data items are generated as per a Zipf distribution. We consider 2 different scenarios, (a) requests are issued for each data item at each bus at the start of the day (see Section 5.8.2.1) (b) requests for the data items are issued at equal inter-arrival times (see Section 5.8.2.2).

### 5.8.2 Results

In this section, we briefly describe the main results from evaluation of the performance of zebroids using the UMassDieselNet traces.

#### 5.8.2.1 Requests issued at the start of the day

As a base-line result for comparison, we first consider an optimistic scenario to evaluate the maximum improvements that can be obtained with zebroids. Requests for each data item are issued at each bus at the start of the day. For each request, we consider the best possible latency that can be obtained by employing zebroids. The scheduled relay teams of zebroids for the different requests do not interfere with each other. In other words, if there is a lower latency path from a server (bus with the requested data item) to the requesting client via other buses then storage constraints within the buses will not prevent the relay team of  $z$  zebroids to deliver the item to the client. We consider three different environments for the case without zebroids where the replicas for the data item are allocated as per the linear, square-root, and random replication schemes. Starting

from these three initial distributions, we employ zebroids to study the improvement in the aggregate availability latency. If a request cannot be satisfied then it is tagged with a maximum trip duration. This maximum trip duration is calculated on the basis of the duration of the traces across the 52 days. Hence, we only consider one metric namely the aggregate availability latency across all the requests.

Figure 5.16 shows the latency performance with the different replication schemes compared with the respective cases when zebroids are employed for 4 different data item popularity distributions. The values of  $(T, \alpha)$  are varied as  $\{(5,1), (10,2), (15,3), (20,4), (25,5)\}$ . In all cases, significant improvements in latency are obtained by employing zebroids. The improvements with the random replication scheme with and without zebroids are independent of the specific  $(T, \alpha)$  value. This is because the random replication scheme allocates replicas blind to the popularity of the data items. The linear replication scheme provides the lowest latency, because of the finite trip duration (see Chapter), the improvements are more pronounced with skewed popularity distributions and larger data item repositories.

#### 5.8.2.2 Requests issued at equal inter-arrival times during the day

Here, requests are generated as per a Zipf distribution with an exponent  $w = -0.73$ . The duration during which the buses were active during a day is determined apriori and subject to this duration requests are issued at equal inter-arrival times. A generated request is assigned to a bus chosen uniformly at random. A request is assumed to be satisfied either if the data item requested is locally stored or another bus carrying the requested item is encountered at some point after the request is issued. Those requests that are not satisfied at the end of the day are tagged as unsatisfied requests. Hence, we consider two separate metrics (i) Average availability latency for satisfied requests (ii) Normalized unsatisfied request rate.

For each request, a relay-team of  $z$  buses may be employed to improve its availability latency. The  $z$ -relay zebroids are scheduled on the basis of the state of the network at the time that the request is issued. Note that this relay team of buses are scheduled across space and time. As per the schedule, the time when one of the buses is supposed

to hand over a copy of the requested item to another one, the recipient may not have any free slots to perform the transfer. This is because the slots may all be occupied by items reserved for previously scheduled requests. When the z-relay team of zebroids is scheduled, the dispatcher only takes into account the spatio-temporal rendezvous of the buses with the client without regards to the content that the buses carry. This is because the content of the buses will change as other requests are issued into the system. Hence, not every scheduled z-relay transfer may be successful. If every zebroid in a z-relay schedule is able to carry the requested item then the item will be successfully delivered to the client yielding lower availability latency. Hence, we consider two other metrics: number of scheduled zebroids and number of zebroids that were actually able to carry the data item. As before the metrics obtained with zebroids are compared with a scenario without zebroids (no-zebroids case).

For the first set of experiments we vary the values of  $(T, \alpha)$  as  $\{(5,1), (10,2), (15,3), (20,4), (25,5), (30,6)\}$ , see Figure 5.17. The latency for satisfied requests with zebroids is about 15 – 20% lower than the no-zebroids scheme while the normalized unsatisfied requests with and without zebroids are the same. The replacement overhead for zebroids is depicted in Figure 5.18. While the zebroids scheduled for all the  $(T, \alpha)$  values is the same, a higher percentage of these scheduled zebroids are employed for higher data item repository sizes.

Figure 5.19 shows the performance with zebroids when the data item repository size is fixed at 10 and the storage per bus is increased. Increase in storage leads to increase in the replicas per data item, hence, as expected the latency with the both cases, with and without zebroids, reduces. Similarly, the normalized unsatisfied requests go down. Initially, the latency with zebroids is lower than the no-zebroids case. But as the storage is increased, the no-zebroids case starts outperforming the case with zebroids. This is because employing zebroids results in changes in the distribution of the data item replicas. Because of the finite duration of the traces, a steady-state for this distribution is never reached. So the changes in the number of data item replicas caused by replacements for the earlier requests have a detrimental effect on the latency for the future requests. With the no-zebroids case, the data item replicas are allocated as per the square-root

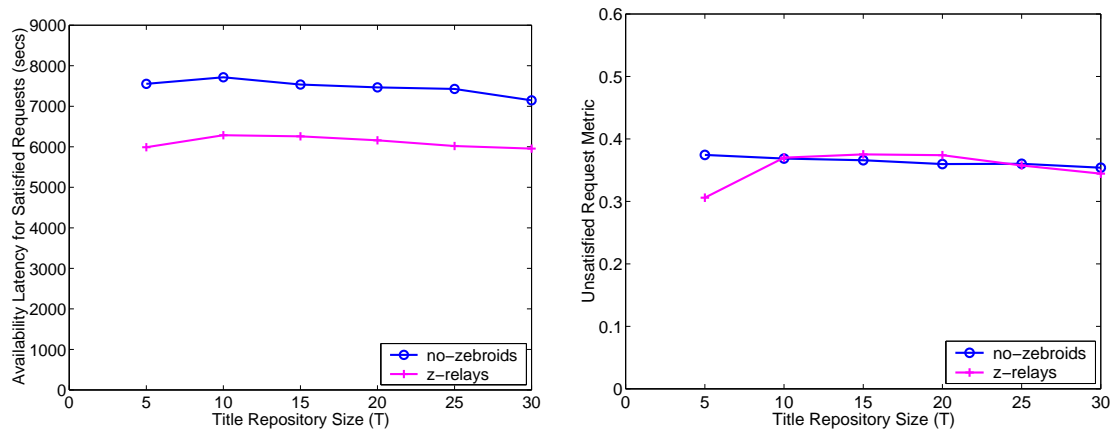


Figure 5.17: Aggregate availability latency and normalized unsatisfied requests with zebroids for the case when the ratio of  $T : \alpha$  is maintained as 5 : 1 and requests are issued as per a Zipf distribution at equal inter-arrival times.

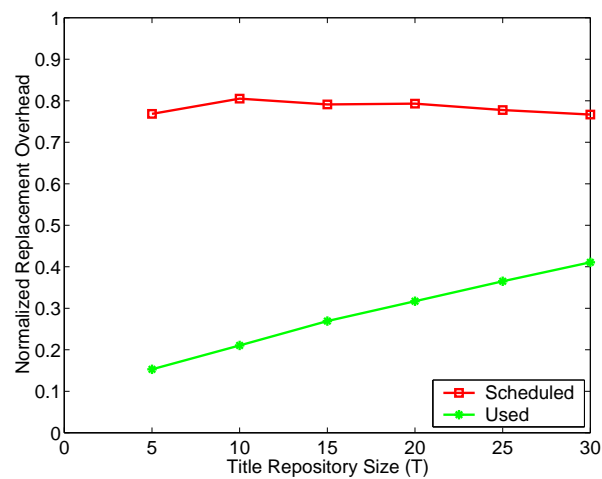


Figure 5.18: Replacement overhead incurred by employing zebroids when the ratio of  $T : \alpha$  is maintained as 5 : 1 and requests are issued as per a Zipf distribution at equal inter-arrival times.

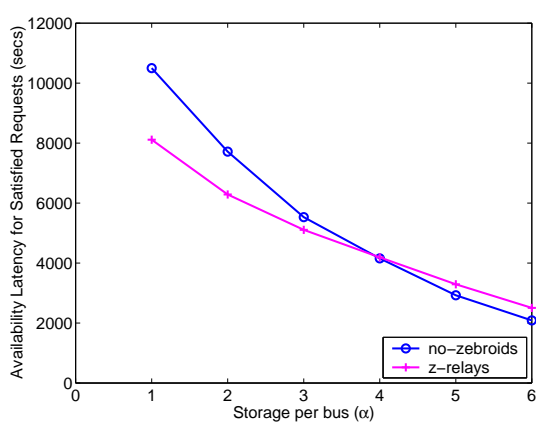
replication scheme and this distribution does not change. This provides a better latency performance on an average. Similar trends are seen in the unsatisfied requests metric beyond a storage per bus value of 2.

As more data item replicas are introduced into the system, lesser number of requests can benefit from employing zebroids. This is captured in Figure 5.19(d) where the average number of zebroids scheduled per request reduces with increase in the storage per bus. Recall that for a given request, a relay team of  $z$  zebroids is only scheduled if it provides a lower latency than that provided by the no-zebroids case.

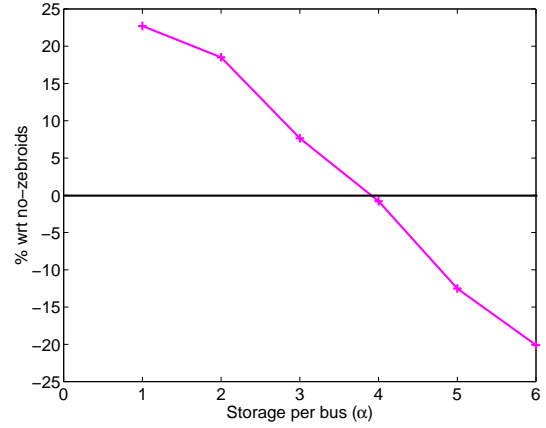
Figure 5.20 shows the performance with zebroids when the storage per bus is fixed at 3 and the data item repository size is increased. As the data item repository size is increased, lesser replicas are allocated per data item resulting in an increase in the latency as well as the unsatisfied requests. This is seen both in the cases with and without zebroids. Moreover, the increase in latency for satisfied requests without zebroids is much sharper, significant latency improvements can be obtained by employing zebroids with larger repositories (see 5.20(b)). This can be seen by the higher number of zebroids scheduled with a larger repository in Figure 5.20(d). However, if the repository size is very large then because of the constraint that at least one copy of every data item must be present in the network at all times, very few buses can be employed as zebroids, yielding a similar latency for the cases with and without zebroids.

## 5.9 Summary

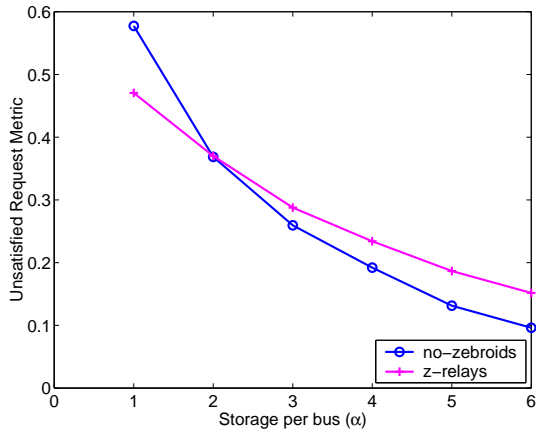
In this chapter, we examined the improvements in latency that can be obtained in the presence of data carriers, termed zebroids, that deliver a data item from a server to a client. We quantified the variation in availability latency as a function of a rich set of parameters such as car density, storage per car, data item repository size, and replacement policies employed by zebroids. Our key findings are as follows. A naive random replacement policy employed by the zebroids exhibits competitive latency benefits at a minimal replacement overhead. Zebroids continue to provide improvements even in the presence of lower accuracy in the predictions of the car routes. Improvements in latency obtained



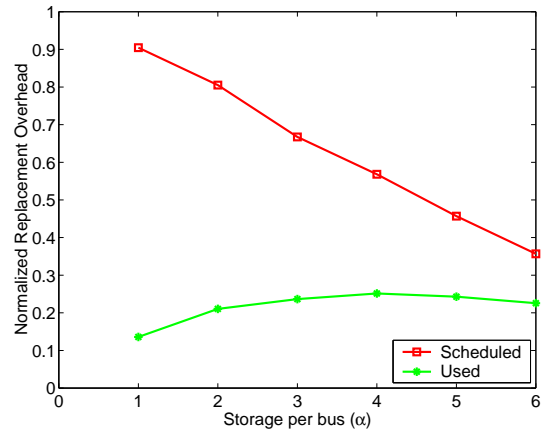
(a) Latency for satisfied requests



(b) Percentage wrt no-zebroids



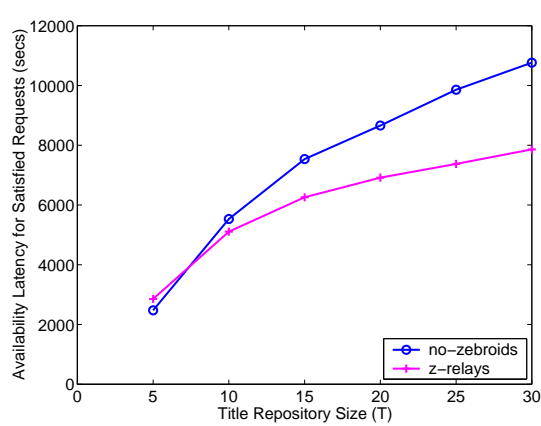
(c) Unsatisfied requests



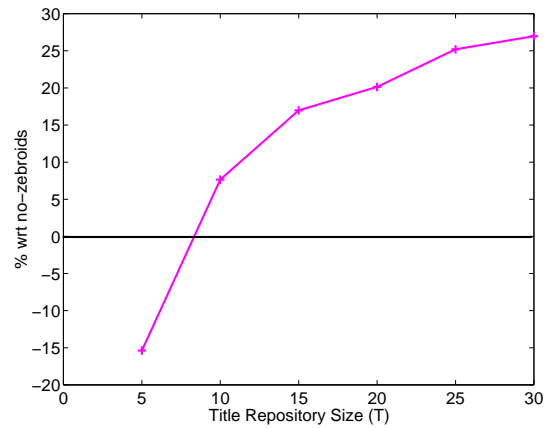
(d) Replacement overhead

Figure 5.19: Performance with zebroids as a function of the storage per car when the data item repository size is held constant at 10. Requests are issued as per a Zipf distribution at equal inter-arrival times.

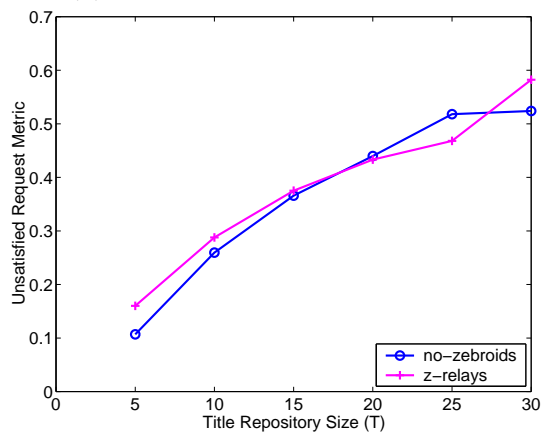




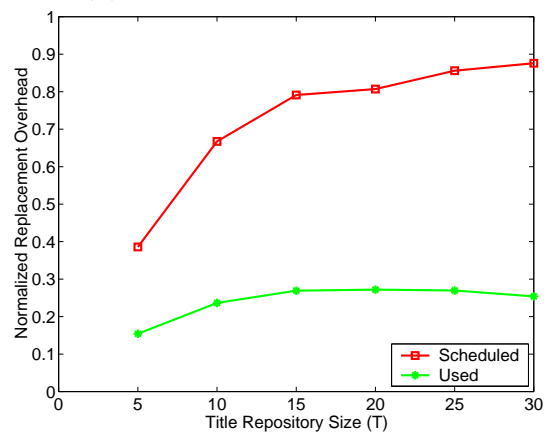
(a) Latency for satisfied requests



(b) Percentage wrt no-zebroids



(c) Unsatisfied requests



(d) Replacement overhead

Figure 5.20: Performance with zebroids as a function of the data item repository size when the storage per car is held constant at 3. Again, requests are issued as per a Zipf distribution at equal inter-arrival times.

with one-instantaneous zebroids are maximized with a uniform input popularity distribution of the data items. Also, for a given total system storage, presence of more cars with a low storage capacity yields higher improvements in latency when compared with fewer cars with a high storage capacity. The conclusions seem to be robust to the fact that the Markov mobility model may be derived from a underlying real city map. Finally, similar observations as obtained with the Markov model are realized from employing zebroids on traces collected from small vehicular test-bed (UMassDieselNet).

Extensions to zebroids that consider data items of larger size remains to be investigated. The larger data items may be divided into chunks and multiple relay team of zebroids may be scheduled to ensure that each chunk reaches the client at the appropriate time. Incorporation of explicit bandwidth constraints into the Markov model so that it can be better equipped for scenarios where multiple simultaneous requests are active in the system also remains a subject of future investigation. Also, zebroids may also be used for delivery of data items that carry delay sensitive information with a certain expiry. Extensions to zebroids that satisfy such application requirements present an interesting future research direction.

## Chapter 6

### Related Work

In this chapter, we provide a brief overview of the related work in the area. We first present a description of various system components that will be a part of an operational AutoMata application. Next, we provide a description of the various studies on replication in mobile ad-hoc networks. Subsequently, we present related works in the area of delay tolerant networks where data carriers like zebroids have been employed. Finally, we conclude with a concise account of some related studies in the area of Intelligent Transportation Systems (ITS) where our results may be directly applicable.

#### 6.1 Other Components of an AutoMata Application

The graphical user interface that displays the list of available titles with their associated latency may be complimented by a number of other components that are needed for realizing an AutoMata application. Below, we describe some typical components that may be part of such a system. Note that the related works described in this subsection are beyond the scope of this thesis, we present them here for providing a unified picture. Even though these components have been described in the introduction, we relist them here for completeness.

Once a user initiates display of a data item, an admission control component [18] ensures availability of both resources and the referenced data. Next, a data delivery scheduling technique [20] utilizes resources as a function of time to deliver the data item to a requesting AutoMata device. This component, may switch between several candidate

servers containing the referenced data item based on their proximity, current availability of resources, and network conditions. This component is tied closely to an ad-hoc network routing protocol which facilitates delivery of data between AutoMata devices. Example protocols are DSR [34], DSDV [47], AODV [48] to name a few. Another system component may monitor whether the system is providing a target AutoMata with the desired QoS and make adjustments as necessary. Besides the above components, there may be others responsible for addressing the security [64] and privacy [16] concerns of the user that may be mandatory for practical use of the system. Additionally, suitable physical and MAC layer optimizations may be needed to adapt to the wireless nature of the communication medium between the vehicles.

## 6.2 Replication in MANETs

Replication in MANETs has been explored in a wide variety of contexts. Several studies predict partitions in the network and pre-emptively replicate data to ensure continuous data availability. Li *et al.* [40, 38] use a group mobility model to predict the movement of the nodes and replicate pre-emptively to ensure existence of at least one server per group for continuous data availability. Our study is different in that not only does it use a different mobility model which is Markovian in nature, but the metric we seek to optimize is the latency until the first encounter of a replica of a requested data item. Our mobility model is more flexible because by suitably adjusting the transition probabilities between the various cells in the map it can approximate other mobility models such as highway, manhattan, random walk mobility model [4] and others. Moreover, we consider storage constraints per node and how available storage can be best utilized to improve availability latency. Storage aspects have not been considered in the studies mentioned above.

Hara [23] proposes three replica allocation methods, one that allocates replicas to nodes only on the basis of their local preference to data items, another that additionally considers the contents of the connected neighbors while performing the allocation to remove some redundancy and finally, one that discovers bi-connected components in the network topology and allocates replicas accordingly. The frequency of access of the nodes

to data items is known in advance and does not change. Moreover, the replica allocation takes place periodically in a specific period termed the relocation period. Several extensions to this work have been proposed where replica allocation methods have been extended to consider data items with periodic [25, 27] and aperiodic [30, 26] updates. Further extensions to the proposed replica allocation methods consider the stability of radio links [32], topology changes [33] and location history of the data item access log [28, 29]. In [31], the authors consider data items that are correlated by virtue of being requested simultaneously and present the performance of the replica allocation methods proposed earlier. All the above studies are based on simulations of the proposed replica allocation methods.

Our study differs from the above studies based on the initial proposals by Hara *et al.* [23] in the following ways. The above studies use a heuristic approach where the methods are evaluated via simulations. On the other hand, in our study, we propose an optimization formulation for the optimal replication strategy that minimizes aggregate availability latency. We analytically compute the optimal replication for sparse density scenarios and employ extensive simulations to study dense cases. Secondly, the above studies consider the number of requests satisfied as a fraction of the total requests as the prime metric of interest without regards to the latency per request. In our studies, we not only strive to increase the number of satisfied requests but also strive to minimize the availability latency per request. Thirdly, we do not assume the presence of any relocation period during which replica allocation takes place.

Related studies [51, 61, 24] have appeared in the context of cooperative caching in mobile ad-hoc networks where the objective is to use caching to reduce the mobile node's latency in accessing data items. The proposed techniques take into account the request pattern of the nodes and the topology of the network as the nodes move while making a caching decision. In our studies, nodes (cars) do not perform any caching. We simply calculate the number of replicas per data item that will minimize availability latency. These replicas are then placed across the cars uniformly at random, hence, car's local storage is fully utilized.

However, after the static replication schemes have allocated the replicas across the cars, we have experimented with zebroids that perform dynamic data reorganization to better equip the system storage to the active user requests. Since the zebroids are cars whose local storage is fully utilized, we use different cache replacement policies that yield different replica distributions over time. We are not aware of any other work that employs dynamic data reorganization, across a distributed storage environment comprised by mobile vehicles, to improve user latency.

### 6.3 Sparse Network Architectures

Recently, several novel and important studies such as ZebraNet [35], DakNet [46], Data Mules [53], Message Ferries [63], SWIM [54], and Seek and Focus [55] have analyzed factors impacting intermittently connected networks consisting of data carriers similar in spirit to zebroids. Table 6.1 provides an overview of these studies. Factors considered by each study are dictated by their assumed environment and target application. A novel characteristic of our study is the impact on availability latency for a given database repository of items. In future work, it may be useful to integrate these diverse studies along with our work under a comprehensive general model/framework that incorporates all possible factors, environmental characteristics, and application requirements.

The various studies can be characterized into reactive and proactive on the basis of whether the mobility of the nodes is controlled. In the reactive case, node movements are dictated by a specific mobility model and applications rely on the movement inherent in these nodes for data delivery. In the proactive case, the node movement can be adapted pro-actively to deliver data and also in some cases reduce data delivery latency. We first summarize the studies that use proactive schemes followed by a brief description of ones that employ reactive schemes.

Li and Rus [41] introduce a scheme that computes the optimal trajectory for relaying a message among nodes. However, for larger networks, supporting multiple simultaneous message transmissions using this algorithm is difficult because the scheduling problem becomes intractable. Along the same lines, in [63], special nodes called message ferries that follow non-random movement paths allow data delivery between other nodes whose

Study	Potentially Mobile Nodes	Mobility Model	Delivery	Energy	Delay	How many copies created?	Storage
ZebraNet [35]	All	Controlled + species dependent	Many to One	X		Many	X
DakNet [46]	One	Controlled	One to One			One	
Message Ferries [63]	All	Random + Controlled	One to One	X	X	One	
SWIM [54]	Most	Random without predictions	Many to Some		X	Many	
Data Mules [53]	Some	Random without predictions	Many to One			One	X
Seek and Focus [55]	All	Random without predictions	One to One	X	X	One	
Our Work	All	Random with predictions	Any to One		X	One or More	X

Table 6.1: Related studies on intermittently connected networks.

movements are governed by a random mobility model. The movement of message ferries can be controlled in order to obtain data from nearby source nodes and deliver it to the appropriate sink nodes when in vicinity. Message transmissions only take place via direct transmissions. In a follow-up work [37], latency-energy tradeoffs are demonstrated by the authors where a single ferry follows a known trajectory and the other nodes schedule their sleep/wakeup schedule in accordance with when the ferry will be in the vicinity. Comparison with reactive protocol like DSR indicate significant energy savings while suffering only a small drop in delay performance.

In the ZebraNet project [35], sensors are attached to zebras to study the wild life behavior. The sparse nature of the network prevents formation of a connected network. Hence, data is obtained from the sensors when humans drive by in a car or some other vehicle. Similarly, in the DakNet project [46], vehicles are used to transfer data between villages and cities using a store and forward mechanism. Our work with zebroids differs from all the above studies that can be categorized in the proactive realm, in that, it is reactive and employs a random walk mobility model. The movement of the various nodes (vehicles in our case) is dictated by this model and cannot be controlled explicitly as was utilized in the above studies.

Nodes that obey a random mobility model have been widely studied in the context of reactive schemes. There have been a large number of studies on the analysis of properties of random walks [2] on a torus. Some recent studies like [55, 53] have used the 2D random walk mobility model in the context of routing in intermittently connected mobile networks. Our mobility model is similar to that used in these studies.

In [53], DataMules are used to route data from static sensors to the base-stations in a sparse sensor network. In our studies, all nodes are mobile and have the same capabilities. In [55], using latency or meeting time and number of transmissions as the metrics of interest, the authors present comparisons of a randomized, utility based and an hybrid (seek and focus) approach with an optimal scheme via analysis as well as simulations. The metric of meeting time used in this study is analogous to our notion of availability latency. In a followup study [56], the authors introduce a new multi-copy routing algorithm where the source “sprays” a certain number of copies into the network and then waits until one of those copies meets the destination. Through analysis and simulations, this spray and wait algorithm is shown to have the lowest average delay and low transmission cost in terms of the number of packets when compared with alternate schemes. However, our study differs from the work by Spyropoulos *et al.* [55], in that, their study does not consider a storage constraint per node as is the case with zebroids. The storage constraint requires zebroids to make decisions about which data item replicas should be kept in local storage to minimize overall availability latency. Moreover, we do not expect energy to be a very constrained resource in a vehicular ad-hoc network, hence we have not consider number of transmissions as a metric in our studies. Finally, their study does not employ an interference model that will constrain the available bandwidths for data transfer in a vehicular ad-hoc network.

Some studies in the mobile infostation context have relied on replication and caching techniques to reduce data delivery latency. Small and Haas [54] propose the Shared Wireless Infostation Model (SWIM) where the infostation architecture is combined with the ad-hoc networking model. Here, the infostations act as data sinks. By replicating and hence spreading data across the mobile nodes data delivery latency at the infostations can be greatly reduced. This study uses a differential equation model to analytically determine



the time until the data is spread to all the nodes. Presence of static infostations, a different mobility model, an absence of a storage constraint per node and a different architectural framework and application makes this study significantly different from ours.

## **6.4 Intelligent Transportation Systems (ITS)**

In the ITS framework, the vehicular network is viewed as a MANET and messages are forwarded from one vehicle to another realizing several applications like vehicle accident notification broadcast, pre-emptive emergency vehicle arrival information etc [9, 10]. Car-Net [43] is a scalable ad hoc network system of cars using a grid location service and geographic routing to achieve scalability for applications such as traffic congestion and fleet tracking. Along similar lines, several other projects such as FleetNet [8], VGrid [3], DRIVE [12] have been proposed. Our work compliments these studies in that the lessons about replication under storage constraints and minimization of availability latency in vehicular networks are directly applicable to these systems. This is because our framework is fairly general, in that, examples of data items across which we seek to minimize latency may be highway repair notification messages etc.

## Chapter 7

### Conclusions

We briefly summarize the major contributions of this dissertation. The proposed AutoMata application for delivery of content such as audio or video clips to passengers in their vehicles which is the primary motivation of this thesis is novel. The problems tackled in this dissertation examine complementary aspects such as data discovery (chapter 3), data replication (chapter 4), and data delivery scheduling (chapter 5) in such an environment. Moreover, the metrics of interest in such an application (for example availability latency, overhead etc.), the questions we propose, our solution approach, and our proposed methodology for evaluation in itself serve as a guideline to future research when data from live vehicular test-beds will be available for testing out the behavior in real deployments. We have also presented a feasibility analysis to show that employing the cellular infrastructure as a control plane and vehicular ad-hoc peer-to-peer network as the data plane has sufficient bandwidth to realize such an application. Next, we summarize the key contributions of each of the three studies that are part of this thesis.

#### 7.1 PAVAN

In the PAVAN work presented in Chapter 3, the key contributions are as follows:

- We introduce PAVAN as a novel policy framework for addressing the availability problem to compute when different data items are available in an ad-hoc network of AutoMata devices

- We propose novel utility models to evaluate the effectiveness of the PAVAN predictions with regards to the false-positives, false-negatives, and true-positives.

## 7.2 Static Replication Schemes

In the static replication schemes study presented in Chapter 4, the key contributions are as follows:

- Given a data item repository, proposed an optimization formulation to minimize the average availability latency subject to a storage constraint per vehicle
- Analytically solved the optimization in the case of a sparse density of replicas yielding the square-root replication scheme as the optimum
- Obtained a mathematical expression that captures the behavior of availability latency as a function of the number of data item replicas
- Examined the relative performance of 3 popular replication schemes: linear, square-root, and random for a large number of parameter settings:
  - Data item size = 1 and unbounded client trip duration
  - Data item size = 1 and finite client trip duration
  - Data item size > 1 and unbounded client trip duration
  - Data item size > 1 and finite client trip duration
- Validated main results employing vehicular movements dictated by an underlying map of the San Francisco Bay Area
- Evaluated the latency performance of the replication schemes using traces obtained from a bus-based DTN test-bed called UMassDieselNet [11]. Provided an equivalence between the Markov model and the mobility model represented by the traces

## 7.3 Zebroids

Finally, in the zebroids study presented in Chapter 5, the key contributions are as follows:

- Introduced the concept of zebroids as data carriers and proposed a modified Bellman Ford’s algorithm to yield the optimum delivery schedule for the relay team of zebroids that minimizes latency for a given client request
- Quantified the variation in availability latency with zebroids as a function of a rich set of parameters such as car density, storage per car, data item repository size, popularity of data items, client trip duration, number of zebroids employed, and replacement policies employed by zebroids.
- For a sparse density of data item replicas, proposed and validated a mathematical formulation to capture the improvements in latency with zebroids when compared to the latency provided by static replication schemes
- Evaluation of the performance of different replacement policies at zebroids that yielded that a random replacement scheme provides superior performance
- Changes in popularity of the data items do not impact the latency gains obtained with one-instantaneous zebroids.
- Validated main results with zebroids using a map of the San Francisco Bay area to dictate and constrain the movements of the vehicles in accordance with the location of major freeways captured by the equivalent Markov model
- Evaluated the performance employing zebroids using traces obtained from the bus-based UMassDieselNet test-bed.

We acknowledge that realistic validation of the Markov model is far from complete. In that regards, much additional work remains to demonstrate the practical applicability of the Markovian approach to realize an AutoMata based application. This represents a concrete direction for future doctoral dissertations.

However, this dissertation does present a concrete step toward understanding the different nuances involved in realizing an application in a vehicular ad-hoc network. The

huge parameter space involved makes it extremely challenging to come up with comprehensive mathematical models that can capture system performance. Starting from first principles and tying the different pieces together as we develop enough understanding of the same presents a promising approach in this direction. In many ways, this thesis along with other missing pieces of the AutoMata application will serve as a proof of concept that it is now realistic to start thinking about deployment of on-demand delivery applications for passengers in their vehicles. New standards such as the IEEE 802.16 WiMax further compliment this thesis in that they may be used to bolster both the control and the data plane, thereby alleviating some of request traffic in such a network.

## Bibliography

- [1] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. *SIGCOMM Comput. Commun. Rev.*, 34(4):121–132, 2004.
- [2] D. Aldous and J. Fill. Reversible markov chains and random walks on graphs. Manuscript under preparation, <http://stat.www.berkeley.edu/users/aldous/RWG/book.html>.
- [3] J. Anda, J. LeBrun, D. Ghosal, C-N. Chuah, and H. M. Zhang. Vgrid: Vehicular ad hoc networking and computing grid for intelligent traffic control. In *IEEE Vehicular Technology Conference*, Stockholm, Sweden, May 2005.
- [4] F. Bai, N. Sadagopan, and A. Helmy. Important: A framework to systematically analyze the impact of mobility on performance of routing protocols for adhoc networks. In *INFOCOM*, 2003.
- [5] F. Bar, W. Baer, S. Ghandeharizadeh, and F. Ordonez. Automata. Video at <http://www.youtube.com/watch?v=nrRFbLRjeZM>.
- [6] A. Bar-Noy, I. Kessler, and M. Sidi. Mobile Users: To Update or Not to Update. In *Proc. IEEE Infocom*, 1994.
- [7] S. Bararia, S. Ghandeharizadeh, and S. Kapadia. Evaluation of 802.11a for Streaming Data in Ad-hoc Networks. In *ASWN Boston, MA*, 2004.
- [8] M. Bechler, W. Franz, and L. Wolf. Mobile internet access in fleetnet. In *In Fachtagung Kommunikation in verteilten Systemen*, Leipzig, Germany, 2003.
- [9] L. Briesemeister and G. Hommel. Role-based multicast in highly mobile but sparsely connected ad hoc networks. In *Proceedings of MobiHOC*, pages 45–50, August 2000.
- [10] L. Briesemeister, L. Schafers, and G. Hommel. Disseminating Messages among Highly Mobile Hosts based on Inter-Vehicle Communication. In *Proceedings of IEEE Intelligent Vehicle Symposium*, pages 522–527, October 2000.
- [11] J. Burgess, B. Gallagher, D. Jensen, and B. Levine. MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networking. In *Proc. IEEE Infocom*, April 2006.
- [12] D. Carrega, K. Gaudin, A. Held, R. Kroh, and H. Muiyal. Delivering location-aware multimedia services with the drive service-provision architecture: A case study. In *3GIS Conference*, Athens, Greece, July 2001.

- [13] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. In *SIGCOMM*, pages 177–190. ACM Press, 2002.
- [14] D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 134–146, New York, NY, USA, 2003. ACM Press.
- [15] A. Dan, D. Dias, R. Mukherjee, D. Sitaram, and R. Tewari. Buffering and Caching in Large-Scale Video Servers. In *COMPCON*, 1995.
- [16] F. Dotzer. Privacy issues in vehicular ad hoc networks. In *VANET '05: Proceedings of the second ACM workshop on Vehicular ad hoc networks*, New York, NY, USA, 2005. ACM Press.
- [17] S. Ghandeharizadeh and T. Helmi. An Evaluation of Alternative Continuous Media Replication Techniques in Wireless Peer-to-Peer Networks. In *MobiDE, in conjunction with MobiCom'03*, September 2003.
- [18] S. Ghandeharizadeh, T. Helmi, S. Kapadia, and B. Krishnamachari. Admission Control and QoS for Continuous Media Displays in Mobile Ad-Hoc Networks of Devices. In *DMS*, 2004.
- [19] S. Ghandeharizadeh, S. Kapadia, and B. Krishnamachari. Comparison of Replication Strategies for Content Availability in C2P2 networks. In *6th International Workshop on Mobile Data Management*, May 2005.
- [20] S. Ghandeharizadeh, S. Kapadia, and B. Krishnamachari. Server replication techniques for display of continuous media in mobile ad hoc networks. Technical report, Department of Computer Science, University of Southern California, 2005.
- [21] S. Ghandeharizadeh and B. Krishnamachari. C2p2: A peer-to-peer network for on-demand automobile information services. In *DEXA Workshops*, pages 538–542, 2004.
- [22] IEEE 802.16 Working Group. Emerging IEEE 802.16 WirelessMAN Standards for Broadband Wireless Access. *Intel Technology journal*, Vol. 08, No. 03, 2004.
- [23] T. Hara. Effective Replica Allocation in Ad Hoc Networks for Improving Data Accessibility. In *IEEE INFOCOM*, pages 1568–1576, 2001.
- [24] T. Hara. Cooperative caching by mobile clients in push-based information systems. In *CIKM'02*, 2002.
- [25] T. Hara. Replica allocation in ad hoc networks with periodic data update. In *MDM '02: Proceedings of the Third International Conference on Mobile Data Management*, pages 79–86, Washington, DC, USA, 2002. IEEE Computer Society.
- [26] T. Hara. Replicating Data with Aperiodic Update in Ad Hoc Networks. In *Proc. of IASTED Int'l Conf. on Communications, Internet and Information Technology (CIIT'02)*, pages 242–247, 2002.

- [27] T. Hara. Replica allocation methods in ad hoc networks with data update. *ACM/Kluwer Journal on Mobile Networks and Applications (MONET)*, 8(4):343–354, 2003.
- [28] T. Hara. Location management of data items in mobile ad hoc networks. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 1174–1175, New York, NY, USA, 2005. ACM Press.
- [29] T. Hara. Strategies for data location management in mobile ad hoc networks. In *Proc. of IEEE Int'l Conf. on Parallel and Distributed Systems (ICPADS'05)*, 2005.
- [30] T. Hara and S. Madria. Dynamic data replication using aperiodic updates in mobile ad-hoc networks. In *Proc. of International Conference on Database Systems for Advanced Applications (DASFAA 2004)*, pages pp.869–881, Jeju Island, Korea, 2004.
- [31] T. Hara, N. Murakami, and S. Nishio. Replica allocation for correlated data items in ad hoc sensor networks. *SIGMOD Rec.*, 33(1):38–43, 2004.
- [32] T. Hara, Y.-H.Loh, and S.Nishio. Data Replication Methods Based on the Stability of Radio Links in Ad Hoc Networks. In *Proc. of International Workshop on Mobility in Databases and Distributed Systems (MDDS)*, pages 969–973, 2003.
- [33] H. Hayashi, T. Hara, and S. Nishio. A Replica Allocation Method Adapting to Topology Changes in Ad Hoc Networks. In *Proc. of International Conference on Database and Expert Systems Applications (DEXA)*, 2005.
- [34] D. Johnson, D. Maltz, and J. Broch. *DSR The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks*, chapter 5, pages 139–172. Addison-Wesley, 2001.
- [35] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet. *SIGARCH Comput. Archit. News*, 30(5):96–107, 2002.
- [36] H. Jun, M. Ammar, and E. Zegura. Power Management in Delay Tolerant Networks: A Framework and Knowledge-Based Mechanisms. In *Proc. of IEEE SECON*, September 2005.
- [37] H. Jun, W. Zhao, M. Ammar, C. Lee, and E. Zegura. Trading latency for energy in wireless ad hoc networks using message ferrying. In *Third IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'05)*, 2005.
- [38] K. and B. Li. Efficient and guaranteed service coverage in partitionable mobile ad-hoc networks. In *INFOCOM*, 2002.
- [39] J. Lee. Peer to peer file sharing systems: What matters to the end users? Technical report, Center for Coordination Science, Sloan School of Management, 2002.



- [40] B. Li and K. Wang. NonStop: Continuous Multimedia Streaming in Wireless Ad Hoc Networks with Node Mobility. *IEEE journal on Selected Areas in Communications*, Vol. 21, No.10, December 2003.
- [41] Q. Li and D. Rus. Sending messages to mobile users in disconnected ad-hoc wireless networks. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 44–55, New York, NY, USA, 2000. ACM Press.
- [42] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and Replication in Unstructured Peer-to-Peer Networks. In *Proceedings of the 16th annual ACM International Conference on Supercomputing*, 2002.
- [43] R. Morris, J. Jannotti, F. Kaashoek, J. Li, and D. Decouto. Carnet: a scalable ad hoc wireless network system. In *EW 9: Proceedings of the 9th workshop on ACM SIGOPS European workshop*, pages 61–65, New York, NY, USA, 2000. ACM Press.
- [44] California Department of Transportation. Caltrans. <http://www.dot.ca.gov/>.
- [45] E. O’Neil, P. O’Neil, and G. Weikum. The lru-k page replacement algorithm for database disk buffering. In *ACM SIGMOD*, pages 297–306, 1993.
- [46] A. Pentland, R. Fletcher, and A. Hasson. DakNet: Rethinking Connectivity in Developing Nations. *Computer*, 37(1):78–83, 2004.
- [47] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsv) for mobile computers. In *ACM SIGCOMM’94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.
- [48] C. Perkins and E. Royer. Ad-hoc on-demand distance vector routing. In *2nd IEEE Workshop on Mobile Computing Systems and Applications, WMCSA '99, February 25-26, 1999, New Orleans, Louisiana, USA*, pages 90–100. IEEE, February 1999.
- [49] E. Royer, P. Melliar-Smith, and L. Moser. An analysis of the optimum node density for ad hoc mobile networks. In *IEEE International Conference on Communications (ICC)*, volume 3, pages 857–861, 2001.
- [50] I. Rubin and C. Choi. Impact of the Location Area Structure on the Performance of Signaling Channels in Wireless Cellular Networks. *IEEE Communications Magazine*, pages 108–115, February 1997.
- [51] F. Sailhan and V. Issarny. Cooperative caching in ad hoc networks. In *The 4th International Conference on Mobile Data Management, MDM’03*, 2003.
- [52] P. Santi. The critical transmitting range for connectivity in mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 4(3):310–317, 2005.
- [53] R. Shah, S. Roy, S. Jain, and W. Brunette. Data Mules: Modeling and Analysis of a Three-Tier Architecture for Sparse Sensor Networks. *Elsevier Ad Hoc Networks Journal*, 1, September 2003.

- [54] T. Small and Z. J. Haas. The shared wireless infostation model: a new ad hoc networking paradigm (or where there is a whale, there is a way). In *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 233–244, New York, NY, USA, 2003. ACM Press.
- [55] T. Spyropoulos, K. Psounis, and C. Raghavendra. Single-Copy Routing in Intermittently Connected Mobile Networks. In *SECON*, April 2004.
- [56] T. Spyropoulos, K. Psounis, and C. Raghavendra. Spray and wait: An efficient routing scheme for intermittently connected mobile networks. In *WDTN Workshop in association with ACM SIGCOMM*, 2005.
- [57] A. Tanenbaum. *Modern Operating Systems, 2nd Edition, Chapter 4, Section 4.4*. Prentice Hall, 2001.
- [58] S. Tewari and L. Kleinrock. Proportional replication in peer-to-peer networks. In *IEEE INFOCOM*, 2006.
- [59] A. Vahdat and D. Becker. Epidemic routing for partially-connected ad hoc networks. Technical report, Department of Computer Science, Duke University, 2000.
- [60] J. Wolf, P. Yu, and H. Shachnai. DASD Dancing: A Disk Load Balancing Optimization Scheme for Video-on-Demand Computer. In *Proceedings of ACM SIGMETRICS*, pages 157–166, May 1995.
- [61] L. Yin and G. Cao. Supporting cooperative caching in ad hoc networks. In *INFOCOM 2004*, 2004.
- [62] W. Yuen and R. Yates. Optimum transmit range and capacity of mobile infostation networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(3):45–47, 2003.
- [63] W. Zhao, M. Ammar, and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, pages 187–198, New York, NY, USA, 2004. ACM Press.
- [64] D. Zhou. *Security issues in ad hoc networks*. CRC Press, Inc., Boca Raton, FL, USA, 2003.
- [65] G. K. Zipf. *The Psychobiology of Language*. Boston: Houghton-Mifflin, 1935.
- [66] M. Zonoozi and P. Dassanayake. User Mobility Modeling and Characterization of Mobility Pattern. *IEEE Journal on Selected Areas in Communications*, 15:1239–1252, September 1997.