

Zebroids: Carrier-based Replacement Policies to Minimize Availability Latency in Vehicular Ad-hoc Networks

Shahram
Ghandeharizadeh
Dept of Computer Science
Univ of Southern California
Los Angeles, CA 90089, USA
shahram@usc.edu

Shyam Kapadia
Dept of Computer Science
Univ of Southern California
Los Angeles, CA 90089, USA
kapadia@usc.edu

Bhaskar Krishnamachari
Dept of Computer Science
Dept of Electrical Engineering
Univ of Southern California
Los Angeles, CA 90089, USA
bkrishna@usc.edu

ABSTRACT

Zebroids are mobile devices that carry a referenced data item from a server containing that data item to a client that requested it. An environment employs zebroids in order to minimize the availability latency incurred by the client. A device acts as a zebroid when it is in close vicinity of a server and travels along a path that rendezvous with the client. Over time, the storage capacity of the zebroid might become occupied with data items. To carry the requested data item, it must evict one or more of its existing data items. The primary contribution of this study is to quantify the latency-overhead tradeoff associated with alternative replacement policies that might be employed by a zebroid. We employ a simulation study to quantify the tradeoffs associated with policies such as Least Recently Used (LRU), Least Frequently Used (LFU), and a simple random policy. Obtained results highlight the parameter space where the use of zebroids provides superior performance when compared with a static data placement strategy that does not use zebroids. We verify this and the other observed trends using a simple analytical approximation of latency for a 2D random walk mobility model.

1. INTRODUCTION

Advances in technology, both in the area of storage and wireless communications, have now made it feasible to envision on-demand delivery of continuous media 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. The radio range of these so called ‘C2P2’ devices is in the order of a few hundred feet enabling them to collaborate to form a mobile ad-hoc network to deliver the requested data to a client.

Typical components of a C2P2 system provide the following functionalities. First, a discovery component determines those data items available in δ time units and vehicles

Display time	$B_{link} = 10$ Mbps	$B_{link} = 20$ Mbps	$B_{link} = 50$ Mbps
1 min	1.76 secs	0.88 secs	0.35 secs
2 min	3.52 secs	1.76 secs	0.7 secs
5 min	8.8 secs	4.4 secs	1.76 secs

Table 1: Typical times to replicate a data-item, in this case an audio clip with display rate of 300 Kbps, from a server to a zebroid.

containing these data items. We term δ as the availability latency of a data item¹. A data item is available immediately when it resides in the local storage of the C2P2 device serving the request. Second, an interface shows this list of data items and their availability latency to a user, facilitating data item selection. Once a user initiates display of a data item, an admission control component [9] (third component) ensures availability of both resources and the referenced data, to support a display free from disruptions and delays termed hiccups, at the client. Fourth, a data delivery scheduling technique utilizes resources as a function of time to deliver the data item to a displaying C2P2 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. Fifth, an ad-hoc network routing protocol facilitates delivery of data between C2P2 devices. Example protocols are DSR [14], ZRP [13], CEDAR [20] to name a few. Another system component may monitor whether the system is providing a target C2P2 with the desired QoS and make adjustments as necessary.

In a typical scenario, a client of this system is provided a list of available data items. Let δ denote the earliest time after which the client encounters a copy of its requested data item. This latency is a function of the current location of the client, its destination and travel path, the mobility model of the C2P2 equipped cars, the number of replicas constructed for the different data items, and the placement of data item replicas across the C2P2 equipped car. (Without loss of generality and in order to simplify the discussion, we assume the term car refers to a C2P2-equipped vehicle.)

The system may employ a variety of techniques to improve δ . First, it may construct a larger number of replicas for

¹Without loss of generality, we will use the term data item in this paper with the understanding that a data item can be an audio title, video title etc.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

those data items with a higher frequency of access [10]. Second, it might control the placement of these replicas across the cars. This might be done in a static manner when the mobility model of cars is pre-deterministic and the requests are known in-advance, e.g., a broadcast delivery model instead of on-demand. Third, one may employ a dynamic placement strategy that requires a car without a data-item that rendezvous with a client.

An implementation of dynamic placement may assume a two-tiered architecture consisting of a low bandwidth control plane (similar to the cellular infrastructure) and a high bandwidth data plane. The data plane supports rates in the order of tens to hundreds of Megabits per second [2] and represents the ad hoc peer to peer network between the vehicles. The bandwidth of the control plane is in the order of a few hundred Kilobits per second used for exchange of control information between a base station and those vehicles in its radio range. Example control information include the data items requested by the vehicles, the location of vehicles and their destination. A dispatcher located at every base-station² uses this control information as follows. 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 identifies those cars, in the vicinity of a server, that rendezvous with a client sooner than the server. Among these, the dispatcher chooses a single car-server pair that minimizes latency. It then coordinates the transfer of a replica for the requested data item to the car which carries it to the client. This data carrier is termed a ‘zebroid’. To simplify discussion, we consider dispatchers that use a single zebroid. Extensions that use multiple zebroids transitively to improve latency is a future research direction.

More formally, a zebroid is a car, other than either the client or a server, whose path intersects with the client in both space and time. The term rendezvous refers to this spatio-temporal intersection. Like the other cars, zebroids have limited storage. When the local storage of the chosen zebroid is completely occupied, it must replace one or more data items to accommodate the new data item that it then carries to the requesting client. This motivates replacement policies to maximize utilization of system storage. The primary contribution of this paper is to quantify the tradeoff associated with alternative policies.

Time to transfer a data item from a server to a zebroid depends on its size and the available link bandwidth. Table 1 shows times for audio clips using typical wireless link bandwidths. Throughout this paper, we assume that the time to spawn a new replica on the zebroid is small compared to the granularity of the mobility of the cars.

Initially, the data item replicas are distributed across the C2P2s using a static replication scheme. This scheme computes the number of data item replicas as a function of their popularity. It is chosen as the baseline for comparison with the replacement policies. It is static since the number of data item replicas in the system do not change and no replacements are performed. Henceforth, we refer to this scheme as the ‘no-replacements’ policy. The replacements incurred by a replacement policy cause fluctuations in the number of data item replicas in the system. However, the dispatchers with the help of the control plane 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. Hence, even though a zebroid may meet a requesting client earlier than other servers, if its local storage contains data items with only a single copy, then such a zebroid is not chosen to carry a new data item.

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 the no-replacements policy incurs neither overhead.

The primary contribution of this work is a zebroid framework in which carrier-based replacement policies are used to reduce availability latency. Using a random walk mobility model, we quantify the improvements in latency that can be obtained using zebroids as compared to the no-replacements policy. We investigate a variety of environments that differ in the contents that the vehicles store as well as the information about predictability of the cars routes’. In each environment, a variety of replacement policies are deployed. The main lessons are summarized in Table 2.

The rest of this paper is organized as follows. Section 2 gives a brief overview of the related work in the area. Section 3 introduces some terminology used in the paper. Section 4 describes the various environments used in this study and a classification of the different carrier-based replacement policies that are deployed in these environments. Section 5 describes the detailed simulation results obtained with the various environments and policies. Section 6 gives the details of our analytical approximation that captures the behavior of a replacement policy. Finally, section 7 presents brief conclusions and future research directions.

2. RELATED WORK

Many prior studies [6, 17], including [8, 11, 10], have investigated techniques to compute both the number of replicas for a data item and their placement across both stationary and mobile devices. These techniques are either centralized or decentralized. They assume the existence of meta information such as frequency of access to the different data items. None investigate techniques to manage available storage of a mobile zebroid as a carrier of data.

Use of mobile nodes as data carriers has been proposed for both intermittent and sensor networks. Intermittent networks [12, 21, 16, 5, 4, 23] consider those environments where a path between a server a client does not exist at all times. The mobile nodes act as carriers of data to facilitate data delivery. With sensor networks, data MULEs have been proposed to collect data from sensors in a sparse networks, buffer this data, and deliver it to wired access points [19, 15, 3]. However, techniques to manage storage of a data carrier is an open research topic that has not been investigated to date. This constitutes our focus, making this study complementary to prior ones.

A policy for content distribution in mobile infostation networks is presented in [24]. This study assumes each node is interested in all files stored at a static infostation. Moreover, a node may serve as a mobile infostation for another node. An exchange occurs between two nodes if each has a file of interest for its companion. This study shows that as the total file repository increases, near optimum resource utilization can be obtained in such an environment even with

²Dispatchers may communicate via the wired infrastructure between base-stations.

Lessons	
1.	It is better to have more cars with lower storage than fewer ones with more storage. (Figure 1)
2.	A simple random eviction policy shows competitive performance. (Figure 1)
3.	For a given storage per car, there exists a critical car density which offers the maximum gains in availability latency. Similarly for a given car density, there exists a critical value for the storage per car where the improvement in latency peaks. (Figures 2, 3)
4.	Even when mobility patterns are less predictable, zebroids continue to improve availability latency. (Figure 4)
5.	The enhancements in availability latency are obtained at the expense of a higher replacement overhead. (Figure 5, 6)

Table 2: Summary of lessons for the replacement policies from the simulation study.

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.
δ_i	Average availability latency of data item i
δ_{agg}	Aggregate availability latency
C2P2 System Parameters	
G	Number of cells in the map (2D-torus).
N	Number of C2P2 devices in the system.
α	Storage capacity per C2P2.
γ	Trip duration of the client C2P2.
S_T	Total storage capacity of the C2P2 system

Table 3: Terms and their definitions

this incentive based non-cooperative policy. In this study, we focus on availability latency and show a swap-based policy is inferior to alternatives such as LRU and LFU.

3. TERMINOLOGY

Table 3 summarizes the notation of the parameters used in the paper. Below, we describe briefly the main quantities. Assume a network of N C2P2-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 C2P2 under consideration be γ .

We now define the normalized frequency of access to the data item i , denoted R_i , is: $R_i = \frac{(f_i)^n}{\sum_{j=1}^T (f_j)^n}$; $0 \leq n \leq \infty$

The exponent n characterizes a particular replication technique. $n = 0.5$ denotes a square-root replication technique. R_i is normalized to a value between 0 and 1. The number of replicas for data item i , denoted as r_i , is: $r_i = \min(N, \max(1, \lfloor \frac{R_i \cdot N \cdot \alpha}{S_i} \rfloor))$

The availability latency for a data item i , denoted as δ_i , is defined as the time after which a client C2P2 will find at least one replica of the data item accessible to it, either directly or via multiple hops. 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 δ_i for every data item i with its f_i to obtain the following weighted availability latency (δ_{agg}) metric: $\delta_{agg} = \sum_{i=1}^T \delta_i \cdot f_i$

4. ENVIRONMENTS AND CARRIER-BASED REPLACEMENT POLICIES

4.1 Environments

In the following, we introduce various environments considered in this study. Our environments assume two types of vehicles: cars and buses. Cars have a limited storage carrying a fraction of the data item repository. Buses have sufficient storage to carry the entire repository (see environment 3). Each vehicle is equipped with a single C2P2 device. The dispatchers at the base stations, with the help of the control plane, ensure that at least one copy of every data item is maintained in the ad hoc network at all times. In other words, none of the environments lose data. Table 4 summarizes the properties exhibited by the various environments.

Environment 1 assumes vehicles are cars. The dispatchers know complete routes of all cars at all times.

Environment 2 also assumes vehicles are cars (no buses). It differs from environment1, in that, the routes of all cars are not known at all times. Instead, the car routes are governed by a certain route prediction accuracy parameter. The prediction accuracy parameter inherently provides a certain probabilistic guarantee on the confidence of the car route predictions. For example, a 70% value for this parameter indicates that the route 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.

Environment 3 assumes presence of both buses and cars. Car and bus movements are governed by the mobility model. Buses can be considered as ‘universal’ servers since they can directly serve any data item request at any time for any client as long as the client is in their vicinity.

4.2 Carrier-based Replacement policies

The replacement policies considered in this paper are reactive since a replacement only occurs in response to a request issued for a certain data item. Recall that the latency for a data item request is defined as the time between when the request was issued at a client to when the client encounters a car containing a replica of the requested data item for the first time. With the help of the control plane, the dispatchers are able to determine if there is a zebroid that will encounter the client earlier than any of the other potential servers. If there are many such zebroids, the one that meets the client earliest is chosen. When multiple zebroids meet the client at the same time, one is chosen randomly. Next,

Characteristic	Environment 1	Environment 2	Environment 3
Requestors/Clients	Cars	Cars	Cars
Servers	Cars	Cars	Cars and Buses
Zebroids	Cars	Cars	Cars
Mobility	Exact routes known for all cars	Routes of cars known with a certain probability	Exact routes known for all cars and buses
Who carries what?	Cars carry a few data items as per their storage capacity	Cars carry a few data items as per their storage capacity	Buses carry the entire data item repository while cars carry only a few data items depending on their storage

Table 4: Summary of characteristic properties exhibited by the various environments.

the dispatcher replicates the requested title on that zebroid. However, the zebroid’s local storage may be completely exhausted. Hence, it might need to evict another data item’s replica in order to accommodate this new one. This problem is analogous to that encountered in case of operating system paging where the goal is to maximize the cache hit ratio to prevent the disk access delay [22]. We present below a list of carrier-based replacement policies adapted from the different page replacement ones.

1. **Least recently used (LRU)** LRU-K [18] maintains a sliding window containing the time stamps of the K^{th} most recent references to pages within the buffer. During eviction, the page whose K^{th} most recent reference is furthest in the past is evicted. Here we consider the case with $K = 1$. (a) **Local (lru-local)**: Each C2P2 keeps track of which data item within its local repository was least recently accessed. During replacement at the chosen zebroid, this is the data item whose replica is evicted. (b) **Global (lru-global)**: The dispatcher maintains the identity of the least recently requested data item computed across all client requests. The dispatcher maintains the list of data items resident on each C2P2. This metadata is in the order of Kilobytes. A zebroid contacts the dispatcher for a victim and the dispatcher chooses the data item that it should evict. This replacement policy is centralized and requires (1) a client to transmit its data item requests to the dispatcher, and (2) for the dispatcher to maintain sufficient metadata to choose a victim for each zebroid.
2. **Least frequently used (LFU)** (a) **Local (lfu-local)**: Each C2P2 keeps track of the least frequently used data item within its local repository. During eviction³, 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 client requests. When a zebroid contacts the dispatcher for a victim data item, the dispatcher chooses the data item with the lowest frequency (global) 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.
4. **Swap (exchange) policies**: With this class of replacement policies, every time a zebroid is chosen to

³The terms eviction and replacement are used interchangeably.

transport the requested data item to the client two replacements are performed, one at the zebroid and the other at the server. The server transfers the requested data item’s replica to the zebroid which in exchange transfers the data item replica it evicted to the server. In this way, the total number of replicas for every data item in the system is the same. Note that the eviction policy used at the zebroid can be LRU, LFU or random.

5. SIMULATION STUDY

5.1 Simulation Set-up

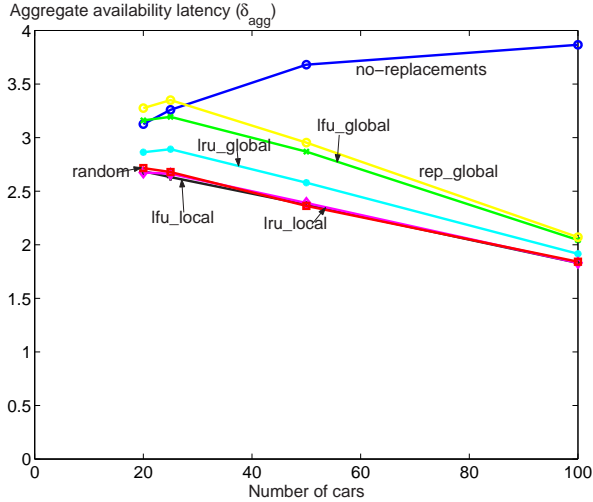
We first list the assumptions of the simulation study and then describe the parameter settings used in our experiments.

- The map used is an $n \times n$ 2D torus.
- A Markovian mobility model representing a 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.
- Only C2P2 equipped cars within the same cell may communicate with each other.
- A car can transition from a cell to any of its neighboring 8 cells.
- In each time slot, every car moves from one cell to another depending on its current position and probability transition matrix $Q = [q_{ij}]$ where q_{ij} is the probability of transition from state i to state j .
- A homogeneous repository of equi-sized data items is used ($S_i = S$).
- The parameters γ , δ have been discretized and expressed in terms of the number of time slots.
- A C2P2 device does not maintain more than one replica of a data item. This is because additional replicas occupy storage without providing benefits.

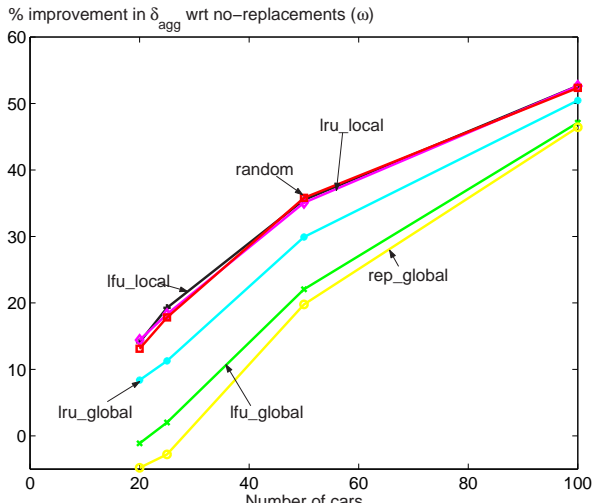
Here, we set S_i for every data item to be 1. α represents the number of storage slots per C2P2. Each storage slot stores one data item. γ represents the duration of the client’s journey in terms of the number of time slots. Hence the possible values of availability latency are between 0 and γ . δ is defined as the number of time slots after which a client

C2P2 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.

Throughout this section, 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 with corresponding scaling in the other experimental parameters. We simulated



1.a



1.b

Figure 1: Figure 1.a and Figure 1.b show δ_{agg} and ω for various replacement policies as a function of (N, α) values when the total storage in the system is kept fixed, $S_T = 200$. Here $T = 50$ and the map is a 5×5 Torus.

a skewed distribution of access to the T data items using a Zipf distribution with a mean of 0.27. This distribution is shown to correspond to sale of movie theater tickets in the United States [7]. Initially, all cars are distributed across the map as per the steady-state distribution governed by Q . Also, the replicas for data items are calculated as per the

square-root replication technique. The data item replicas are distributed uniformly across the C2P2 devices. At every time slot a new request is issued. The client that issues the requests is chosen in a round-robin manner. After a maximum period of γ , the latency encountered by this request is recorded.

The 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 metrics considered comprise δ_{agg} , percentage improvement in δ_{agg} with a replacement policy as compared to the no-replacements case and overhead of a replacement policy. This later metric is quantified using number of replacements performed by zebroids.

5.2 Results

The lessons we obtain through an extensive simulation study have been summarized in table 2. In the following, we describe the results to support each lesson.

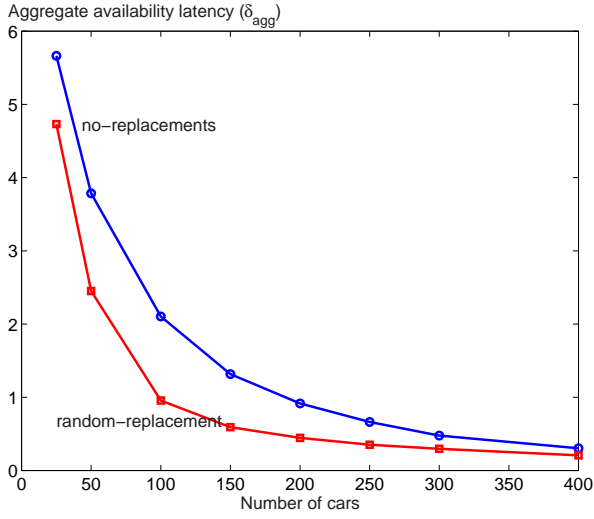
Lesson 1: First, we present the scale-up experiments with environment 1 to indicate how having more cars with low storage as opposed to fewer ones with high storage is more beneficial to the replacement policies in achieving lower latency. In these experiments, we change α and N proportionally to keep the total system storage, S_T , as well as the database size constant (T is held constant). The results for the scale-up experiment are shown in Figure 1. Here, we set $T = 50$ and $S_T = 200$. Then we choose the following values of $(N, \alpha) = \{(20, 10), (25, 8), (50, 4), (100, 2)\}$. As α decreases and N increases, all the policies show a rapid improvement in latency as compared to the no-replacements case. This is because additional cars (N) increase the number of zebroids. With this increase in the zebroid density, the dispatcher is almost always able to find a zebroid that can deliver the requested data item to the client earlier than any of the potential servers.

Somewhat surprisingly, δ_{agg} for the no-replacements policy goes up as N is increased, while the δ_{agg} curves for the replacement policies show a downward trend. There are two reasons for this. First, both the total storage capacity and the database size are fixed. Second, as N increases, the sample space of the potential clients that issue requests increases. For example, if 10000 requests were generated, with $N = 20$, each C2P2 initiated 500 data item requests. With $N = 100$, each C2P2 initiated only 100 requests. Moreover, α per C2P2 device reduces from 10 when $N = 20$ to 2 when $N = 100$. Note that the no-replacements policy assigns the same number of data item replicas for all (N, α) values since S_T remains fixed.

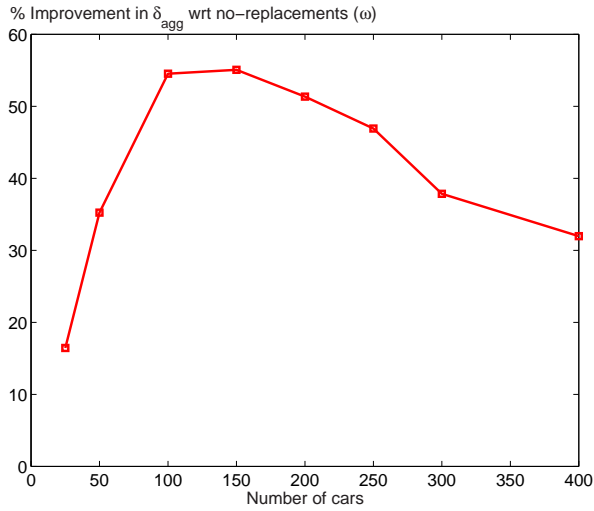
Lesson 2: The results in figure 1 indicate that the local policies lru-local, lfu-local and random show similar performance. They do better than the global policies in terms of latency. A replacement policy in which the zebroids evict data items randomly performs very well. This policy is simple, decentralized and easy to implement. It makes an eviction decision that is blind to the popularity of the data items and the previous history of the data item requests. Hence, for the remainder of the paper, the random replacement policy is used as the representative of the local policies.

Lesson 3: In order to separate out the variation of N and α , we next present results capturing the variation of N

while keeping α constant (case 1) and vice-versa (case 2). Here we set $T = 25$. Figure 2 shows the variation in δ_{agg} when α is fixed to 2 and N is increased from 25 to 400 (environment 1). There are two main observations from these



2.a



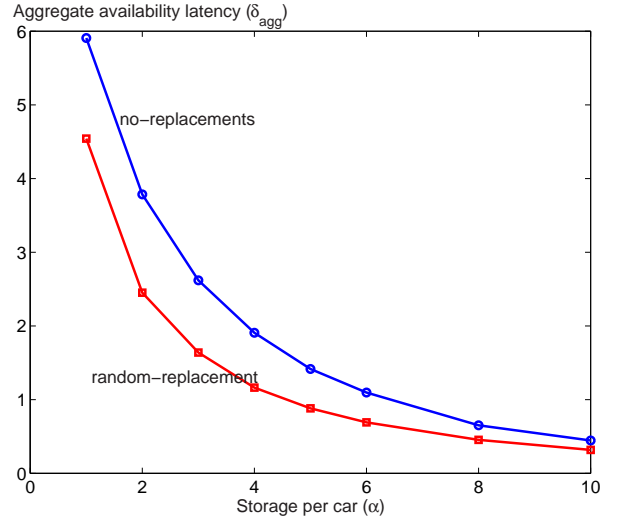
2.b

Figure 2: Figure 2.a and Figure 2.b show δ_{agg} and ω as a function of the car density when α is fixed at 2. Here $T = 25$ and the map is a 5x5 Torus.

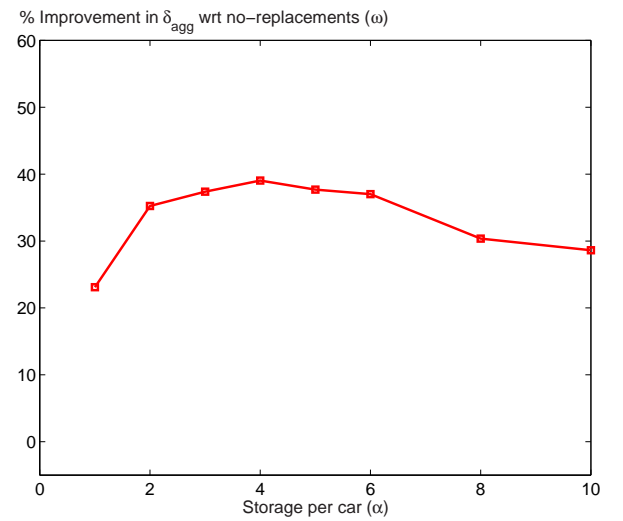
figures in support of lesson 3 which indicates the existence of a peak that marks the highest improvement in latency that can be obtained with the replacement policies. First, keeping α constant, increase in car density has higher benefits because increasing N introduces more zebroids in the system as observed with lesson 1. Second, the curves that indicate the % improvement in δ_{agg} as compared to the no-replacements case (ω) show an inverted U-shape. The reason that ω reduces is because increasing N also increases the total storage in the system. Hence, the number of replicas per data item goes up thereby increasing the number of servers. Consequently, the number of zebroids decreases. Hence, the replacement policy cannot find a zebroid as often to trans-

port 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-replacements case in bringing δ_{agg} down. The net effect results in reduction in ω for larger values of N .

Figure 3 depicts the variation in δ_{agg} when N is fixed to 50 while α is increased from 1 to 10 in steps of 1 (environment 1). This figure further supports lesson 3 in that it indicates that given a data item repository, for a constant car density if we keep increasing the storage per C2P2 device the enhancements in latency with the replacement policies first increase and then go down. This is because the number of zebroids shows a similar trend as α is increased.



3.a



3.b

Figure 3: Figure 3.a and Figure 3.b show δ_{agg} and ω as a function of α when N is fixed at 50. Here $T = 25$ and the map is a 5x5 Torus.

Lesson 4: The next set of results elaborate on lesson 4 which applies for environment 2. Here, we present a representative scenario that shows that with reduced predictability replacement policies still show improvements in δ_{agg} . A

probabilistic error is associated with the prediction of the future path of the C2P2 equipped cars. We use a single metric to quantify the accuracy of these predictions. Figure 4 shows the variation in δ_{agg} as a function of this route prediction accuracy metric. We observe a smooth reduction

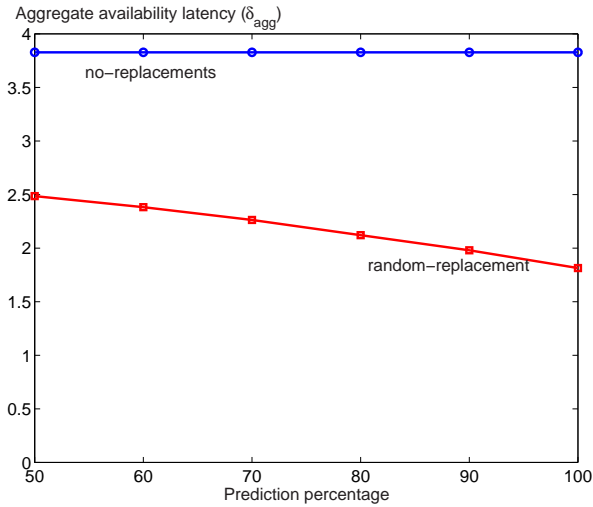


Figure 4: Figure 4 shows δ_{agg} as a function of the prediction accuracy metric with a 5x5 torus. Here $N = 100$, $\alpha = 2$ and $T = 50$.

in the improvement in δ_{agg} as the prediction accuracy metric reduces. Note that here as in the other cases only a maximum of one zebroid per client request may be used by the replacement policy even though the car routes prediction information is not perfect. In order to provide higher gains in latency a policy may choose to employ multiple zebroids as data carriers in order to adapt to the lower accuracy in the route predictions. We intend to consider such variants as part of our future work.

Lesson 5: The following set of results elaborate on lesson 5 which indicates that the improvement in latencies are obtained at the cost of a replacement overhead. Figure 5 shows the number of replacements for the various policies as a function of the different $(N:\alpha)$ values. As N increases and α reduces, the replacement policies perform a larger number of replacements thereby enabling the latency to be reduced. Note that a policy only performs a replacement when it leads to an improvement in the latency. Hence, higher replacements lead to higher improvements in δ_{agg} .

When the storage is so scarce that only one replica per data item exists in the C2P2 network, the number of replacements will be zero since any replacement will cause a data item to be lost from the database. The dispatchers along with the control plane prevent any data loss. At the other end of the spectrum, when the storage is so abundant that the entire database can be replicated on every car then number of replacements is again zero since each request can be satisfied locally. However, there is a storage spectrum in the middle where performing replacements results in improvements in δ_{agg} . Figure 6 shows the number of replacements for the two cases 1 and 2 mentioned above. These figures indicate an inverted U-shape. This behavior is similar to that seen with ω . This is because

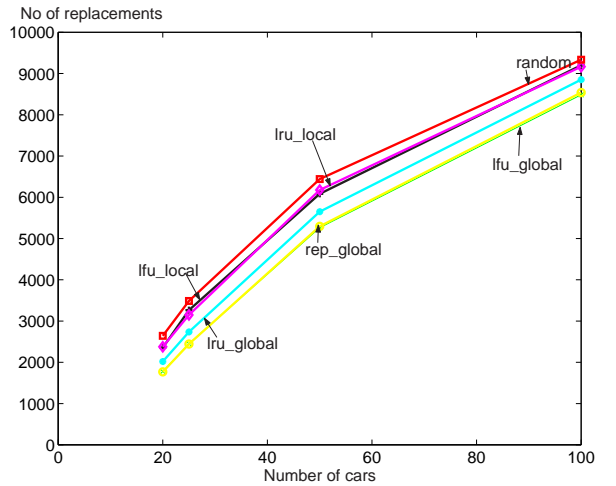


Figure 5: Figure 5 shows the number of replacements incurred by the various replacement policies as a function of (N,α) values when the total storage in the system is kept fixed, $S_T = 200$. Here $T = 50$ and the map is a 5x5 Torus.

initially as storage is increased the benefits obtained with the replacements policies increase as more replacements can be performed thereby reducing latency. Eventually a peak is hit where suitable zebroids that can carry the requested data item to the client become available more readily resulting in a higher replacements giving a higher ω . Thereafter, as the storage is increased further more data item replicas are allocated, hence number of zebroids decreases resulting in a lower number of replacements.

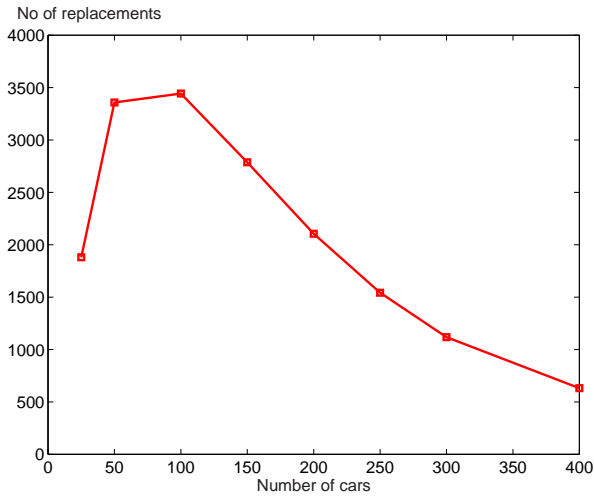
With environment 3, recall that in addition to the cars, we have an additional type of vehicle, ‘buses’ which carry the entire data item repository. Since the buses always have a copy of every data item they cannot be zebroids. Except for the additional presence of buses the simulation setup for the environment3 is similar to that of the other environments. Here, we study the variation in the system performance, namely, δ_{agg} and replacement overhead as we increase the number of buses in the system.

Figure 7.a and 7.b show the variation in δ_{agg} as a function of the number of buses for a 5x5 torus when $N = 50$. The performance is compared against the no-replacements case. Increase in number of buses reduces availability latency, trends seen are similar to that seen with lesson 3.

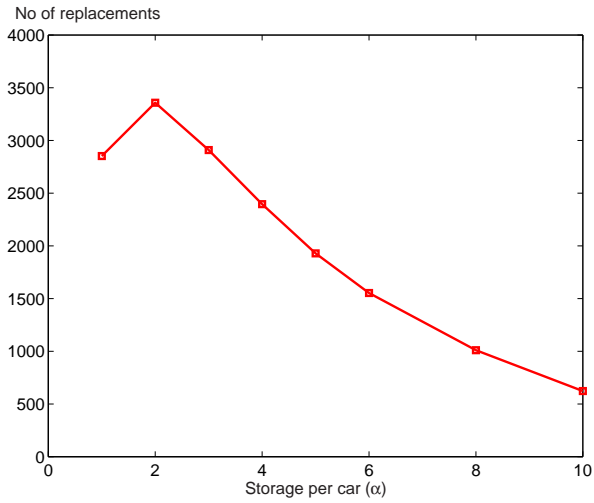
Figure 7.c shows the trend in the number of replacements as a function of the number of buses.

These results support lesson 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 goes up but 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 number of buses goes up, since the number of servers increases.

The results for the various swap policies have not been presented here since they provide similar improvements in



6.a) $\alpha = 2$

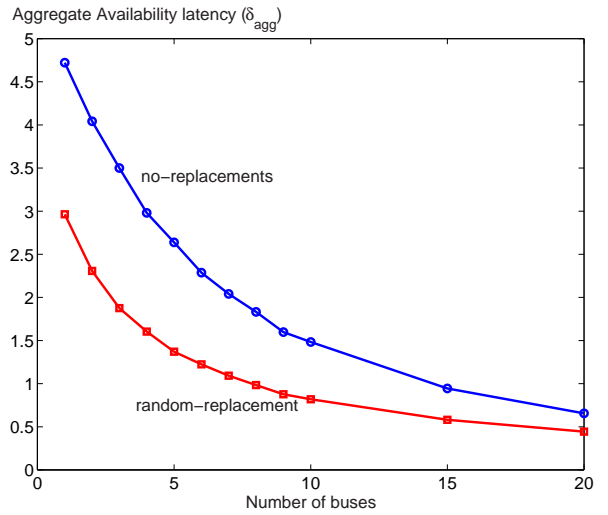


6.b) $N = 50$

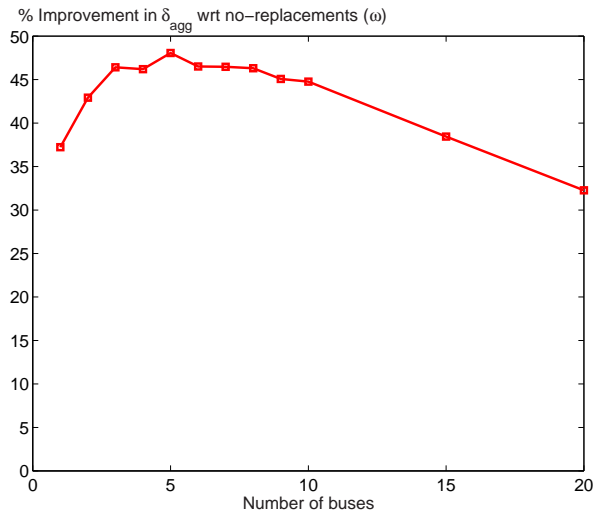
Figure 6: Figure 6.a and Figure 6.b show the number of replacements as a function of N keeping α and vice-versa respectively. Here $T = 25$ and the map is a 5×5 Torus.

latency but the number of replacements incurred is twice that of the non-swap policies. This is because, as mentioned earlier with the swap policies, whenever there is an eviction two replacements are incurred by the swap.

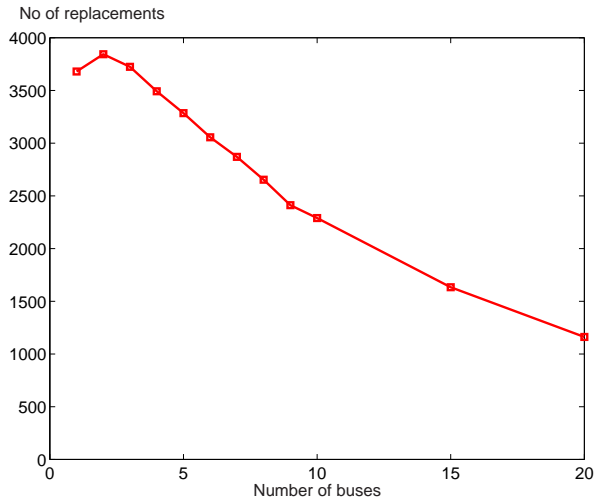
Finally, if we remove the constraint that at least one copy of every data item must be present in the system at all times then we get the following main observation. When storage is scarce, the global policies (lru-global, lfu-global) produce the lowest latency and lowest replacement overhead but at the cost of a higher loss of data. The local policies (lfu-local, lru-local) produce the highest latency and incur higher replacement overhead but lose less data. The random policy's performance is between the local and the global ones. When system storage is above a certain threshold, none of the policies lose any data. Due to lack of adequate space we have not shown these results here.



7.a



7.b



7.c

Figure 7: Figure 7.a and Figure 7.b show δ_{agg} and ω as a function of the number of buses with framework3 with a 5×5 torus for the random replacement policy. Figure 7.c shows the overhead of this policy in terms of the number of replacements as a function of the number of buses with $N = 50$. Here $\alpha = 2$ and $T = 50$.

6. ANALYTICAL EVALUATION

In this section, we provide a simple probabilistic analysis that gives an upper bound on the latency improvements obtained using carrier-based replacements. We consider the movement of N cars as an unbiased random walk on a 2D torus consisting of G total cells. At the moment that a data item i is requested by a single client, we assume that there are r_i servers with replicas of that data item. Let \overline{N}_i^c be the expected total number of nodes that are in the same cell as the servers, which is given by the following expression:

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

Consider first the scenario where no replacements are made. 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 one of random walks by any of the servers. Aldous *et al.* [1] show that the meeting time of two random walks in such a setting can be modelled as an exponential distribution with the mean $cG \log G$, where the constant $c \simeq 0.34$ 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 (we should note that this formulation is valid only for sparse cases when $G \gg r_i$):

$$\overline{\delta}_i = \frac{cG \log G}{r_i}$$

The aggregate availability latency of the no-replacement case is then this expression averaged over all data items, weighted by their frequency of access:

$$\overline{\delta_{agg}(no-repl)} = \sum_{i=1}^T \frac{f_i cG \log G}{r_i} \quad (2)$$

Now consider when replacements are made. In the analytical model, we assume that that N_i^c new replicas are created through replacements, so that the total number of replicas is increased to $r_i + N_i^c$. The available latency is reduced since the client is more likely to meet a replica earlier. The aggregated expected availability latency over all data items, with replacements, is then:

$$\overline{\delta_{agg}(repl)} = \sum_{i=1}^T \frac{f_i cG \log G}{r_i + \overline{N}_i^c} \quad (3)$$

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 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 meeting time between the client.

It should be noted that the performance of the scheme with replacements is similar to an oracle-based algorithm that is aware of all future movement of cars in terms of latency. The oracle-based algorithm 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. In the replacements scheme this selected zebroid is included in the N_i^c new replicas.

To get the fractional difference (labelled ω) in the latency between the no-replacements and replacements case we take the difference between equations 2 and 3 and divide it by the former. This captures the fractional improvement in the availability latency obtained by performing replacements.

$$\omega = \frac{\sum_{i=1}^T \frac{f_i \cdot C}{r_i} - \sum_{i=1}^T \frac{f_i \cdot C}{r_i + (N - r_i) \cdot \left(1 - \left(1 - \frac{1}{G}\right)^{r_i}\right)}}{\sum_{i=1}^T \frac{f_i \cdot C}{r_i}} \quad (4)$$

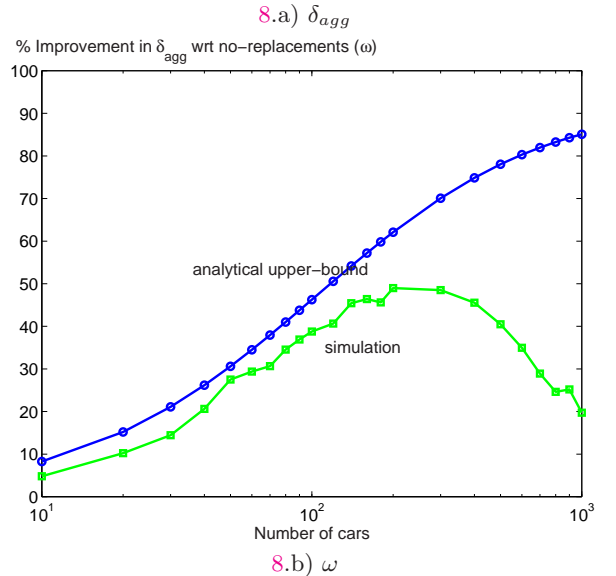
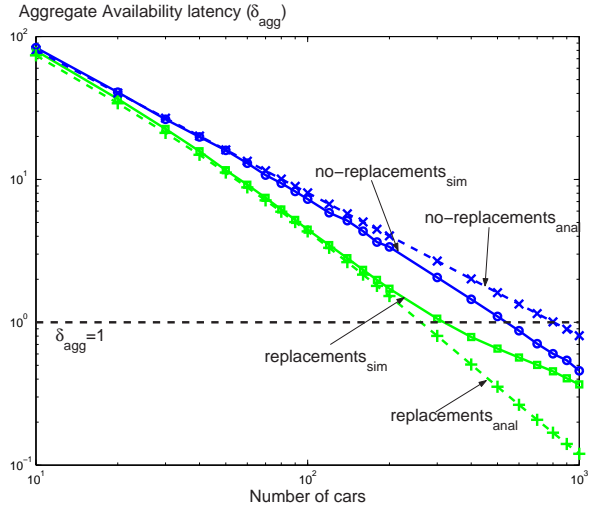


Figure 8: Figure 8.a shows δ_{agg} obtained through simulations as a function of car density with the no-replacements and replacement policies respectively for a 10x10 Torus. Figure 8.b shows the % improvement in δ_{agg} obtained wrt the no-replacements policy via simulations as well as the analytical approximation for a 10x10 Torus. Here $T = 10$.

Figure 8.a shows the variation in δ_{agg} as a function of N for $T = 10$ with a 10x10 torus. Both the x-axis and y-axis are drawn to a log-scale. Figure 8.b show the % improvement in δ_{agg} obtained with dynamic replication for the 10x10

torus respectively. In this case, only the x-axis is drawn to a log-scale. The analysis and simulation curves match quite well until the point where δ_{agg} starts becoming less than 1. Also, initially, when the network is sparse the analytical approximation given by Equation 4 is close to the simulation results. However, as N increases the two curves rapidly diverge since then N becomes greater than G and the network is no longer sparse. Moreover, as mentioned earlier, the analysis provides an upper bound on the latency improvements, as it treats the carriers given by $\overline{N^c}$ independently while the simulations do not have that constraint.

7. CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

This study introduces a zebroid framework in which zebroids are used as data carriers to transport data items to clients thereby reducing their observed latency. Various replacement policies are considered to determine which candidate data item replicas to evict from the zebroids during the replacement process. We have quantified the performance improvement of these policies using availability latency as the metric of interest. The base line for comparison is a static no-replacements scheme. Moreover, the overhead incurred by a replacement policy is captured in the number of replacements incurred by it. The primary conclusion is that for a given database repository using more cars each with lower storage provides higher gains in latency as compared to using a few with more storage. We present an analytical formulation of the improvements in latency with a replacement policy in a sparse network.

We intend to extend this study in several directions. First, we intend to explore alternate definitions for the route prediction accuracy parameter (see Section 4.1, environment 3) as part of our future work. Second, instead of using only one zebroid per client request, multiple zebroids can be employed thereby making it possible to improve the latency even further. However, this will lead to a higher replacement overhead. We intend to study such replacement schemes that employ multiple zebroids. Third, to better reflect reality we would like to validate the lessons from this study with some real world simulation traces of vehicular movements.

8. REFERENCES

- [1] D. Aldous and J. Fill. Reversible markov chains and random walks on graphs. Under preparation.
- [2] S. Bararia, S. Ghandeharizadeh, and S. Kapadia. Evaluation of 802.11a for streaming data in ad-hoc networks. In *ASWN Boston, MA*, 2004.
- [3] A. Beafour, M. Leopold, and P. Bonnet. Smart tag based data dissemination. In *ACM WSNA*, Oct 2002.
- [4] I. Chatzigiannakis, S. Nikolettseas, and P. Spirakis. An efficient communication strategy for ad-hoc mobile networks. In *PODC*, pages 320–322, New York, NY, USA, 2001. ACM Press.
- [5] Z. Chen, H. Kung, and D. Vlah. Ad hoc relay wireless networks over moving vehicles on highways. In *ACM MobiHoc*, pages 247–250, New York, 2001.
- [6] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. In *SIGCOMM*, pages 177–190. ACM Press, 2002.
- [7] A. Dan, D. Dias, R. Mukherjee, D. Sitaram, and R. Tewari. Buffering and Caching in Large-Scale Video Servers. In *Proceedings of COMPCON*, 1995.
- [8] 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.
- [9] 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.
- [10] 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.
- [11] S. Ghandeharizadeh, B. Krishnamachari, and S. Song. Placement of Continuous Media in Wireless Peer-to-Peer Networks. *IEEE Transactions on Multimedia*, April 2004.
- [12] M. Grossglauser and D. Tse. Mobility increases the capacity of ad hoc wireless networks. *IEEE/ACM Trans. Netw.*, 10(4):477–486, 2002.
- [13] Z. Haas. A new routing protocol for the reconfigurable wireless networks. In *In IEEE Int. Conf. on Universal Personal Communications*, 1997.
- [14] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.
- [15] P. Juang, H. Oki, and Y. Wang. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebnet. In *ASPLOS*, Oct 2002.
- [16] Q. Li and D. Rus. Sending messages to mobile users in disconnected ad-hoc wireless networks. In *ACM MobiCom*, pages 44–55, New York, NY, USA, 2000.
- [17] 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.
- [18] 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.
- [19] 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.
- [20] P. Sinha, R. Sivakumar, and V. Bharghavan. CEDAR: a core-extraction distributed ad hoc routing algorithm. In *INFOCOM (1)*, pages 202–209, 1999.
- [21] T. Spyropoulos, K. Psounis, and C. Raghavendra. Single-Copy Routing in Intermittently Connected Mobile Networks. In *SECON*, April 2004.
- [22] A. Tanenbaum. *Modern Operating Systems, 2nd Edition, Chapter 4, Section 4.4*. Prentice Hall, 2001.
- [23] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. In *Technical Report CS-200006, Duke University*, April 2000.
- [24] W. Yuen, R. Yates, and S. Mau. Noncooperative content distribution in mobile infostation networks. In *IEEE WCNC*, 2003.