An Evaluation of Availability Latency in Carrier-based Vehicular Ad-Hoc Networks

Shahram Ghandeharizadeh[1], Shyam Kapadia[1] and Bhaskar Krishnamachari[1,2]
[1]Department of Computer Science
[2]Department of Electrical Engineering
University of Southern California
Los Angeles, CA 90089, USA
Email:shahram,kapadia,bkrishna@usc.edu

Abstract

On-demand delivery of audio and video clips in peer-to-peer vehicular ad-hoc networks is an emerging area of research. Our target environment uses data carriers, termed zebroids, where a mobile device carries a data item on behalf of a server to a client thereby minimizing its availability latency. In this study, we quantify the variation in availability latency with zebroids as a function of a rich set of parameters such as car density, storage per device, repository size, and replacement policies employed by zebroids. Using analysis and extensive simulations, we gain novel insights into the design of carrier-based systems. Significant improvements in latency can be obtained with zebroids at the cost of a minimal overhead. These improvements occur even in scenarios with lower accuracy in the predictions of the car routes. Two particularly surprising findings are: (1) a naive random replacement policy employed by the zebroids shows competitive performance, and (2) latency improvements obtained with a simplified instantiation of zebroids are found to be robust to changes in the popularity distribution of the data items.

1 Introduction

Technological advances in areas of storage and wireless communications have now made it feasible to envision on-demand delivery of data items, for e.g., video and audio clips, in vehicular peer-to-peer networks. In prior work, Ghandeharizadeh *et al.* [11] introduce the concept of vehicles equipped with a Car-to-Car-Peer-to-Peer device, termed AutoMata, for in-vehicle entertainment. The notable features of an AutoMata include a mass storage device offering hundreds of gigabytes (GB) of storage, a fast processor, and several types of networking cards. Even with today's 500 GB disk drives, a repository of diverse entertainment content may exceed the storage capacity of a single AutoMata. Such repositories constitute the focus of this study. To exchange data, we assume each AutoMata is configured with two types of networking cards: 1) a low-bandwidth networking card with a long radio-range in the order of miles that enables an AutoMata device to communicate with a nearby cellular or WiMax base station, and 2) a high-bandwidth networking card with a limited radio-range in the order of hundreds of feet.

The high bandwidth connection supports data rates in the order of tens to hundreds of Megabits per second [4] and represents the ad hoc peer to peer network between the vehicles. This is labelled as the data plane and, as suggested by its name, is employed to exchange data items between devices. The low-bandwidth connection serves as the control plane, enabling AutoMata devices to exchange meta-data with one or more centralized servers. This connection offers bandwidths in the order of tens and hundreds of Kilobits per second. The centralized servers, termed dispatchers, compute schedules of data delivery along the data plane using the provided meta-data. These schedules are transmitted to the participating vehicles using the control plane. The technical feasibility of such a two-tier architecture is presented in [9], with preliminary results to demonstrate that the bandwidth of the control plane is sufficient for exchange of control information needed for realizing such an application.

In a typical scenario, an AutoMata device presents a passenger with a list of data items[1], showing both the name of each data item and its availability latency. The latter, denoted as δ, is defined as the earliest time at which the client encounters a copy of its requested data item. A data item is available immediately when it resides in the local storage of the AutoMata device serving the request. Due to storage constraints, an AutoMata may not store the entire repository. In this case, availability latency is the time from when the user issues a request until when the AutoMata encounters another car containing the referenced data item. (The terms car and AutoMata are used interchangeably in this study.)

The availability latency for an item is a function of the current location of the client, its destination and travel path, the mobility model of the AutoMata equipped vehicles, the number of replicas constructed for the different data items, and the placement of data item replicas across the vehicles. A method to improve the availability latency is to employ data carriers which transport a replica of the requested data item from a server car containing it to a client that requested it. These data carriers are termed 'zebroids'.

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.[2] 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 4.1 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.

In this study, 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.

Our main findings are as follows. A naive random replacement policy employed by the zebroids shows competitive performance in terms of availability latency. With such a policy, substantial improvements in latency can be obtained with zebroids at a minimal replacement overhead. In more practical scenarios, where the dispatcher has inaccurate information about the car routes, zebroids continue to provide latency improvements. A surprising result is that changes in popularity of the data items do not impact the latency gains obtained with a simple instantiation of z-relay zebroids called one-instantaneous zebroids (see Section 4.1). This study suggests a number of interesting directions to be pursued to gain better understanding of design of carrier-based systems that improve availability latency.

The rest of this paper is organized as follows. Section 2 provides a brief overview of the related work in the area. Section 3 provides an overview of the terminology along with the factors that impact availability latency in the presence of zebroids. Section 4 describes how the zebroids may be employed. Section 5 provides details of the analysis methodology employed to capture the performance with zebroids. Section 6 describes the details of the simulation environment used for evaluation. Section 7 enlists the key questions examined in this study and answers them via analysis and simulations. Finally, Section 8 presents brief conclusions and future research directions.

2 Related Work

In this section, we provide a brief overview of the related work in the area. First, 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

[1] Without loss of generality, the term data item might be either traditional media such as text or continuous media such as an audio or video clip.

[2] 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.

tolerant networks where data carriers such as zebroids have been employed.

2.1 Replication in MANETs

Replication in MANETs has been explored in a wide variety of contexts. Hara [12] proposes three replica allocation methods. The first one that allocates replicas to nodes only on the basis of their local preference to data items. The second technique extends the first by considering the contents of the connected neighbors while performing the allocation to remove some redundancy. The last technique discovers bi-connected components in the network topology and allocates replicas accordingly. The frequency of access to data items is known in advance and does not change. Moreover, the replica allocation is performed 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 [14, 16] and aperiodic [19, 15] updates. Further extensions to the proposed replica allocation methods consider the stability of radio links [21], topology changes [22] and location history of the data item access log [17, 18]. In [20], 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. Related studies [28, 34, 13] 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. All the above studies are based on simulations of the proposed replica allocation methods.

Our study differs from the above in the following ways. We use cars as data carriers (zebroids) to reduce the latency of a currently active request. The choice of a zebroid and the process of transferring the requested item from a server to an appropriate zebroid is made possible by the presence of a two-tier architecture comprising a low bandwidth control plane and a high bandwidth data plane [9]. This architecture was used to predict the list of data items that will be encountered by a client during its journey. In prior work [10], we evaluated the latency performance of three popular static replication schemes: linear, square-root and random in a vehicular ad-hoc network. The square-root scheme provided a competitive performance across a large parameter space. In this study, we extend our prior work by employing zebroids as data carriers to improve the client observed latency. Once a static square-root replication scheme has allocated the replicas across the cars, we study the role of dynamic data reorganization to better equip the system storage to the active user requests. Since the zebroids are AutoMatas whose local storage is fully utilized, we use different cache replacement policies that yield different replica distributions as a function of time. We are not aware of related work that employs dynamic data reorganization, across a distributed storage environment comprised by mobile vehicles, to improve user latency.

2.2 Delay Tolerant Networks

Recently, several novel and important studies such as ZebraNet [23], DakNet [26], Data Mules [29], Message Ferries [35], SWIM [30], and Seek and Focus [31] have analyzed factors impacting intermittently connected networks consisting of data carriers similar in spirit to zebroids. Table 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.

Below, we provide an overview of the different projects and their target applications. In ZebraNet, data sensed by sensors attached to zebras is collected by humans as they drive by in a vehicle. In DakNet, vehicles are used to transport data between villages and cities using a store and forward mechanism. Message Ferries capture a more generalized scenario where the movement of the ferries can be controlled to carry data from a source node to a destination node. With Data Mules, intermediate carriers that follow a random walk mobility model are used to carry data from static sensors to base-stations. When all nodes move as per this mobility model, end-to-end routing can be performed using a Seek and Focus strategy presented in [31]. With the Shared Wireless Infostation Model (SWIM), data collected by sensors on whales is replicated when two sensors are in the vicinity of each other and ultimately relayed to a small

Study	Potentially Mobile Nodes	Mobility Model	Delivery	Energy Efficiency	Optimize Delay	How many copies created?	Storage Constraint
ZebraNet [23]	All	Controlled + species dependent	Many to One	X		Many	X
DakNet [26]	One	Controlled	One to One			One	
Message Ferries [35]	All	Random + Controlled	One to One	X	X	One	
SWIM [30]	Most	Random without predictions	Many to Some		X	Many	
Data Mules [29]	Some	Random without predictions	Many to One			One	X
Seek and Focus [31]	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 1: Related studies on intermittently connected networks.

number of static on-shore base-stations when the whales come to the surface.

We now describe how the previous studies are different from ours, detailing the novel features of our study. None of these studies predict the future movement trajectory of the nodes to accomplish data delivery as we do in our study with zebroids. Moreover, while DakNet, Message Ferries, and Seek and Focus employ end-to-end data delivery, studies such as Data Mules, ZebraNet and SWIM require data from many nodes to be sent to a single (or a few) base-station(s) yielding a many-to-one or many-to-some delivery mode. In our study, the mode of data delivery is any-to-one since a request for a certain data item can be satisfied by any node that stores that item.

Studies such as ZebraNet, Message Ferries, and Seek and Focus seek to optimize the energy usage, especially with sensors. This is not a constraint with a AutoMata environment. Instead, we seek to optimize meeting time or latency to request satisfaction in the presence of storage constraints. This metric is similar to that considered in the seek and focus study which ignores storage constraints per node. Also, even though the Data Mules study considers a storage constraint per node, it does not optimize for latency subject to this constraint. Instead, that study provides the scaling behavior needed in the storage per sensor (and mule) to maintain the data delivery ratio over a certain threshold.

While reliable data delivery is required with DakNet, Seek and Focus, and Message Ferries, storage constraints may cause data loss, preventing Data Mules and Zebranet from achieving the same. In our study, the notion of reliable data delivery has a different connotation. This is because for each data item request, the client stipulates a deadline, which is the maximum amount of time it is willing to wait to encounter a copy of the requested item. Zebroids are employed to improve the availability latency for each request, increasing the likelihood of the client's deadline being met.

Finally, studies like ZebraNet, DakNet and Data Mules can tolerate high delay as long as the appropriate data is delivered. However, the Seek and Focus and Ferries approach use delay as a metric of interest. Also, with zebroids, SWIM, and ZebraNet, multiple replicas of the data are employed to facilitate reduction in the delivery latency. Replicas are created when two nodes encounter each other. While there are no storage constraints with SWIM, with ZebraNet, excess data is dropped from the sensors using a drop-tail like scheme. Moreover, none of the studies mentioned above use a two-tier architectural framework like ours that facilitates intelligent selection of data carriers and creation of appropriate data replicas to reduce latency for currently active requests. With storage constraints, we also explore a wide variety of replacement policies deployed at the zebroids.

In prior work, Ghandeharizadeh *et al.* [8] examine the performance of replacement policies that employ location demographic information about the areas native to a zebroid in order to perform a replacement. The key extension here is the use of z-relay zebroids.

Database Parameters	
T	Number of data items.
S_i	Size 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	Number of replicas for data item i
n	Characterizes a particular replication scheme.
$\overline{\delta_i}$	Average availability latency of data item i
δ_{agg}	Aggregate availability latency, $\delta_{agg} = \sum_{j=1}^{T} \overline{\delta_j} \cdot f_j$
AutoMata System Parameters	
G	Number of cells in the map (2D-torus).
N	Number of AutoMata devices in the system.
α	Storage capacity per AutoMata.
γ	Trip duration of the client AutoMata.
S_T	Total storage capacity of the AutoMata system, $S_T = N \cdot \alpha$.

Table 2: Terms and their definitions

3 Overview and Terminology

Table 2 summarizes the notation of the parameters used in the paper. Below we introduce some terminology used in the paper.

Assume a network of N AutoMata-equipped cars, each with storage capacity of α bytes. The total storage capacity of the system is S_T=$N \cdot \alpha$. There are T data items in the database, each with size S_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 γ.

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

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

The exponent n characterizes a particular replication technique. A square-root replication scheme is realized when $n = 0.5$ [6]. This serves as the base-line for comparison with the case when zebroids are deployed. R_i is normalized to a value between 0 and 1. The number of replicas for data item i, denoted as r_i, is: $r_i = \min\left(N, \max\left(1, \lfloor \frac{R_i \cdot N \cdot \alpha}{S_i} \rfloor\right)\right)$. This captures the case when at least one copy of every data item must be present in the ad-hoc network at all times. In cases where a data item may be lost from the ad-hoc network, this equation becomes $r_i = \min\left(N, \max\left(0, \lfloor \frac{R_i \cdot N \cdot \alpha}{S_i} \rfloor\right)\right)$. In this case, a request for the lost data item may need to be satisfied by fetching the item from a remote server.

The availability latency for a data item i, denoted as δ_i, is defined as the earliest time at which a client AutoMata will find the first replica of the item accessible to it. If this condition is not satisfied, then we set δ_i to γ. This indicates that data item i was not available to the client during its journey. Note that since there is at least one replica in the system for every data item i, by setting γ to a large value we ensure that the client's request for any data item i will be satisfied. However, in most practical circumstances γ may not be so large as to find every data item.

We are interested in the availability latency observed across all data items. Hence, we augment the average availability latency for every data item i with its f_i to obtain the following weighted availability latency (δ_{agg}) metric: $\delta_{agg} = \sum_{i=1}^{T} \overline{\delta_i} \cdot f_i$

We have identified the following critical parameters that affect availability latency in the presence of zebroids: (i) title database size (T), (ii) car density (N), (iii) storage per car (α), (iv) trip duration (γ), (v) replacement scheme employed

by the zebroids (see Section 4.2), and (vi) prediction accuracy of the car routes. We study the variation in availability latency as a function of each of these parameters in the presence of zebroids. Next, we present our solution approach describing how zebroids are selected.

4 Solution Approach

4.1 Zebroids

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. 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 [33] under the assumptions of infinite storage and no interference.

A simple instantiation of z-relay zebroids occurs when $z = 1$ and the client's request triggers a transfer of a copy of the requested data item from a server to a zebroid in its vicinity. Such a zebroid is termed **one-instantaneous zebroid**. 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 replicas might be computed using Equation 1. 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, majority of the storage capacity of a car will be exhausted.

4.2 Carrier-based Replacement policies

The replacement policies considered in this paper 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 [32]. We present below a list of carrier-based replacement policies employed in our study which are adapted from different page replacement policies.

1. **Least recently used (LRU)** LRU-K [25] 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 [3], 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, δ_{agg}, percentage improvement in δ_{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.

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 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 [31, 29] 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 [3, 27, 36].
- 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,

[3] The terms eviction and replacement are used interchangeably.

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 γ.

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 δ_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 $G >> r_i$): $\overline{\delta_i} = \frac{cGlogG}{r_i}$

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

$$\delta_{agg}(no - zeb) = \sum_{i=1}^{T} \frac{f_i \cdot c \cdot G \cdot \log G}{r_i} = \sum_{i=1}^{T} \frac{f_i \cdot C}{r_i} \tag{2}$$

5.1 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 $\overline{N_i^c}$ be the expected total number of nodes that are in the same cell as any of the r_i servers. Then, we have

$$\overline{N_i^c} = (N - r_i) \cdot (1 - (1 - \frac{1}{G})^{r_i}) \tag{3}$$

In the analytical model, we assume that $\overline{N_i^c}$ new replicas are created, so that the total number of replicas is increased to $r_i + \overline{N_i^c}$. 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,

$$\delta_{agg}(zeb) = \sum_{i=1}^{T} \frac{f_i \cdot c \cdot G \cdot \log G}{r_i + \overline{N_i^c}} = \sum_{i=1}^{T} \frac{f_i \cdot C}{r_i + \overline{N_i^c}} \tag{4}$$

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 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 4.1). 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.

5.2 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.* [31]. 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,

$$\overline{\delta_i} = \frac{C}{N+1-r_i} \sum_{m=r_i}^{N} \sum_{k=r_i}^{m} \frac{1}{k \cdot (N+1-k)} \tag{5}$$

Evaluating the above expression, we get,

$$\overline{\delta_i} = \frac{C}{N+1} \cdot \frac{1}{N+1-r_i} \cdot [N \cdot \log \frac{N}{r_i} - \log (N+1-r_i)] \tag{6}$$

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

$$\begin{aligned} \delta_{agg}(zeb) &= \sum_{i=1}^{T} [f_i \cdot \frac{C}{N+1} \cdot \frac{1}{N+1-r_i} \cdot \\ &\quad (N \cdot \log \frac{N}{r_i} - \log (N+1-r_i))] \end{aligned} \tag{7}$$

6 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 γ, δ have been discretized and expressed in terms of the number of time slots.

- A 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. α represents the number of storage slots per AutoMata. Each storage slot stores one data item. γ 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 γ. δ 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 5×5 2D-torus with γ 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 with a mean of 0.27. This distribution is shown to correspond to sale of movie theater tickets in the United States [7]. 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 [10]. 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 γ, 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.

7 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.

7.1 How does a replacement scheme employed by a zebroid impact availability latency?

For illustration, we present 'scale-up' experiments where one-instantaneous zebroids are employed (see Figure 1). By scale-up, we mean that α and N are changed proportionally to keep the total system storage, S_T, constant. Here, we

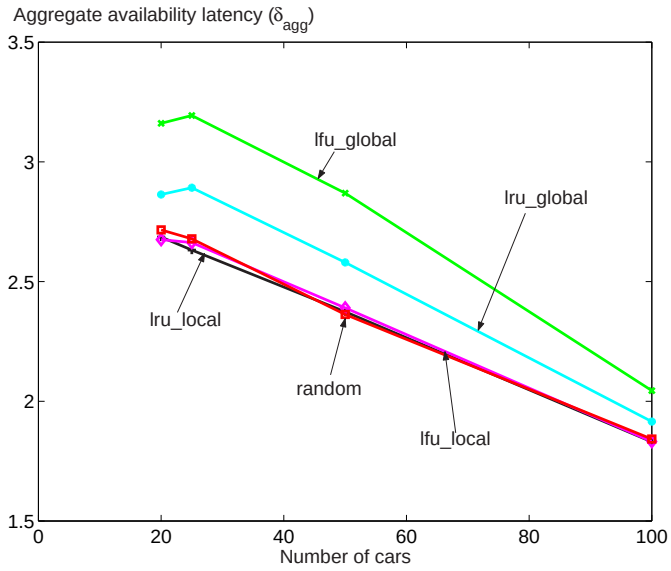


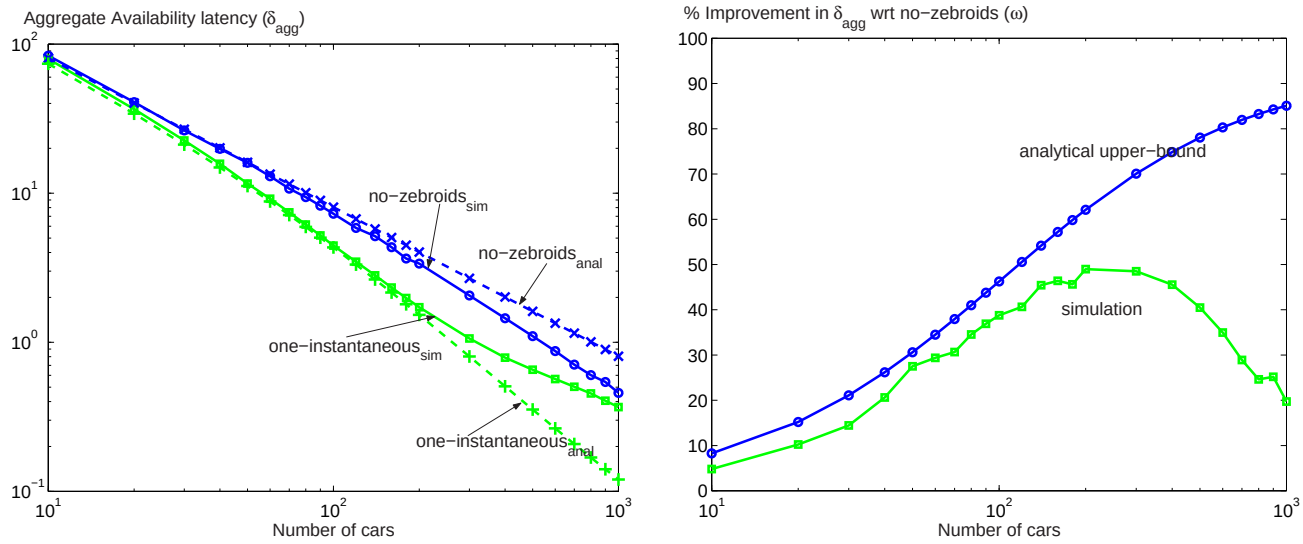
Figure 1: Shows the availability latency when employing one-instantaneous zebroids as a function of (N,α) values, when the total storage in the system is kept fixed, $S_T = 200$.

set $T = 50$ and $S_T = 200$. We choose the following values of (N,α) = {(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.

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 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.



2.a) δ_{agg} 2.b) ω

Figure 2: Shows the latency performance with one-instantaneous zebroids via simulations along with the analytical approximation for a 10×10 torus with $T = 10$.

7.2 Do zebroids provide significant improvements in availability latency?

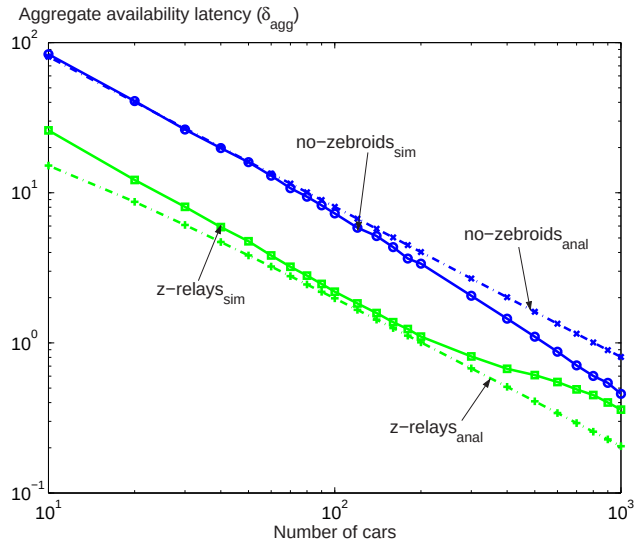
We find that in many scenarios employing zebroids provides substantial improvements in availability latency.

7.2.1 Analysis

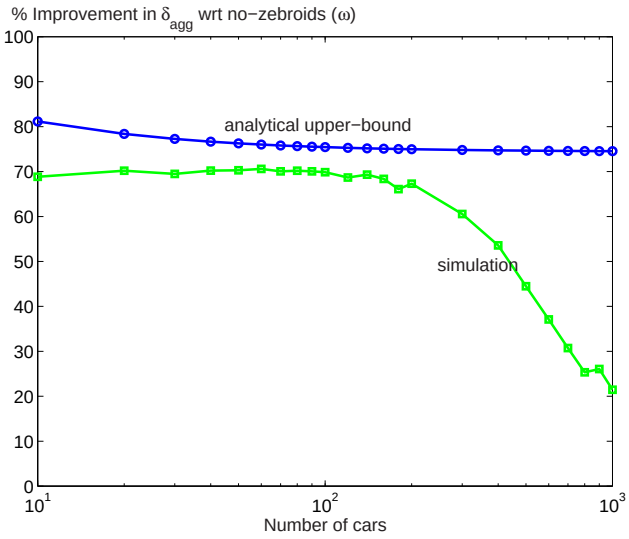
We first consider the case of one-instantaneous zebroids. Figure 2.a shows the variation in δ_{agg} as a function of N for $T = 10$ and $\alpha = 1$ with a 10×10 torus using Equation 4. Both the x and y axes are drawn to a log-scale. Figure 2.b show the % improvement in δ_{agg} obtained with one-instantaneous 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 2 and 4, closely matches the simulation results. However, as N increases, the sparseness assumption for which the analysis is valid, namely $N << G$, 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 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 δ_{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.



3.a) δ_{agg}



3.b) ω

Figure 3: Shows the latency performance with z-relay zebroids via analysis and simulations for a 10×10 torus with $T = 10$.

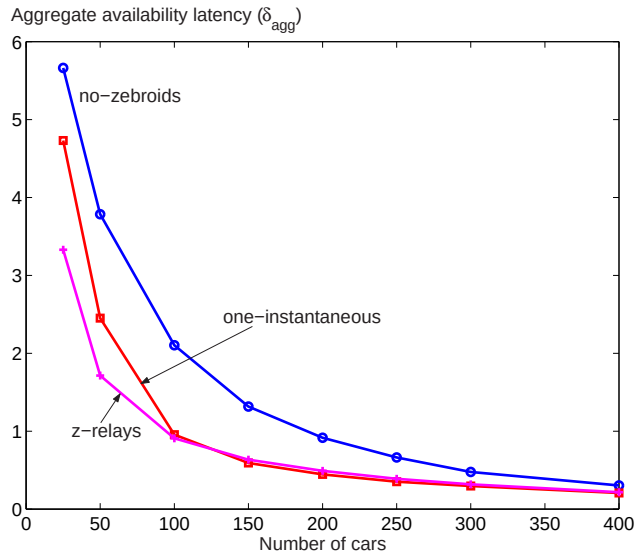
7.2.2 Simulation

We conduct simulations to examine the entire storage spectrum obtained by changing car density N or storage per car α 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 α by capturing the variation of N while keeping α 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 4 and 5 respectively capture the two cases mentioned above. With Figure 4.b, keeping α constant, initially increasing car density has higher latency benefits because increasing N introduces more zebroids in the system. As N is further increased, ω 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 δ_{agg} down. The net effect results in reduction in ω for larger values of N. Similar trends are seen by keeping N constant and increasing α (see Figure 5.b).

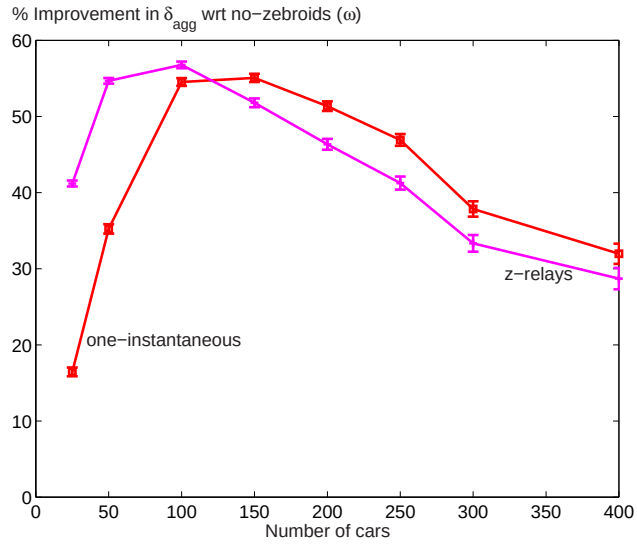
The trends mentioned above are similar to that obtained from the analysis. However, somewhat counter-intuitively with relatively higher system storage, z-relay zebroids provide slightly lower improvements in latency as compared to one-instantaneous zebroids. We speculate that this is due to the different 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. We are currently exploring suitable choices of parameters that can capture these changing replica distributions.

7.3 What is the overhead/cost associated with achieving improvements in latency with zebroids?

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



4.a



4.b

Figure 4: Depicts the latency performance with both one-instantaneous and z-relay zebroids as a function of the car density when $\alpha = 2$ and $T = 25$.

7.3.1 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,

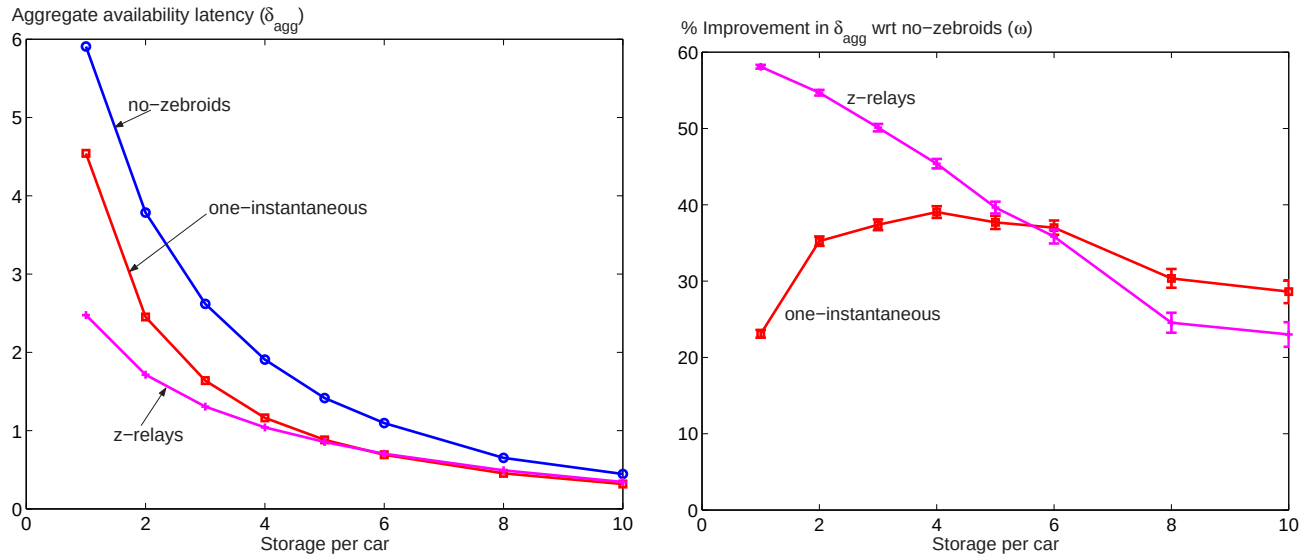
$$Overhead/request = 1 \cdot P(X_{\overline{N_i^c}} < X_{r_i}) + 0 \cdot P(X_{r_i} \leq X_{\overline{N_i^c}}) \tag{8}$$

$$Overhead/request = \frac{\frac{r_i}{C}}{\frac{r_i}{C} + \frac{\overline{N_i^c}}{C}} = \frac{r_i}{r_i + \overline{N_i^c}} \tag{9}$$

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 2, we get

$$Overhead/request = 1 - \frac{G}{G + N \cdot (1 - k \cdot \alpha)} \tag{10}$$

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 α to go down. This implies that a higher replacement overhead is incurred for higher N and lower α values. Moreover, when $r_i = N$, this means that



5.a 5.b

Figure 5: Shows latency performance with both one-instantaneous and z-relay zebroids as a function of α when $N = 50$ and $T = 25$.

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.

7.3.2 Simulation

Figure 6 shows the replacement overhead with one-instantaneous zebroids when (N,α) 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 (α). Figures 7.a and Figure 7.b respectively indicate the replacement overhead incurred with both one-instantaneous and z-relay zebroids when α 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 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 δ_{agg} (see Figures 4.b and 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

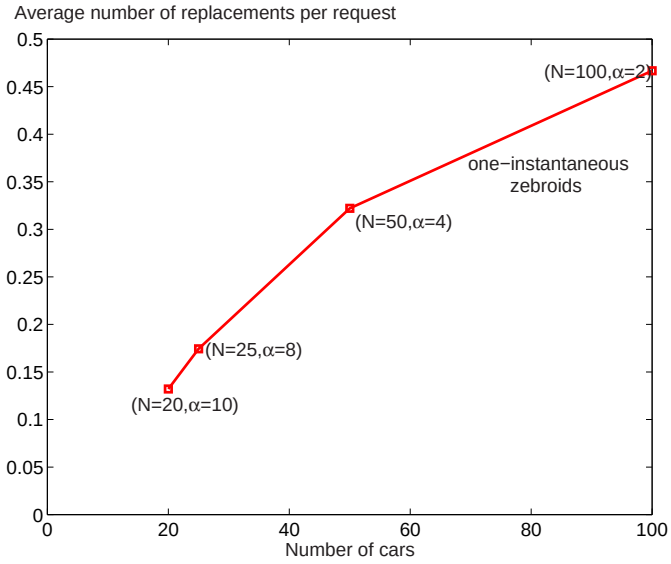


Figure 6: Captures replacement overhead when employing one-instantaneous zebroids as a function of (N,α) values, when the total storage in the system is kept fixed, $S_T = 200$.

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.

7.4 What happens to the availability latency with zebroids in more practical scenarios with inaccuracies in the car route predictions?

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.

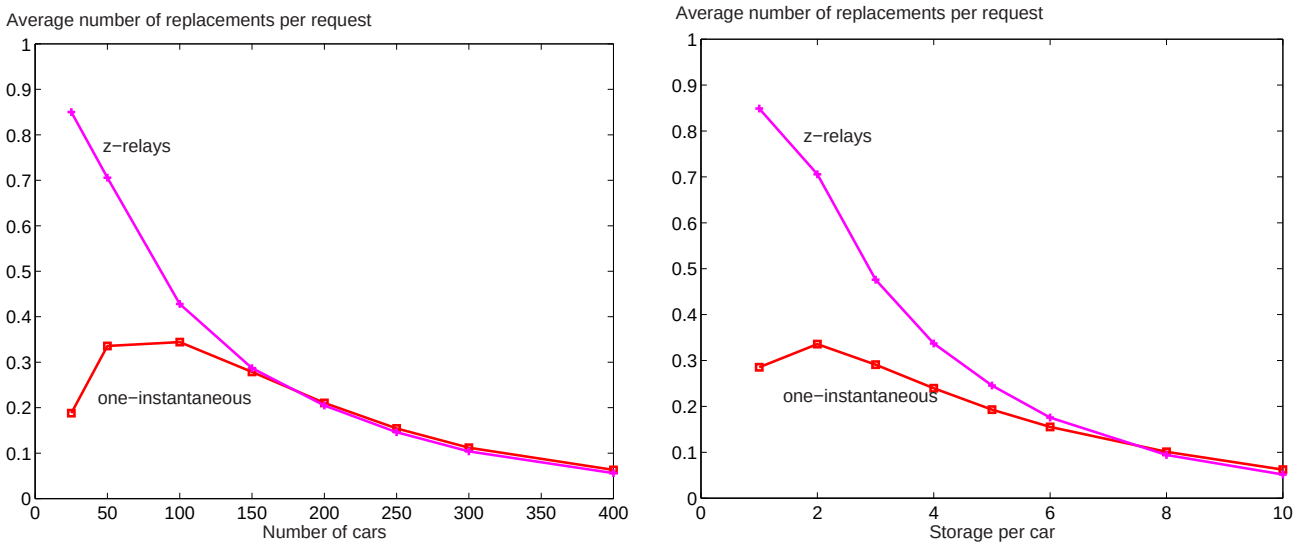
7.4.1 Analysis

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,

$$\delta_{agg}^{err} = p \cdot \delta_{agg}(zeb) + (1-p) \cdot \delta_{agg}(no-zeb) \tag{11}$$

$$\delta_{agg}^{err} = p \cdot \delta_{agg}(zeb) + (1-p) \cdot \frac{C}{r_i} \tag{12}$$

Expressions for $\delta_{agg}(zeb)$ can be obtained from Equations 4 (one-instantaneous) or 7 (z-relay zebroids).



7.a 7.b

Figure 7: Shows the replacement overhead with zebroids for the cases when N is varied keeping $\alpha = 2$ and α is varied keeping $N = 50$.

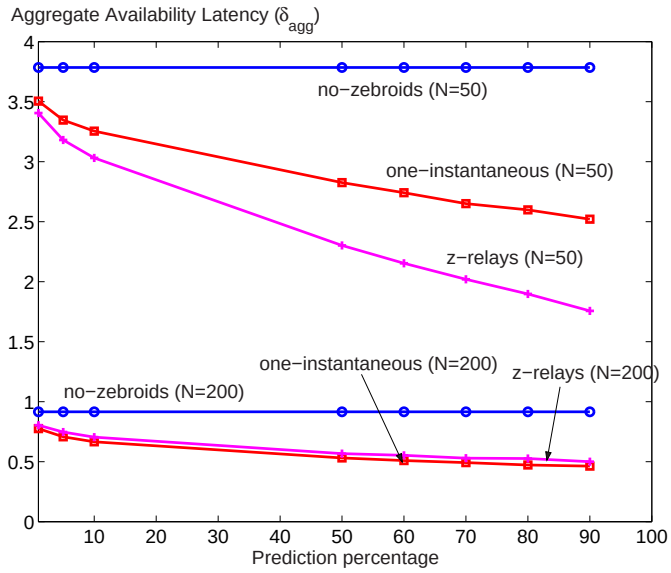


Figure 8: Shows δ_{agg} for different car densities as a function of the prediction accuracy metric with $\alpha = 2$ and $T = 25$.

7.4.2 Simulation

Figure 8 shows the variation in δ_{agg} as a function of this route prediction accuracy metric. We observe a smooth reduction in the improvement in δ_{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.* [24] 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.

7.5 Under what conditions are the improvements in availability latency with zebroids maximized?

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.

7.5.1 Analysis

The fractional difference (labelled ω) in the latency between the no-zebroids and one-instantaneous zebroids is obtained from equations 2, 3, and 4 as

$$\omega = \frac{\sum_{i=1}^{T} \frac{f_i \cdot C}{r_i} - \sum_{i=1}^{T} \frac{f_i \cdot C}{r_i + (N - r_i) \cdot \left(1 - \left(1 - \frac{1}{G}\right)^{r_i}\right)}}{\sum_{i=1}^{T} \frac{f_i \cdot C}{r_i}} \tag{13}$$

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,

$$\omega = 1 - \frac{\sum_{i=1}^{T} \frac{r_i}{1 + \frac{N - r_i}{G}}}{\sum_{i=1}^{T} r_i} \tag{14}$$

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

$$\textbf{Maximize } \omega, \textbf{ subject to } \sum_{i=1}^{T} r_i = S_T \tag{15}$$

Theorem 1 *With a square-root replication scheme, improvements obtained with one-instantaneous zebroids are independent of the input popularity distribution of the data items.*

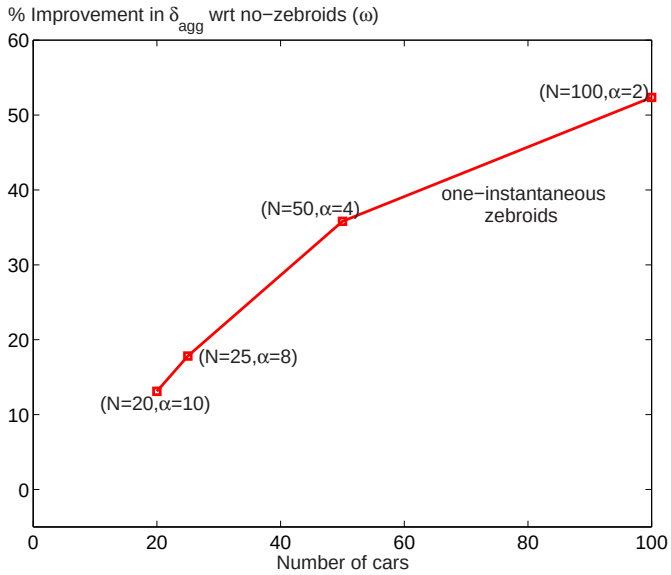


Figure 9: Captures improvement in availability latency with one-instantaneous zebroids as a function of (N,α) values, when the total storage in the system is kept fixed, $S_T = 200$.

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

$$Max \left\{ 1 - \sum_{i=1}^{T} \frac{G \cdot r_i}{N \cdot (G + N - r_i)} + \lambda \cdot \left[\sum_{i=1}^{T} r_i - N \right] \right\} \tag{16}$$

We solve for r_i to obtain:

$$r_i = G + N - G \cdot \sqrt{\frac{G + N}{G \cdot N \cdot \lambda}} \tag{17}$$

Note that in Equation 17, while r_i is independent of i, it is the same for all titles i. It can be verified that the maximum ω occurs at this value of r_i since $\frac{\partial^2 \omega}{\partial r_i^2} < 0$. This implies that as long as the constraint $\sum_{i=1}^{T} r_i = N$ is satisfied, improvements in latency can be maximized. □

7.5.2 Simulation

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

7.6 What combinations of car-density and storage per car offer the highest improvements in latency with zebroids for the same total system storage?

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.

7.6.1 Analysis

Consider the case of one-instantaneous zebroids. The fractional difference (labelled ω) in δ_{agg} between the no-zebroids and one-instantaneous zebroids cases is obtained in Equation 13. Using the approximation $(1-x)^n \simeq 1 - n \cdot x$ for small x, we simplify the above equation to get,

$$\omega = 1 - \frac{\sum_{i=1}^{T} \frac{G \cdot f_i}{r_i \cdot (G+N-r_i)}}{\sum_{i=1}^{T} \frac{f_i}{r_i}} \tag{18}$$

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 18 are constant. Also, with increasing N, ω increases. However, for a constant $N \cdot \alpha$, if N increases, α 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.

7.6.2 Simulation

Figure 9 validates the insight obtained from the analysis in that the improvements in latency go up with higher N and lower α 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.

7.7 What is the impact of different trip durations and repository sizes on availability latency in the presence of zebroids?

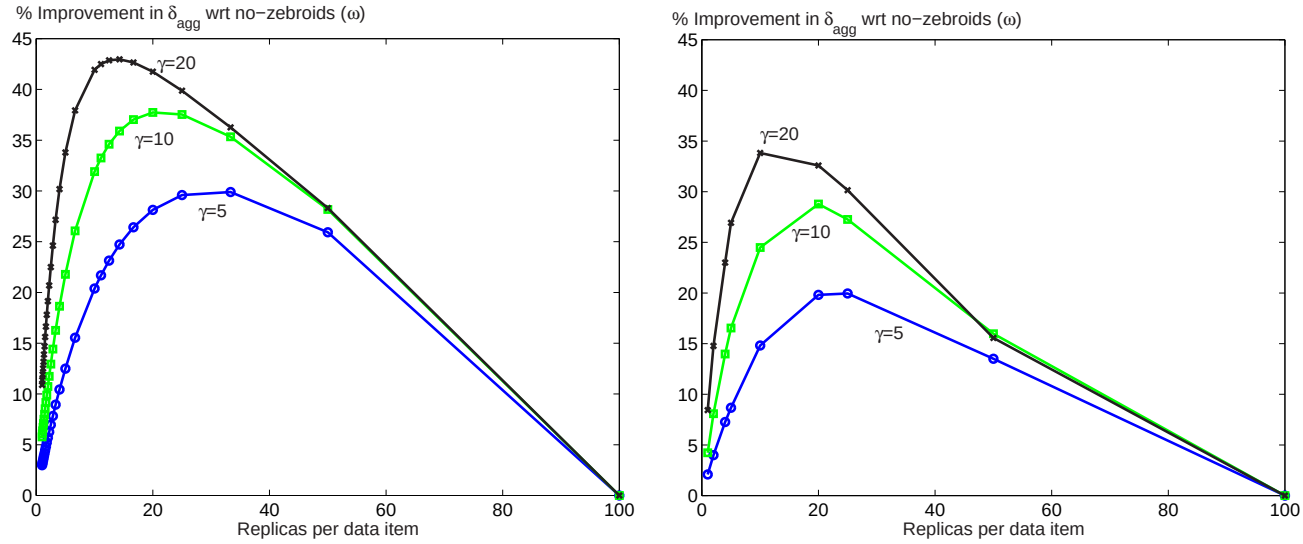
7.7.1 Analysis

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 γ, similar to that considered in the simulation environment ($\gamma = 10$). The availability latency, δ_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:

$$P(\delta_i > t) = \lambda \cdot \exp(-\lambda \cdot t) \tag{19}$$

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



10.a) Analysis 10.b) Simulation

Figure 10: Shows improvement in δ_{agg} with one-instantaneous zebroids for different client trip durations in case of 10×10 torus with a fixed car density, N = 100.

$$\overline{\delta_i} = \int_0^{\gamma} x \cdot \lambda \cdot \exp(-\lambda \cdot t) dx + \int_{\gamma}^{\infty} \gamma \cdot \lambda \cdot \exp(-\lambda \cdot t) dx \tag{20}$$

Hence, we get

$$\overline{\delta_i} = \frac{c \cdot G \cdot \log G}{r_i} \cdot [1 - \exp(\frac{-\gamma \cdot r_i}{c \cdot G \cdot \log G})] \tag{21}$$

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

$$\delta_{agg}(no-zeb) = \sum_{i=1}^{T} f_i \cdot \frac{c \cdot G \cdot \log G}{r_i} \cdot [1 - \exp(\frac{-\gamma \cdot r_i}{c \cdot G \cdot \log G})] \tag{22}$$

In the presence of one-instantaneous zebroids, the aggregate availability latency can be obtained using a procedure similar to that used in Section 5.1, giving

$$\delta_{agg}(zeb) = \sum_{i=1}^{T} f_i \cdot \frac{c \cdot G \cdot \log G}{(r_i + \overline{N_i^c})} \cdot [1 - \exp(\frac{-\gamma \cdot (r_i + \overline{N_i^c})}{c \cdot G \cdot \log G})] \tag{23}$$

The above equations yield the improvements in latency with finite trip duration. We consider a 10×10 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 10.a and 10.b capture the latency performance obtained via analysis and simulations respectively for different trip durations.

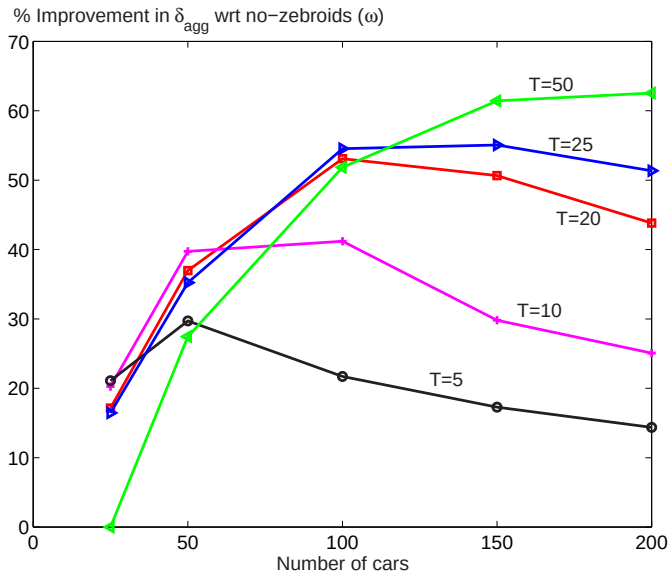


Figure 11: Shows improvement in availability latency as a function of the car density for different repository sizes with $\alpha = 2$ and $\gamma = 10$.

We now describe the behavior of the curves for a given finite trip duration γ. 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 γ increases, peak improvements in latency are obtained with higher T.

7.7.2 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 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.

We have also considered environments where there are a different mix of vehicles such as cars and buses. Buses are universal servers that carry the entire data item repository. The performance in terms of latency and replacement overhead in such an environment is presented in the Appendix (see Section 10).

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 heterogenous 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. We intend to consider repositories with different-sized data items as part of our future work.

8 Conclusions and Future Research Directions

In this study, 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, title database 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 with one-instantaneous zebroids are independent of the input distribution of the popularity 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.

A contribution of this work is to identify a significant research direction for each question presented and discussed in Section 7. Comprehensive analytical models that enable a study of the different factors for each question remain an important future research topic. Below we summarize some key future research directions we intend to pursue. First, we are currently investigating the effect of different replica distributions on availability latency in the presence of zebroids. Second, we intend to explore alternate definitions for the parameter that controls the prediction accuracy of the car routes known at the dispatcher. Third, to better reflect reality we would like to validate the observations obtained from this study with some real world simulation traces of vehicular movements (for example using CORSIM [1]). This will also serve as a validation for the utility of the Markov mobility model used in this study. We are currently analyzing the performance of zebroids on a real world data set comprising of an ad-hoc network of buses moving around a small neighborhood in Amherst [5].Fourth, we intend to include bandwidth constraints in our model so that it can be better equipped for scenarios where multiple simultaneous requests are active in the system. Finally, 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 presents an interesting future research direction.

9 Acknowledgments

This research was supported in part by NSF grants numbered CNS-0435505 (NeTS NOSS), CNS-0347621 (CAREER), IIS-0307908 and an Annenberg fellowship.

References

[1] Federal Highway Administration. Corridor simulation (corsim/tsis). Version 5.1, http://www.ops.fhwa.dot.gov/trafficanalysistools/cors im.htm.

[2] D. Aldous and J. Fill. Reversible markov chains and random walks on graphs. Under preparation.

[3] A. Bar-Noy, I. Kessler, and M. Sidi. Mobile Users: To Update or Not to Update. In *Proc. IEEE Infocom*, 1994.

[4] S. Bararia, S. Ghandeharizadeh, and S. Kapadia. Evaluation of 802.11a for streaming data in ad-hoc networks. In *ASWN Boston, MA*, 2004.

[5] J. Burgess, B. Gallagher, D. Jensen, and B. Levine. MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networking. In *Proc. of IEEE Infocom*, April 2006.

[6] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. *SIGCOMM Comput. Commun. Rev.*, 32(4):177–190, 2002.

[7] A. Dan, D. Dias, R. Mukherjee, D. Sitaram, and R. Tewari. Buffering and Caching in Large-Scale Video Servers. In *Proc. of the 40th IEEE Computer Society International Conference (COMPCON)*, 1995.

[8] S. Ghandeharizadeh and S. Kapadia. An Evaluation of Location-Demographic Replacement Policies for Zebroids. In *IEEE Consumer Communications and Networking Conference (CCNC)*, 2006.

[9] S. Ghandeharizadeh, S. Kapadia, and B. Krishnamachari. PAVAN: A Policy Framework for Content Availabilty in Vehicular Ad-hoc Networks. In *VANET*, New York, NY, USA, 2004. ACM Press.

[10] S. Ghandeharizadeh, S. Kapadia, and B. Krishnamachari. Comparison of Replication Strategies for Content Availability in C2P2 networks. In *Proc. of 6th International Conference on Mobile Data Management (MDM)*, May 2005.

[11] S. Ghandeharizadeh and B. Krishnamachari. C2P2: A Peer-to-Peer Network for On-Demand Automobile Information Services. In *First International Workshop on Grid and Peer-to-Peer Computin g Impacts on Large Scale Heterogeneous Distributed Database Systems (Globe)*, 2004.

[12] T. Hara. Effective Replica Allocation in Ad Hoc Networks for Improving Data Accessibility. In *Proc. of IEEE Infocom*, pages 1568–1576, 2001.

[13] T. Hara. Cooperative caching by mobile clients in push-based information systems. In *Proc. of the eleventh international conference on Information and knowledge management (CIKM)*, 2002.

[14] T. Hara. Replica allocation in ad hoc networks with periodic data update. In *Proc. of the Third International Conference on Mobile Data Management(MDM)*, pages 79–86, Washington, DC, USA, 2002. IEEE Computer Society.

[15] 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)*, pages 242–247, 2002.

[16] 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.

[17] T. Hara. Location management of data items in mobile ad hoc networks. In *Proc. of the 2005 ACM symposium on Applied computing (SAC)*, pages 1174–1175, New York, NY, USA, 2005. ACM Press.

[18] 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)*, 2005.

[19] 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)*, pages pp.869–881, Jeju Island, Korea, 2004.

[20] 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.

[21] 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.

[22] 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.

[23] 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*, 2002.

[24] 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.

[25] 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.

[26] A. Pentland, R. Fletcher, and A. Hasson. DakNet: Rethinking Connectivity in Developing Nations. *Computer*, 37(1):78–83, 2004.

[27] 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.

[28] F. Sailhan and V. Issarny. Cooperative caching in ad hoc networks. In *Proc. of 4th International Conference on Mobile Data Management (MDM)*, 2003.

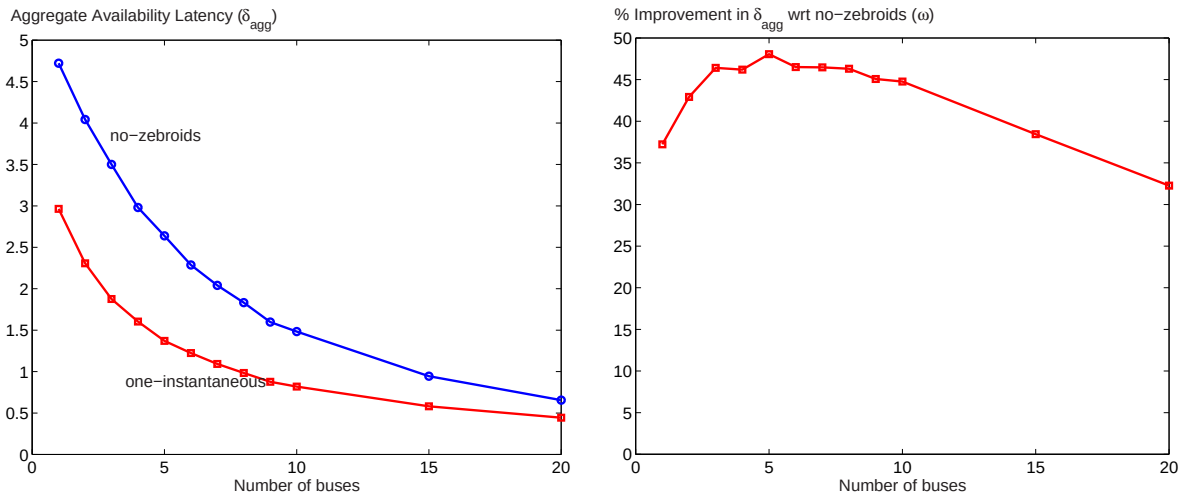
[29] R. Shah, S. Roy, S. Jain, and W. Brunette. Data mules: Modeling and analysis of a three-tier architecture for sparse sensor networks. *Elsevier Ad Hoc Networks Journal*, 1, September 2003.

[30] 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 *Proc. of the 4th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, pages 233–244, New York, NY, USA, 2003. ACM Press.

[31] T. Spyropoulos, K. Psounis, and C. Raghavendra. Single-Copy Routing in Intermittently Connected Mobile Networks. In *Proc. of IEEE SECON*, April 2004.

[32] A. Tanenbaum. *Modern Operating Systems, 2nd Edition, Chapter 4, Section 4.4* . Prentice Hall, 2001.

[33] A. Vahdat and D. Becker. Epidemic routing for partially-connected ad hoc networks. Technical report, Department of Computer Science, Duke University, 2000.



12.a 12.b

Figure 12: Shows δ_{agg} and ω as a function of the number of buses with a 5x5 torus for the random replacement policy when deploying one-instantaneous zebroids. Here $N = 50$, $\alpha = 2$, and $T = 50$.

[34] L. Yin and G. Cao. Supporting cooperative caching in ad hoc networks. In *Proc. of IEEE Infocom*, 2004.

[35] W. Zhao, M. Ammar, and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *Proc. of the 5th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, pages 187–198, New York, NY, USA, 2004. ACM Press.

[36] 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.

10 Appendix

We investigated other environments with the presence of special vehicles equipped with high capacity storage devices that contain the entire data item repository. These vehicles, termed 'buses', may not be zebroids because they contain the entire repository. Here, we study the variation in δ_{agg} and replacement overhead as we increase the number of buses in the system. Except for the additional presence of buses, the simulation setup is similar to that described in Section 6.

Figures 12.a and 12.b show the variation in δ_{agg} as a function of the number of buses for a 5×5 torus when $N = 50$. The performance is compared against the no-replacements case. Figure 13 shows the trend in the number of replacements as a function of the number of buses.

Increase in number of buses reduces availability latency. This is identical to the trend observed in Figures 4 and 5. The same behavior is obtained by increasing the number of buses as was obtained by varying N and keeping α constant and vice-versa. Here, however, as we increase the number of buses, the number of servers increases. It does not change the number of zebroids. Note that every bus increases the total number of replicas per data item by 1. The possibility of finding a suitable zebroid that can transport the requested data item to the client sooner becomes more difficult as the ratio of the buses relative to cars increases beyond a certain threshold, since the number of servers increases. This threshold was approximately 10% in our experiments (see Figure 12.b).

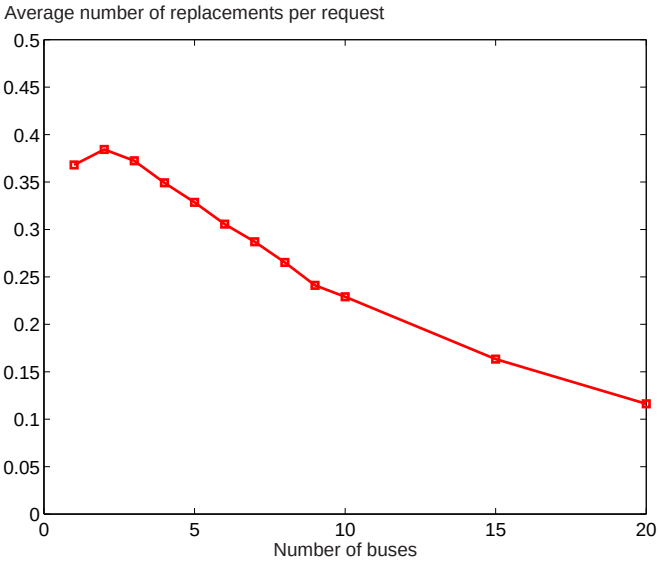


Figure 13: Shows the replacement overhead of the random policy as a function of the number of buses when deploying one-instantaneous zebroids. Here $N = 50$, $\alpha = 2$, and $T = 50$.