Data Management in Mobile P2P Systems

Masataka Kan
Visiting Scholar(Hitachi,Ltd.)
Computer Science Department
Stanford University

September 6, 2005

Abstract

We focus on effective data sharing in mobile P2P system. In order to realize data sharing, data have to be stored in the system so it can be retrieved by every peer. Our data sharing mechanism consists of a data storing method, a query processing method, and a data dissemination method. We adopt a limited search approach for the query processing in order to retrieve data effectively in highly mobile environment. Therefore, the goal of the data storing method is to distribute data in wide area. We adopt data hashing approach for the data storing method and propose data distribution algorithm HID and redundancy control algorithm ARw/D. Our results show that HID with ARw/D performs significantly well.

1 Introduction

As wireless communication becomes more useful, the needs of managing data in mobile environment are increasing. For example, data sharing in emergency situation [2] and data sharing among automobiles [3] are suggested.

In such a situation, a mobile Peer-to-Peer(P2P) approach, where peers communicate directly with other peers and data is stored in peer's local storage, may be cost effective and easy to apply in the real world because it does not need any fixed infrastructure, for example, a central server or communication stations.

Our paper focuses on how to effectively sharing data in mobile P2P system. In order to realize data sharing, data have to be stored in the system so it can be retrieved by every peer. Thus, our goal in this study is to realize no loss of data and easiness of retrieving data.

Our data sharing mechanism consists of the data storing method, the query processing method, and the data dissemination method. For the data storing method, We present the data distribution algorithm, redundancy control algorithm, and storage management algorithm and evaluate them. To keep the anonymity of peers, these algorithms are executed in a distributed manner.

For the query processing method, we present two approaches, a limited search and a wide search. Most of the mobile ad-hoc network studies [4] [5] adopted the wide search approach. However, in highly mobile environment, the communication cost for query processing is very high because peer's location is unstable. Thus, we select the limited search. In order to process the limited search efficiently, we use the data storing method which distributes data in wide area.

For the data dissemination method, we utilize the SR algorithm which is presented in our former work [1].

2 Strategy

The goal of this study is to propose a data management mechanism for peers to share data in a mobile P2P system. Data sharing will be accomplished by a distributed shared storage that can be accessed by all peers.

We make the following assumptions:

- The size of the area where peers share data is limited. Data should be shared in a particular area, which we call the data sharing area.

- The composition of peers in the area may change over time. In particular each peer may exit from the data sharing area.
- The density of peers in the data sharing area is constant. When a peer exits from the data sharing area, another peer enters the data sharing area.
- Data to be shared are generated in the the sharing area by the peers.
- There is no infrastructure which controls communications among the peers. Each peer communicate autonomically.

We also make the following assumptions:

- Each peer moves randomly and may exit from the data sharing area.
- Each peer knows local information (such as its own location, the number of receptions of each data,...).
- Each peer does not know global information (such as other peer's locations, what data other peer holds,...).
- Each peer has local storage.
- Each peer communicate wirelessly.
- Each peer has a unique id.

In this environment, a storage table has the following properties:

- There is only one table in the data sharing area.
- Each tuple of the table consists of a unique key and a value.
- The table is treated in append-only manner (no deletion).

To realize data sharing, data has to be stored in the system so it can be retrieved by every peer. So there are two data processing phases. The first phase is a data storing one. When data is generated, a data source, i.e., a peer which generates that data, disseminates the data. And Some of the peers in the P2P system receive and store the data in their local storage. The second phase is a data retrieval one. When a peer wants to retrieve data, the peer sends a query to the peers. A peer which holds the data of interest sends back data as an answer to the query. Thus, our mechanism consists of the following methods:

- A method for storing data in P2P systems;
- A method for processing queries on data (sending and answering queries).

We also need a data dissemination method for data storage and query processing.

2.1 Data Storage

The goal of the storage system is to store the data so that all data are kept in the data sharing area and so that every peer can retrieve data effectively in the data retrieval phase.

Because there is no central manager who controls peers, the system has to mange storage in a distributed manner. Thus, each peer has to follow a same protocol to mange data. What each peer can do is to store data and to retransmit data. When a peer receives data, following the protocol it decides whether it stores the data and whether it retransmits the data to neighbors. When a storage size of each peer is limited, each peer also has to decide what data to keep and what data to drop. As a result of processing by each peer, data are replicated and distributed in the system.

2.2 Query Processing

To retrieve data, peers in the system have to process queries. A querier, a peer which wants to retrieve data, sends a query and a data holder, a peer which holds the data, sends back the answer for the query.

There are two ways to handle these queries. One way is to perform a wide search in which a querier sends the query over a wide range. The other way is to perform a limited search, in which a querier sends a query to peers within a limited range.

In a wide search, a querier floods the query, to ensure that every possible data holder gets the query. And a data holder also floods an answer to all peers in the system because the location of the querier is uncertain when the querier is moving randomly. In this way, a querier can get the answer if there is at least one peer tha holds the data of interest. However, the communication cost for query processing is very high because of the wide area flooding.

In a limited search, a querier floods a query to peers in a limited area. Similarly, a data holder also floods a query in a limited area. In this way, it is much easier for the data holder to send the answer to the querier because the impact of a peer's mobility is much less when the distance between them is short. So communication cost for query processing is lower. However, each data has to be stored at many peers to increase the likelihood that a querier find the data it needs nearby.

As shown above, there is a tradeoff between communication cost in the data retrieval phase and storage cost in the data storing phase. In this paper, we consider the case where queries are more significant than data storage operations. So we select the limited search as the way to handle queries. And moreover, we adopt a one-hop area as the limited area in order to reduce the influence of peer's mobility. Thus, we propose a data storing method which replicates and distributes data effectively and efficiently.

2.3 Data Dissemination

We adopt flooding-based communication to disseminate data and queries. In a high-mobility environment, flooding-based communication can be used to disseminate messages effectively because it does not use any routes. When peers are highly mobile, maintaining a route is very difficult.

We present flooding-based dissemination algorithm which we studied for event dissemination in high-mobility ad-hoc networks [1]. The algorithm uses three functions: a suppression function, a delay function, and a broadcast function. When a peer receives a message, it decides if the message is discarded using the suppression function. If the message was not discarded, the peer waits for a period calculated using a delay function before transmitting the message. It is likely that the peer will encounter peers which have not received the message by moving for some period of time. After waiting, it transmit the message using a broadcast function.

For the suppression function there are two options:

- TTL(Time To Live): Each message has a TTL. The TTL is set at an initial TTL value when the message is created. A peer decrements the TTL by 1 when it receives a message. If the TTL reaches 0, the message is discarded.
- SR Count: Each peer locally has a SR Count for each message. The initial SR Count value is 0. When a peer receives a message, it increment the SR Count by 1. Once the SR Count of a message reached some threshold value, the peer discards the received message.

For the delay function, we adopt FIXED(n,m). FIXED(n,m) makes a peer wait for a period selected uniformly from the interval [n,m] time units, starting from the time the message is received. The function uses the interval [n,m] to avoid a collision during transmission.

The broadcast function implements two optimizations:

- If a collision occurs, the message can be re-transmitted soon after.
- If we can tell no peers are within range, we can postpone the transmission briefly until peers appear.

2.4 Data Sharing Mechanism

Each peer has data sharing mechanism which consists of the data storing method, the query processing method, and the data dissemination method. Each peer stores data using the data storing method in the data sharing phase. And each peer retrieves data using query processing method in the data retrieval phase. In both phases, a peer use the data dissemination method to send data or queries.

The objective of the data sharing mechanism is to make each peer able to access as many data as possible efficiently. Each peer can retrieve data from its one-hop neighbors. Therefore, to achieve this objective, the data sharing method has to distribute as many data as possible within a one-hop area of each peer with as little overhead as possible.

3 Algorithm for Data Management

In this section, we present our data management algorithm. In the data storage phase, each peer stores data according to a data storage function. Similarly, a peer decides what data that correspond to it to keep and what data to drop using a storage management function. The storage management function is called by the data storage function when there is no room in the storage area to store received data. In the data retrieval phase, each peer processes queries using a query processing function.

3.1 Data Storage Algorithm

The data storage function has two sub-functions, a distribution function and a redundancy control function. The distribution function decides what data to store in order to distribute data. The redundancy control function is an extension to the data storing. When a peer has extra room in its storage, the function decides whether the peer stores data which it does not have to store. This function makes data well-distributed using peer's storage effectively.

3.1.1 Data Distribution

For the distribution function we consider two options:

- Hash on ID(HID): Each peer decides whether it stores a data item depending on the item's key. Peers are classified into groups depending on their id and data are hashed into the groups depending on the key. So, the number of group corresponds to the number of buckets in the hashing. A peer stores the data which hashes into its group. That is, each peer stores a data item if the following condition is satisfied:

  $$Key \mod Number\ of\ Groups = PeerID \mod Number\ of\ Groups$$

- Storing in Every K-Hop(K-Hop): Each peer decides whether it stores a data item depending on the hop count from the data source that is adding the item to the table. Thus, each item is stored every K hops, where K is fixed. That is, each peer store a data if the following condition is satisfied:

  $$Hop\ Count\ from\ The\ Data\ Source \mod K = 0$$

3.1.2 Redundancy Control

Combined with the distribution function, the redundancy control function decides what data to actually store at a peer. This function groups data by the distribution function. Then it prioritizes the group of data and decides what data to store depending on the group to which the data belongs.

The method of prioritization varies with the distribution function. When HID is used as the distribution function, the priority of each data group is set using the following formula:

$$P_0 = Key \mod Number\ of\ Groups - PeerID \mod Number\ of\ Groups$$

$$Priority = \begin{cases} P_0 & (P_0 \geq 0) \\ P_0 + Number\ of\ Groups & (P_0 < 0) \end{cases}$$

We explain the example of the prioritization mechanism, where the number of groups is 4. There are 4 groups, *Group 0-3* and 4 buckets of hashing, *Bucket 0-3*. For peers(Peer id = 1,5,9,...) belonging to *Group 1*, *Bucket 1*(Key = 1,5,9,...)'s *Priority* is 0, *Bucket 2*(Key = 2,6,10,...)'s *Priority* is 1, *Bucket 3*(Key = 3,7,11,...)'s *Priority* is 2, and *Bucket 0*(Key = 4,8,12,...)'s *Priority* is 3.

When K-Hop is used as the distribution function, the priority of each data group is set using the following formula.

$$Priority = Hop\ Count\ from\ The\ Data\ Source \mod K$$

In these methods, 0 is the highest priority.

For deciding what data to store, we consider three options:

- Fixed Redundancy(FR): Each peer stores data belonging to the top N priority groups, where N is a fixed value.
- Adaptive Redundancy(AR): Each peer stores data which belonging to the top N priority groups, where N is set by the following adaptive formula:

  $$N = N_0 - \frac{R_{max} - 1}{\alpha}$$

  where

  $N_0 : Initial\ Degree\ of\ Redundancy$

  $R_{max} : Max(R_i)$

  $R_i : Frequency\ of\ receiving\ data\ item\ d_i$

  α : An adaptation factor

  We explain this algorithm using the example where $N_0 = 4$ and $\alpha = 3$. When data items d_1, d_2, and d_3 are stored in a peer's storage and the peer has received d_1 7 times, d_2 3 times, and d_3 5 times respectively, R_{max} becomes 7, where $i = 1$, and N becomes 2. Generally, as the car density increases, the number of receiving each data item increases. Thus, this algorithm can adapt N to the car density.
- Adaptive Redundancy with Deletion(ARw/D): Each peer stores data in the same way as with AR. However, this method drops all data which belong to top *N+1* or more priority groups when N was changed. For example, the data belonging to the group whose priority(beginning with 0) is 3 are dropped when N was changed from 4 to 3.

3.2 Storage Management Algorithm

The storage management function decides what data to drop when there is no room in the storage area. When each peer is using the redundancy control function, the data which belongs to the lowest priority group is dropped.

When all stored data belong to the highest priority group or when each peer is not using the redundancy control function, the peer decides which data to drop using the following method:

- No Deletion(NoDel): Each peer does not drop any data in its storage. Thus, once the peer's storage is occupied by the data of the highest priority group, the peer does not store other data.
- Delete the Oldest(DelOld): Each peer drops the oldest data in its storage. Thus, each peer manages its storage using a first-in-first-out manner.

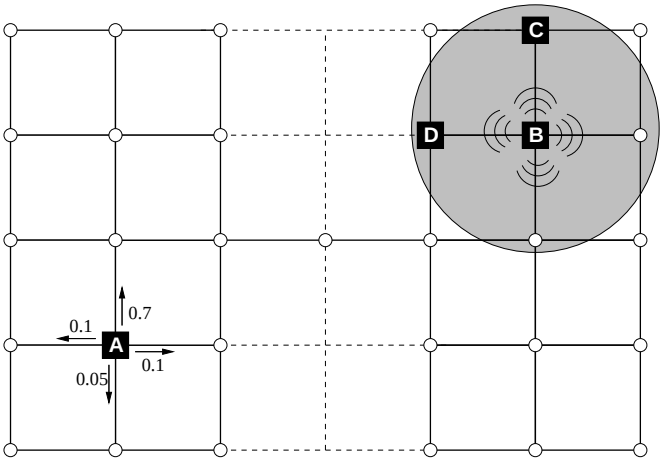

Figure 1: **Our model**

- Delete Depending on the Generation of Replication(RplGen): Each peer drops the data item which has been most often replicated when it was stored at the peer. Generally, the more often the data was replicated, the more copies are generated. Thus, each peer is likely to delete the data which has more copies in the system.
- Delete Randomly(DelRand): Each peer randomly picks up the data to drop.

3.3 Query Processing Algorithm

The query processing function has a two sub-functions, a querying function and a answering function.

- Querying Function: Each peer uses the querying function when it wants to retrieve data. This function generates a query to retrieve the data and broadcasts it within a one-hop area.
- Answering Function: Each peer uses the answering function when it received a query. This function checks if the peer holds the answer and broadcasts the answer within a one-hop area if the peer holds the answer.

4 Modeling a Data Sharing Mechanism

In order to study our data sharing mechanism in mobile P2P systems, we use a simulation model of automobiles moving around in physical space. Each car is treated as a mobile peer and each car has wireless communication ability.

We use a *map* to model the data sharing area. The *map* is a regular two dimensional grid (see Figure 1). Each intersection of the grid is a *location* which corresponds to some part of a road in real space. There may be zero or more cars at a *location*. Cars move from one location to another location along an edge.

To simulate the motion of cars on the map, we use a biased random mobility model. Cars move at regular intervals. A car is likely to move in the same direction that it moved during the last interval, but it can change the direction or not move with some fixed probability. For example, Car A in Figure 1 moves up with possibility 0.7, moves right or left with possibility 0.1, moves down with possibility 0.05, and stays in place with possibility 0.05. When a car reaches the edge of a map, it can exit from the map. Once a car leaves, it can't communicate with other cars in the map and can't reenter the map. When a car exits from the map, another car enters the map. Thus, the density of cars is kept constant.

To simulate communication among peers, we use a wireless communication model. Each car can wirelessly transmit a message within a limited range. All the cars in the range of the car hear the message. For Example, in Figure 1, Car B has a communication radius of 1, so Cars C and D hear

its broadcast. We also consider a collision of transmissions. For handling collisions, we use a basic CSMA MAC protocol; a car wants to transmit will do if it detects that channel is clear.

The simulation goes through the following phases:

1. Data Generation Phase: All data are generated at the beginning of the simulation. Some cars are randomly selected as the data sources. Data sources generate data and transmit it.

2. Data storage phase: The cars move around the map, store data using the data storage function and the storage management function, and disseminate data.

3. Data Retrieval Phase: The cars retrieve data using the query processing function in this phase. In this simulation, we do not actually make the cars process queries. In order to evaluate the query processing, we examine whether each data item is stored in the one-hop area of each car.

For simplicity, we execute these phases separately and every data item is simultaneously generated at the beginning of the simulation, but we believe this simulation still capture the essential features of the data sharing mechanism.

4.1 Metrics

We evaluate the efficacy of the data sharing mechanism on data loss, data accessability, communication overhead, and storage cost.

Data Keep Rate(DKR). We define data keep rate to be the ratio of the number of data items kept in the data sharing area to the total number of items. More formally:

$$Data\ Keep\ Rate = \frac{Number\ of\ data\ items\ kept\ in\ the\ data\ sharing\ area}{Total\ number\ of\ data\ items}$$

High DKR means low data loss. Thus, we can evaluate on data loss using the DKR.

Successful Query Rate(SQR). This is the average ratio of queries which succeed in retrieving data to the total number of queries. More formally:

$$Successful\ Query\ Rate = \frac{Number\ of\ queries\ that\ succeeded\ in\ retrieving\ data}{Total\ number\ of\ queries}$$

A query can succeed in retrieving data only when the data is held by a peer within the one-hop area of the peer. So, we redefine SQR to be the average ratio of the number of data items kept in the one-hop area of each peer to the total number of items. More formally:

$$sq_{ij} = \begin{cases} 1 & (item_i\ is\ kept\ in\ the\ one-hop\ area\ of\ peer_j) \\ 0 & (otherwise) \end{cases}$$

$$SQR_j = \frac{\sum_i sq_{ij}}{Total\ number\ of\ data\ items}$$

$$Successful\ Query\ Rate = \frac{\sum_j SQR_j}{Total\ number\ of\ peers}$$

The higher the SQR is, the more easily a peer accesses many data items. Notice that the SQR does not have to be 1. Even though a peer can't retrieve data at the moment, the peer has a chance to retrieve the data after moving around.

Number of Transmissions. The number of transmission is the total number of messages broadcast per single data item in the data storage phase. This is the communication overhead of data storage. We count the number of transmissions from the beginning to the end of a simulation.

Storage Consumption. The storage consumption is the average number of items in each peer's storage. This expresses the storage cost of the data sharing mechanism. We count the number at the end of a simulation.

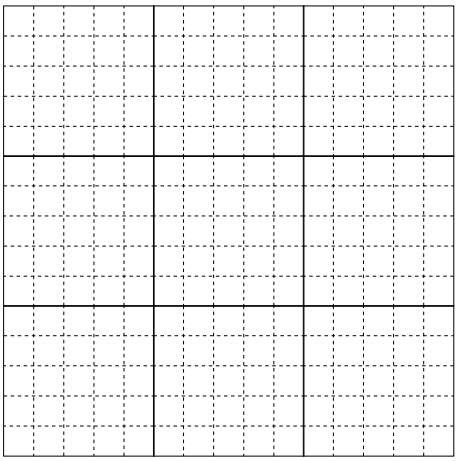

Figure 2: **City block layout**

4.2 Framework

4.2.1 Scenario Parameters

In order to simulate various settings, we use the following scenario parameters:

- **Map size** We describe the map size in terms of overall dimensions. In the simulations, we fix this parameter to a 41x41 grid.
- **Map layout** We use a 'city block' layout, illustrated in Figure 2, which is a large grid simulating the regular layout of streets in a city, where each block is five map locations per side.
- **Car density** We define the density to be the number of cars per map location where a car can be present. By default, we set the car density to 1.
- **Simulation time** This is the total amount of time that the simulation is run for measured in ticks (1000 ticks = 1 second). By default, we set the simulation time to 150,000.
- **Communication radius** This is the range of wireless communications. We fixed this parameter to a range of 2 map squares, meaning the message is visible to any car at a location within a distance of 2 grid square from the sender.
- **Communication bandwidth** We model limited communication bandwidth by time to transmit a message. The size of message used was 10,000 bits and the bandwidth was set to 1,000 bits/tick. Thus, we fixed the time to 10 ticks.
- **Car speed** A car moves at the speed selected uniformly from the interval [0, maximum speed]. By default, the maximum speed of a car is 1 grid square every 1000 ticks.
- **Car motion** In the simulations, a car moves in the same direction that it moved during the last interval with possibility 0.7, turns right with possibility 0.1, turns left with possibility 0.1, reverses with possibility 0.05, and does not move at all with possibility 0.05.
- **Car exit probability** This is the probability with which a car exits from a map when it located at the edge of the map. We fixed this probability to 0.25.
- **Total number of data items in a table** This is the total number of items in the table which is stored in the system. We fixed this number to 100.

Table 1: **Scenario Parameters**

Parameter	Default value
Map Size	41x41 grid
Map Layout	city block layout
Car Density	1 car/location
Simulation Time	150,000 ticks
Communication Radius	2 map squares
Communication bandwidth	1,000 bits/tick
Max Car Speed	1/1000 grid square/tick
Car Motion	go straight:0.7, turn right:0.1, turn left:0.1, reverse:0.05, and do not move: 0.05
Car Exit Probability	0.25
Total Number of Data Items in a Table	100

4.2.2 Protocol Parameters

Each protocol has its own protocol parameters. In the simulations, we seek the best value for each parameter.

1. Data Distribution
   (a) HID: The parameter of this protocol is the number of buckets used for hashing. This parameter has an impact on data distribution and storage consumption.
   (b) K-Hop: This parameter the interval for storing, which is described in terms of hop count, K. This parameter also has an impact on data distribution and storage consumption.
2. Redundancy Control
   (a) FR: This parameter is the degree of redundancy N, that is, the number of priority groups stored in a peer.
   (b) AR and ARw/D: These parameters are the the initial degree of redundancy N_0, and the adaptation factor α. The adaptation factor α determines the sensitivity of the redundancy control to message flux.
3. Storage Management: This parameter defines storage size. It has an impact on data distribution and storage consumption.
4. Data Dissemination: These parameters are the suppression threshold and the delay interval. Both parameters have an impact on data distribution and the number of transmissions. Referring to the result of [1], we fixed the delay interval to [5000, 5500] ticks.

4.2.3 Default Settings

The default values of the scenario parameters and the protocol parameters in the experiments are summarized in Table 1 and Table 2 respectively.

5 Result

In this section, we show the results of simulations to evaluate the effect of communication protocols, data storage protocols and storage management protocols.

5.1 Effect of Communication Protocols

In this section, we show the the trends of TTL-based dissemination and SR-based dissemination and the comparison of SR and TTL.

Table 2: **Protocol Parameters**

Protocol	Parameter	Default value
HID	Number of Buckets	4
K-Hop	Interval of Storing	4
FR	Degree of Redundancy	3
AR & ARw/D	Initial Degree of Redundancy N_0	4
	Adaptation Factor α	5
Storage Management	Storage Size	unlimitted
TTL	TTL	10
	Delay Interval	[5000, 5500] ticks
SR	SR Count	4
	Delay Interval	[5000, 5500] ticks

5.1.1 Trends of TTL-based Dissemination

Our results show that the DKR(Data Keep Rate) is almost 1 for every TTL. When TTL is greater than 8 at least, data can be kept in the data sharing area with high probability.

The relationship between the suppression threshold, SQR(Successful Query Rate), and the communication overhead for TTL-based dissemination is captured by Figure 3 and 4. In this simulation, we vary TTL from 8 to 40.

Figure 3 plots the effect of varying the TTL on SQR. The graph shows that the SQR is proportional to the TTL. As the degree of suppression is decreased, data is forwarded a greater number of times. Thus, each item is likely to be stored at more peers as the TTL increases and the SQR is likely to become higher.

Figure 4 plots the effect of varying the TTL on communication overhead. The graph shows that the communication overhead is proportional to the TTL. Reducing the degree of suppression will obviously increase the number of transmissions.

Thus, these results show that increasing the TTL can yield a higher SQR but requires a higher communication overhead.

5.1.2 Trends of SR-based Dissemination

Our results show that the DKR is almost 1 for every SR count. When we use the SR-based dissemination, data can be kept in the data sharing area with high probability.

The relationship between the suppression threshold, SQR, and the communication overhead for SR-based dissemination is captured by Figure 5 and 6. In this simulation, we vary SR count from 2 to 8.

Figure 5 plots the effect of varying the SR count on SQR. The graph shows that the SQR becomes higher as the SR increases. As the degree of suppression is decreased, data is forwarded a greater number of times. Thus, each data is likely to be stored at more peers as the SR count increases.

Figure 6 plots the effect of varying the SR count on communication overhead. The graph shows that the communication overhead becomes higher as the SR count increases. Reducing the degree of suppression will obviously increase the number of transmissions.

Similar to the TTL-based dissemination, increasing the SR can yield a higher SQR but requires a higher communication overhead.

5.1.3 Comparison of SR vs. TTL

To select a dissemination protocol adequate for the data sharing mechanism, we compare SR-based dissemination and TTL-based one. The DKR is almost 1 in both cases. So, we compare them on the SQR and the communication overhead.

Each point in Figure 7 gives the SQR and the communication overhead for a given SR count or TTL. As we travel the SR plot from left to right, the SR count increases from 2 to 8. Similarlr, for

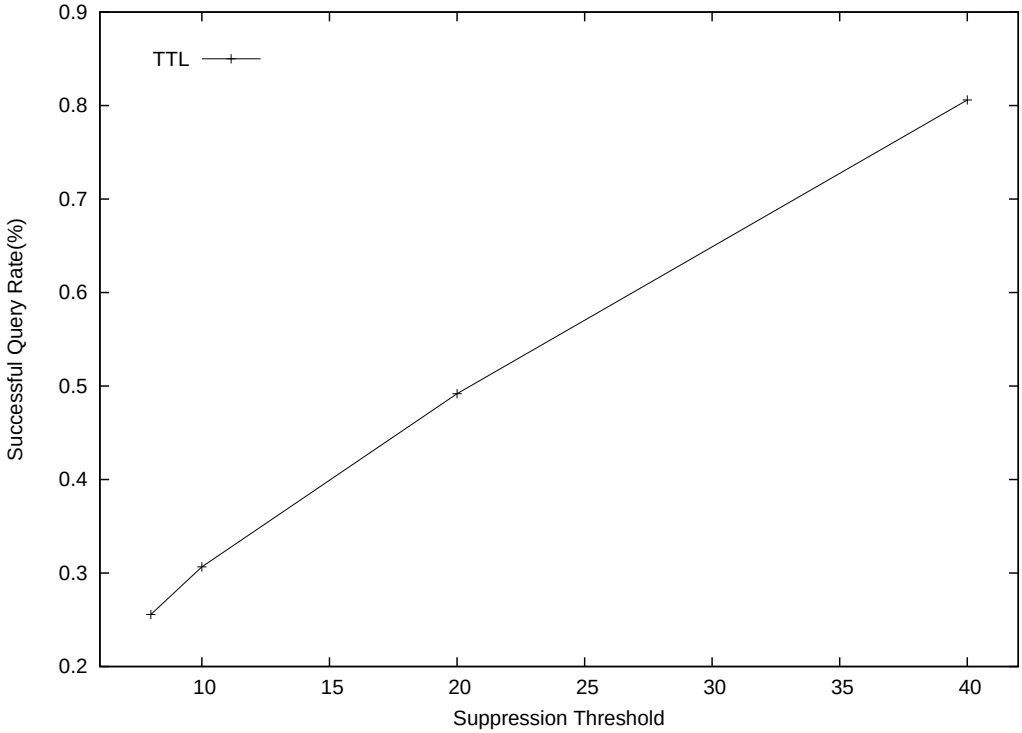

Figure 3: **TTL vs. successful query rate**

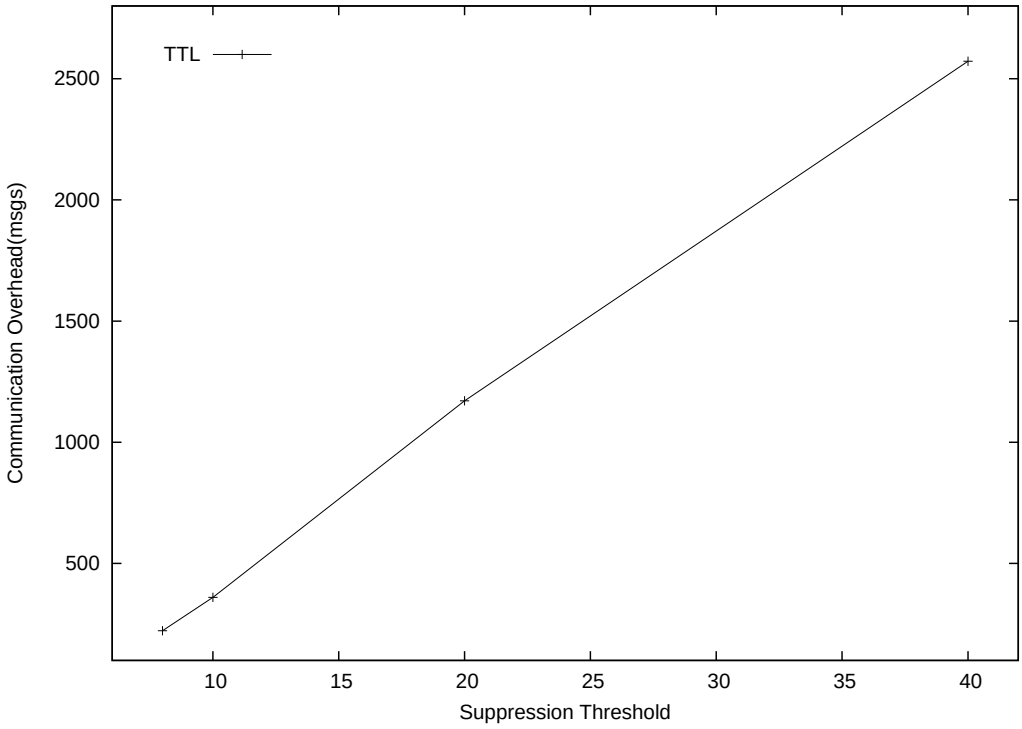

Figure 4: **TTL vs. communication overhead**

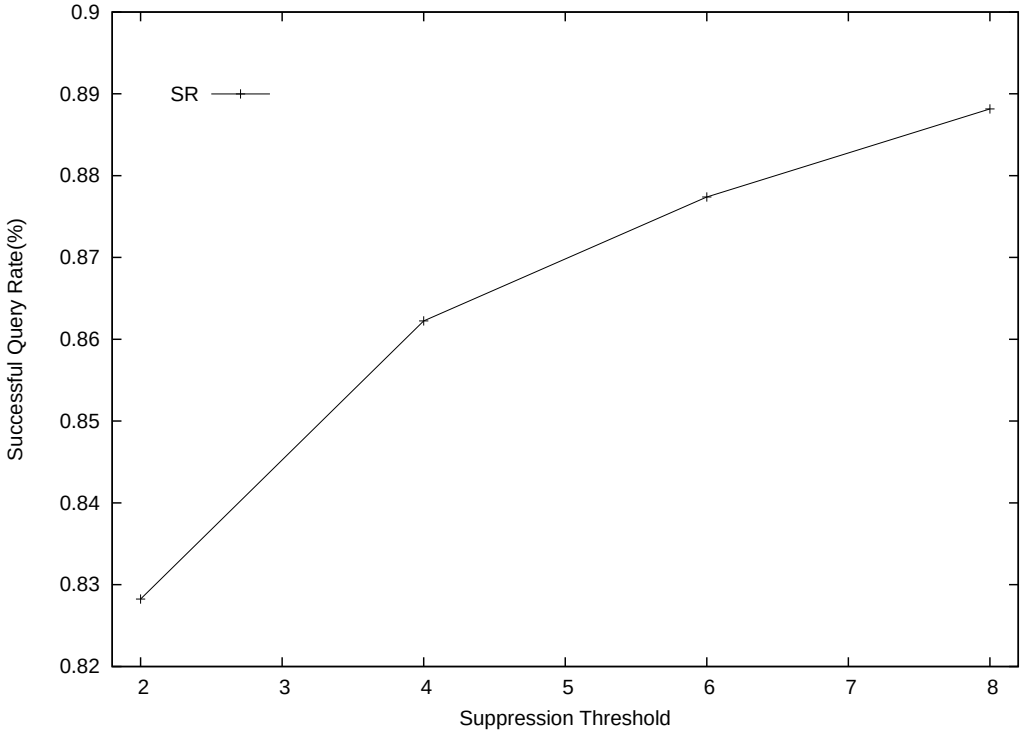

Figure 5: **SR count vs. successful query rate**

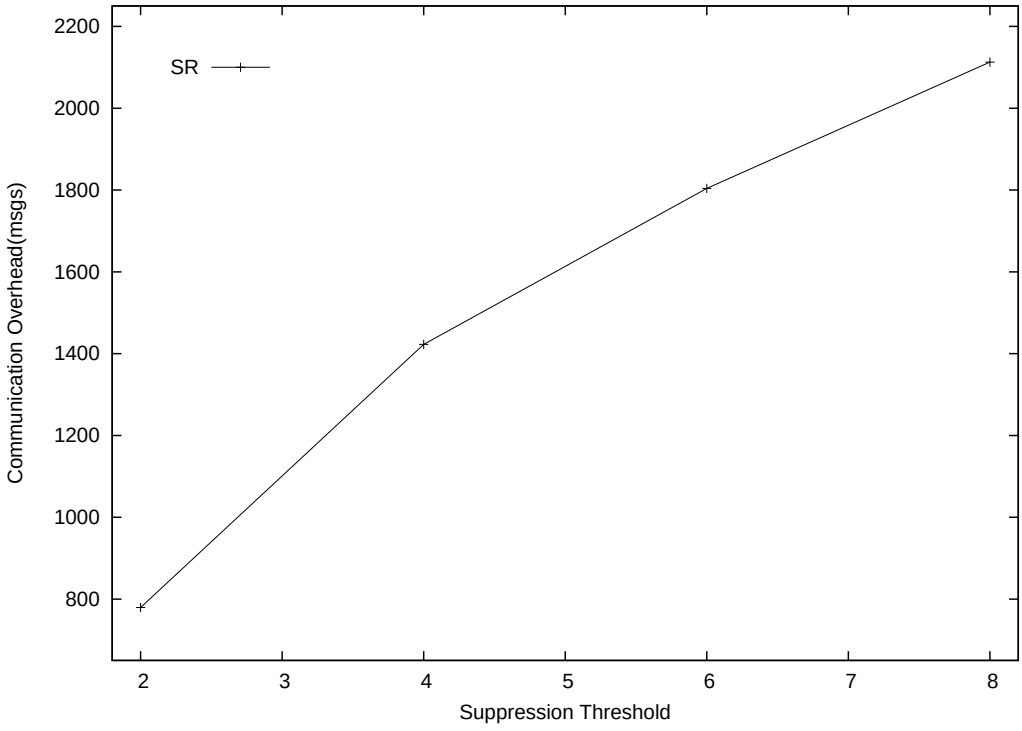

Figure 6: **SR count vs. communication overhead**

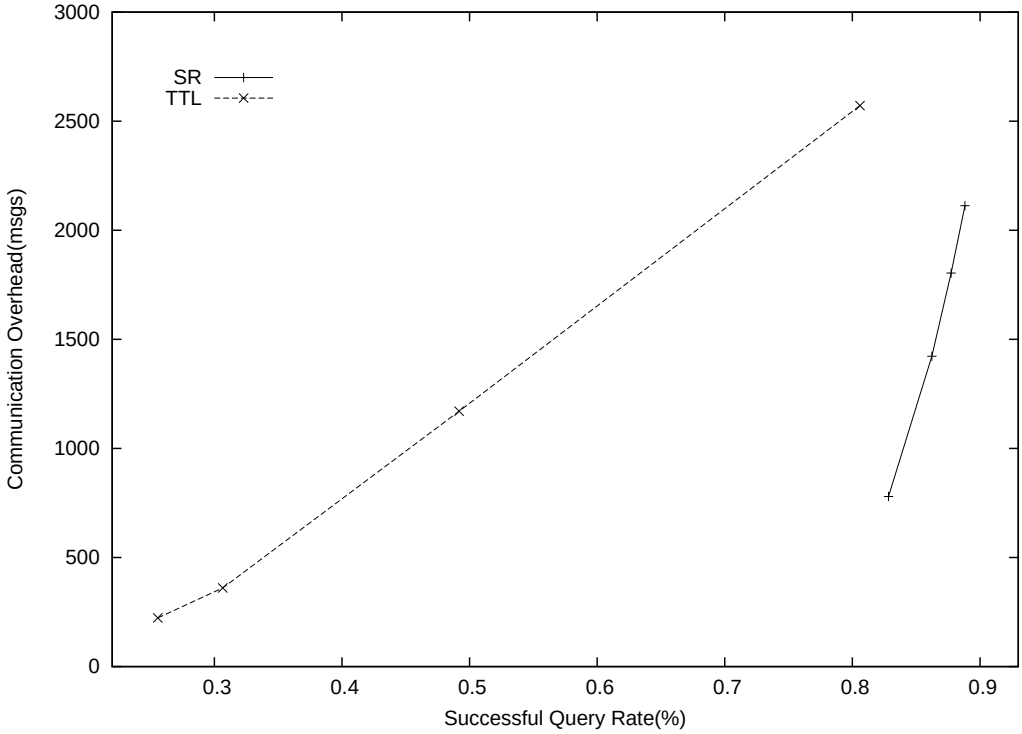

Figure 7: **SR vs. TTL**

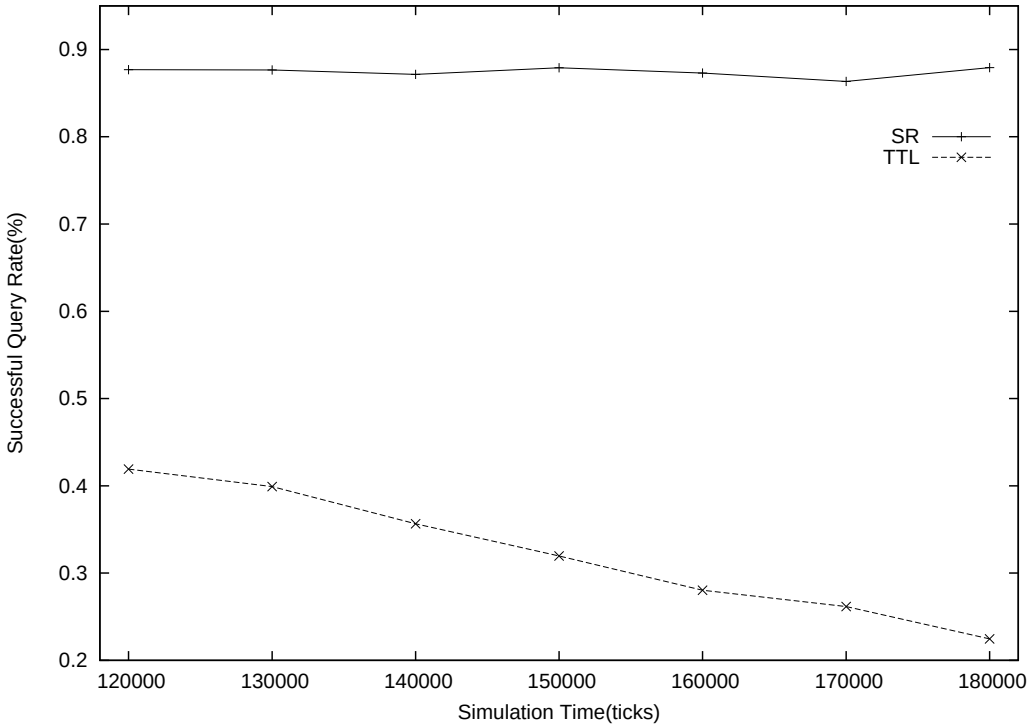

Figure 8: **Effect of simulation time**

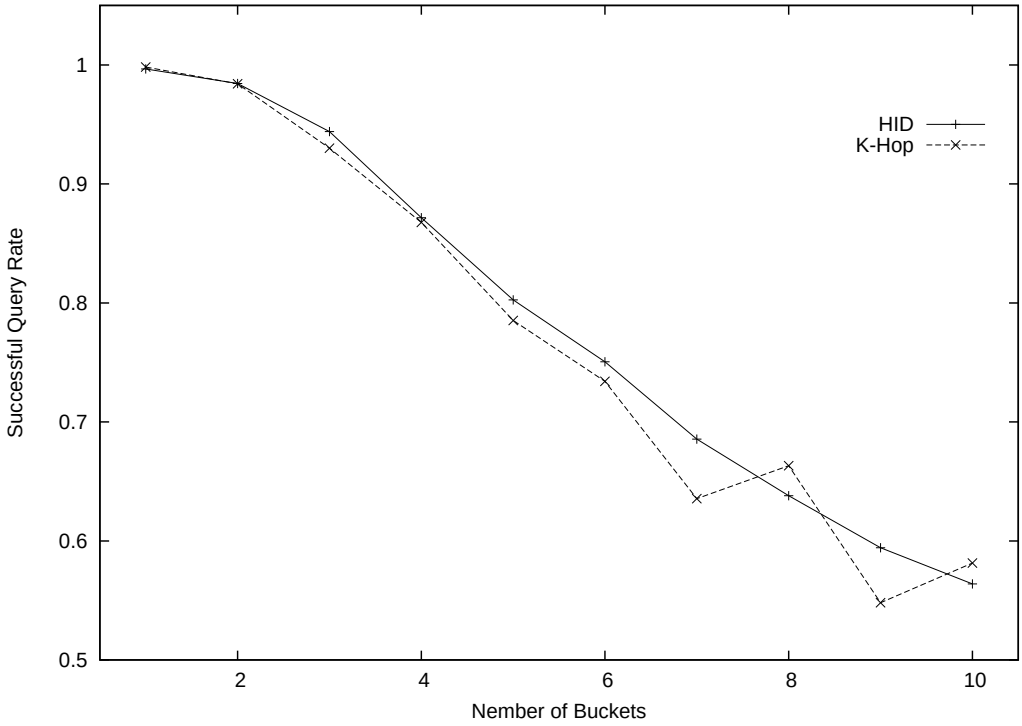

Figure 9: **Number of buckets vs. successful query rate**

the TTL plot from left to right, the TTL increases from 8 to 40. The graph shows that SR-based dissemination realizes higher SQR with less communication overhead, that is, SR-based dissemination is more efficient than he TTL-based one.

Figure 8 plots the effect of varying the simulation time on the SQR for the SR-based dissemination with SR count of 4 and the TTL-based dissemination with TTL of 10. The graph shows that the SQR for the SR-based dissemination is constant as the simulation time increases and the SQR for the TTL-based dissemination decreases as the simulation time increases.

For the TTL-based dissemination, after the TTL of certain data becomes 0, the data can't be forwarded any more and nobody can increase the TTL. So the cars, which entered to the map after the TTL is exhausted, can't receive and store the data. Thus, the SQR becomes lower as the ratio of the cars which entered to the map after the TTL is exhausted.

For the SR-based dissemination, each car has its own suppression threshold. Even if the cars which entered to the map earlier have run out of the SR count, the cars which entered later can forward the data because they still have valid SR count. So data keeps on being forwarded. Thus, the SR-based dissemination is adaptive to the simulation time.

Thus, we select the SR-based as the dissemination protocol. And we use the SR count of 4 because it is most cost-effective as shown in Figure 7.

5.2 Effect of Data Storage Protocols

In this section, we show the results of simulations to evaluate the effect of distribution protocols and redundancy control protocols.

5.2.1 Effect of Distribution Protocols

The relationship between the number of buckets, SQR, and the storage consumption is captured by Figure 9 and 10. In this simulation, we vary the number of buckets from 1 to 10 and we use two protocols, HID and K-Hop. In the case of K-Hop, we treat the interval for storing K as the number of buckets.

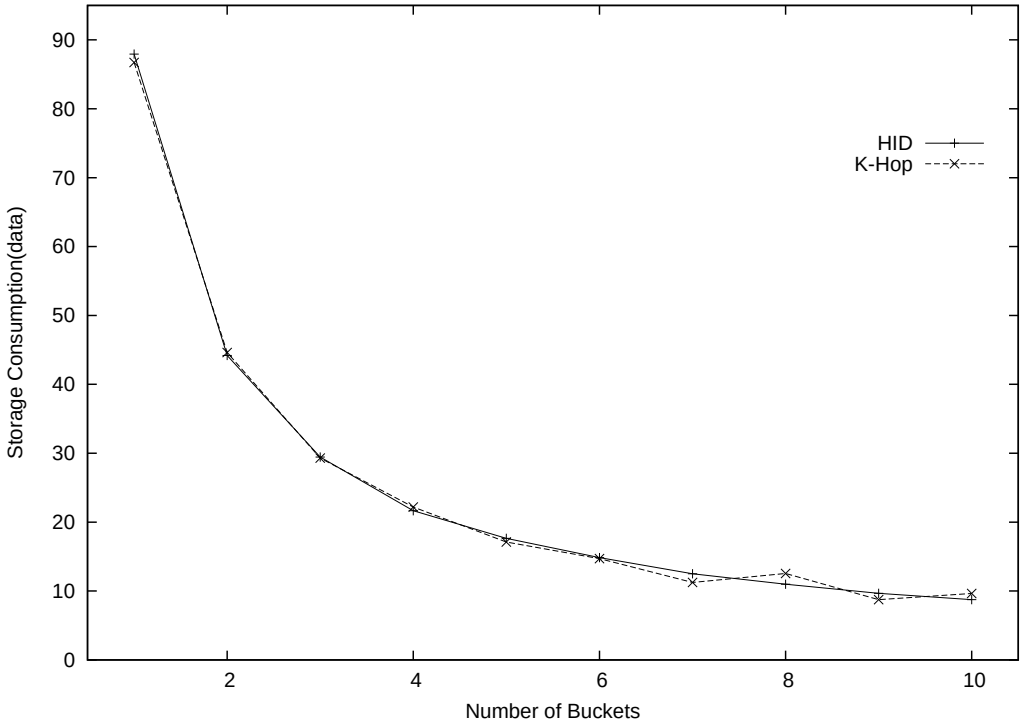

Figure 10: **Number of buckets vs. storage consumption**

Figure 9 plots the effect of varying the number of buckets on SQR. The graph shows that the SQR becomes lower as the number of buckets increases. As the number of buckets increases, the number of storing data becomes lesser because the probability of storing certain data at a peer is the inverse of the number of buckets. Thus, each data is likely to be held in the one-hop area of each peer when we use smaller number of buckets.

Figure 10 plots the effect of varying the number of buckets on storage consumption. The graph shows that the storage consumption becomes smaller as the number of buckets increases. As the number of buckets increase, the number of data items to be stored at a peer becomes lesser because the maximum number of data items to be stored at a peer is expressed by the following formula:

$$\frac{Total\ number\ of\ data}{Number\ of\ buckets}$$

Our results show that the DKR is almost 1 in every case. Thus, using either the HID or the K-Hop, data can be kept in the data sharing area with high probability.

Each point in Figure 11 gives the SQR and the storage consumption for a given number of buckets. As we travel the HID plot from left to right, the number of buckets decreases from 10 to 1. Similarly, for the K-Hop plot from left to right, the number of buckets decreases from 10 to 1. The graph shows that there is no significant difference between HID and K-Hop. This shows that K-Hop protocol has a same effect with a hashing. Thus, when we can't get the id of a peer for some reasons, we can use the K-Hop as the distribution protocol.

In this graph, the points in the lower right hand quadrant of the figure are preferable because simulations in this position will have low storage consumption and high SQR. Thus, we use the number of buckets of 4 as a default.

As shown in Figure 9, when we use the distribution protocol, which means that we use the number of buckets of more than 2, the SQR is lower than the SQR achieved when we do not use the distribution protocol, which means that we use the number of buckets of 1. But as shown in Figure 10, when we do not use the distribution protocol, the storage consumption is much larger than the SQR achieved when we use the distribution protocol. So there is the tradeoff between the SQR and the storage consumption, To solve this tradeoff, we use the redundancy control.

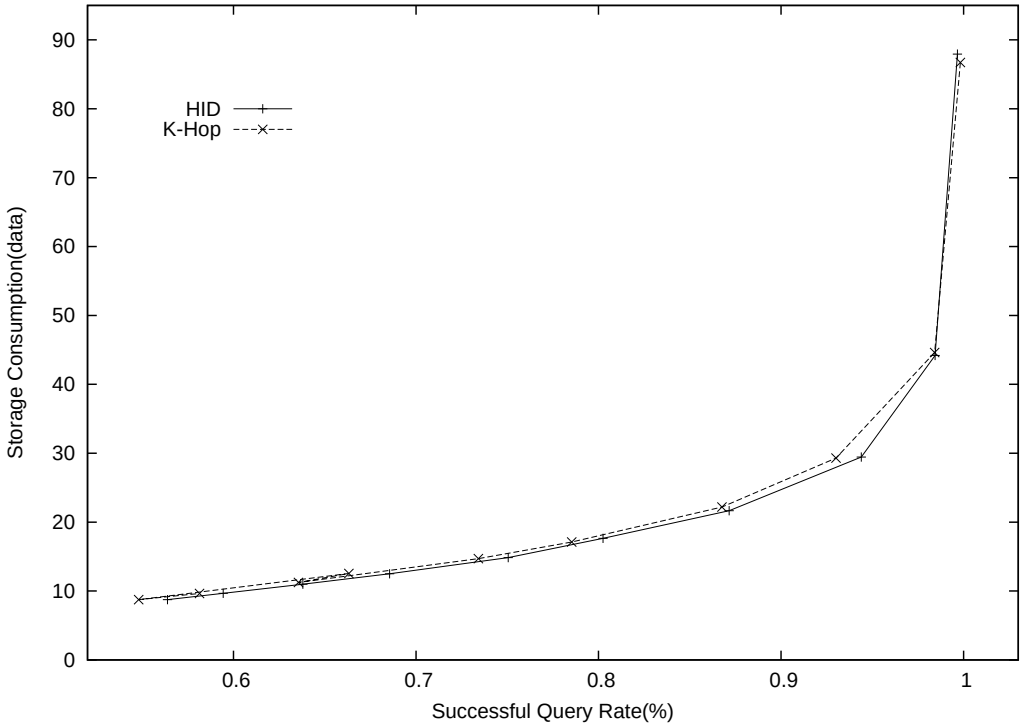

Figure 11: **HID vs. K-Hop**

5.2.2 Effect of Redundancy Protocol: Simple Model (FR)

The relationship between the degree of redundancy, SQR, and the storage consumption is captured by Figure 12 and 13. In this simulation, we vary the degree of redundancy from 1 to 4.

Figure 12 plots the effect of varying the degree of redundancy on SQR. The graph shows that the SQR becomes higher as the degree of redundancy increases. As the degree of redundancy increases, the number of storing data becomes higher because the probability of storing certain data at a peer is proportional to the degree of redundancy. Thus, each data is likely to be held in the one-hop area of each peer with high probability when we use larger degree of redundancy.

Figure 13 plots the effect of varying the degree of redundancy on storage consumption. The graph shows that the storage consumption becomes bigger as the degree of redundancy increases. As the degree of redundancy increases, the number of data items to be stored at a peer becomes larger because the maximum number of data items to be stored at a peer is expressed by the following formula:

$$\frac{Total\ number\ of\ data}{Number\ of\ buckets} \times Degree\ of\ redundancy$$

As shown above, the FR protocol improve the SQR but it consumes more storage. Actually, increasing the degree of redundancy has just the same effect with decreasing the number of buckets. For example, the number of data items to be stored at a peer using the number of buckets of 4 with the degree of redundancy of 2 is same with the number of data items to be stored at a peer using the number of buckets of 2 without redundancy. In both cases, the SQR is about 0.98(see Figure 9 and 12) and the storage consumption is about 44(see Figure 10 and 13). Moreover, the number of data items to be stored at a peer using the number of buckets of 4 with the degree of redundancy of 4 is same with the number of data items to be stored at a peer using the number of buckets of 1 without redundancy. In both cases, the SQR is about 1(see Figure 9 and 12) and the storage consumption is about 88(see Figure 10 and 13). Thus, the simple redundancy model can't solve the tradeoff between the SQR and the storage consumption.

Figure 14 plots the impact of varying the car dencity on SQR. The graph shows that the SQR

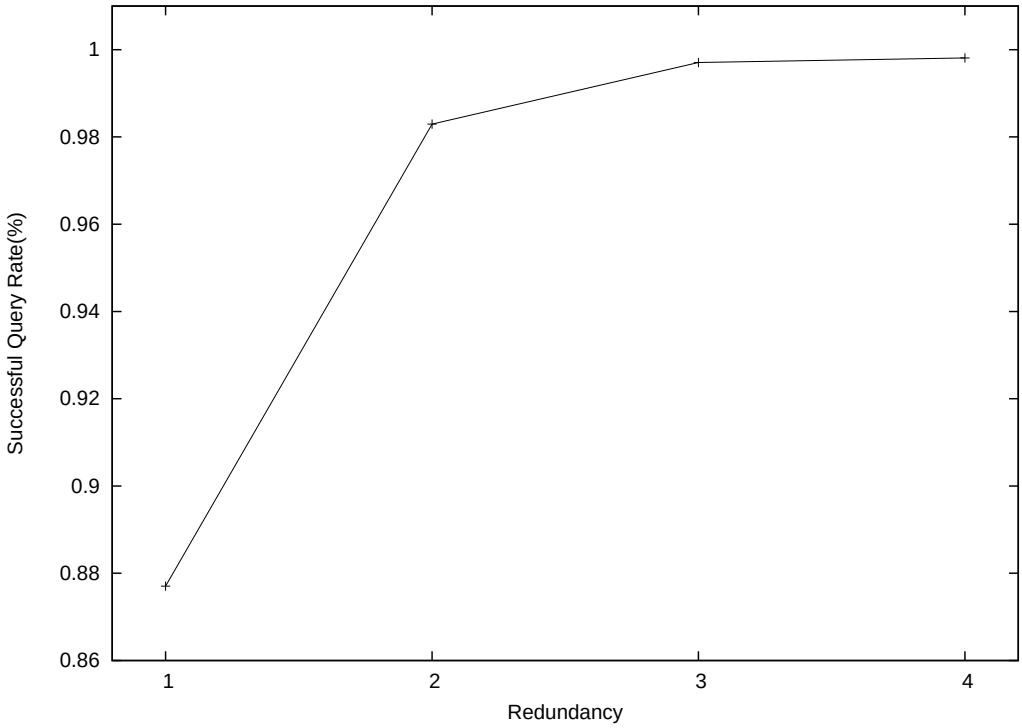

Figure 12: **Degree of redundancy vs. successful query rate**

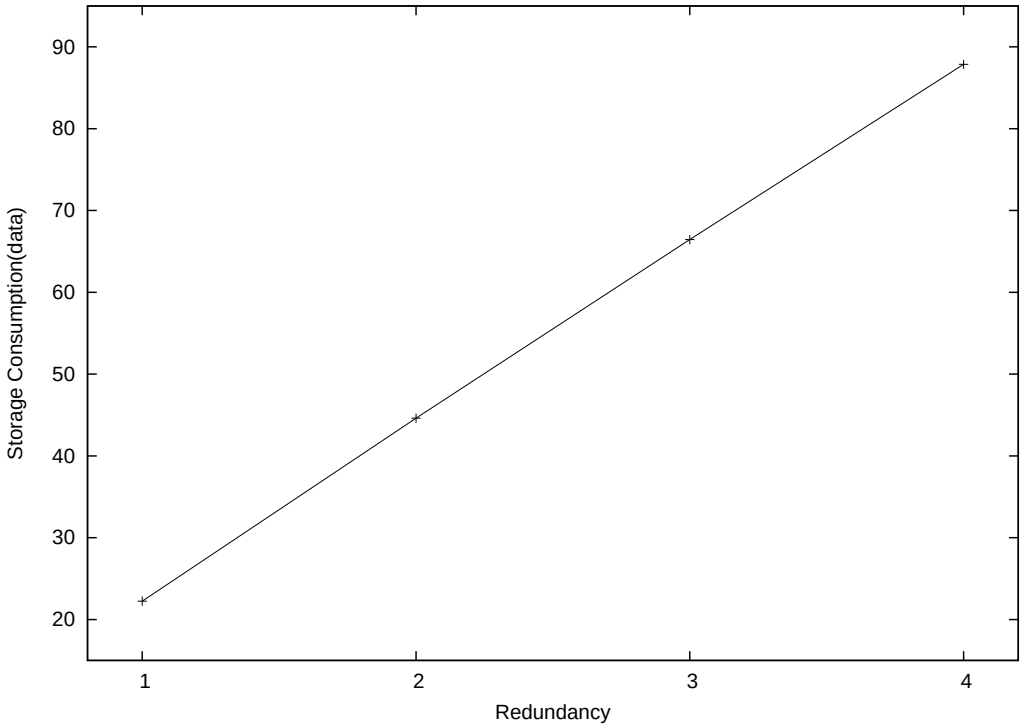

Figure 13: **Degree of redundancy vs. storage consumption**

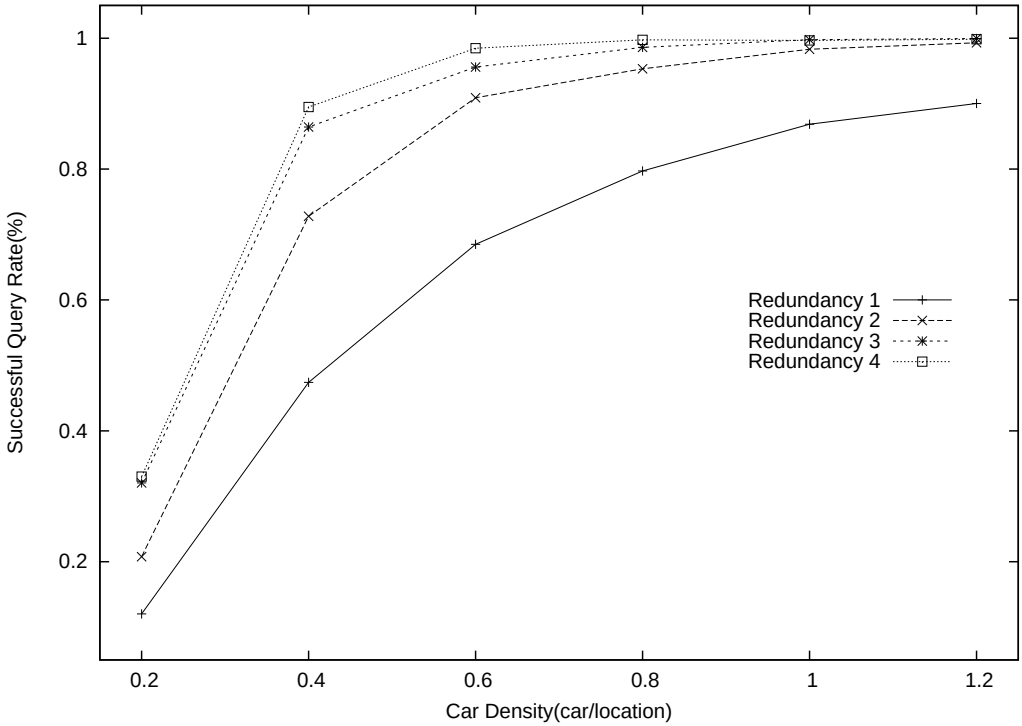

Figure 14: **Car density vs. successful query rate for FR**

becomes higher as the car density increases and that the lower the car density is, the bigger the effect of redundancy is. This means that higher redundancy is needed at lower density but that low redundancy is enough at higher density. Using this property, we solve the tradeoff between the SQR and the storage consumption. So we consider the redundancy adaptation to the car density.

5.2.3 Effect of Redundancy Adaptation: AR and ARw/D

The relationship between the car density, SQR, and the storage consumption is captured by Figure 15 and 16. In this simulation, we vary the adaptation factor α from 1 to 7 and we use two protocols, AR and ARw/D.

Figure 15 plots the impact of varying the car density on SQR. "adaptation factor 1-7" plots show the SQR of AR with the adaptation factor α of from 1 to 7, "adaptation factor 1-7 w/deletion" plots show the SQR of ARw/D with the adaptation factor α of from 1 to 7, and "Original" shows the SQR of the data storage without distribution (in which each peer can store every data). The graph shows that in the most of the case except AR with the adaptation factor α of 1 and ARw/D with the adaptation factor α of 1 and 3, the redundancy adaptation protocols achieve as high SQR as the data storage without distribution.

Figure 16 plots the impact of varying the car density on storage consumption. "adaptation factor 1-7" plots show the storage consumption of AR with the adaptation factor α of from 1 to 7, "adaptation factor 1-7 w/deletion" plots show the storage consumption of ARw/D with the adaptation factor α of from 1 to 7, and "Original" shows the storage consumption of the data storage without distribution. The graph shows that in the most of the case except AR with the adaptation factor α of 5 and 7, the redundancy adaptation protocols consume smaller storage than the data storage without distribution.

Using Figure 17, we compare the AR, ARw/D, and the data storage without distribution. As we travel each plot from the left to right the car density increases from 0.2 to 1.2. This graph shows that ARw/D achieves as high SQR as the data storage without distribution with less storage consumption.

The relationship between the adaptation factor α, SQR, and the storage consumption is captured by Figure 18. "density 0.2" plots show the storage consumption of AR at the car density is 0.2,"den-

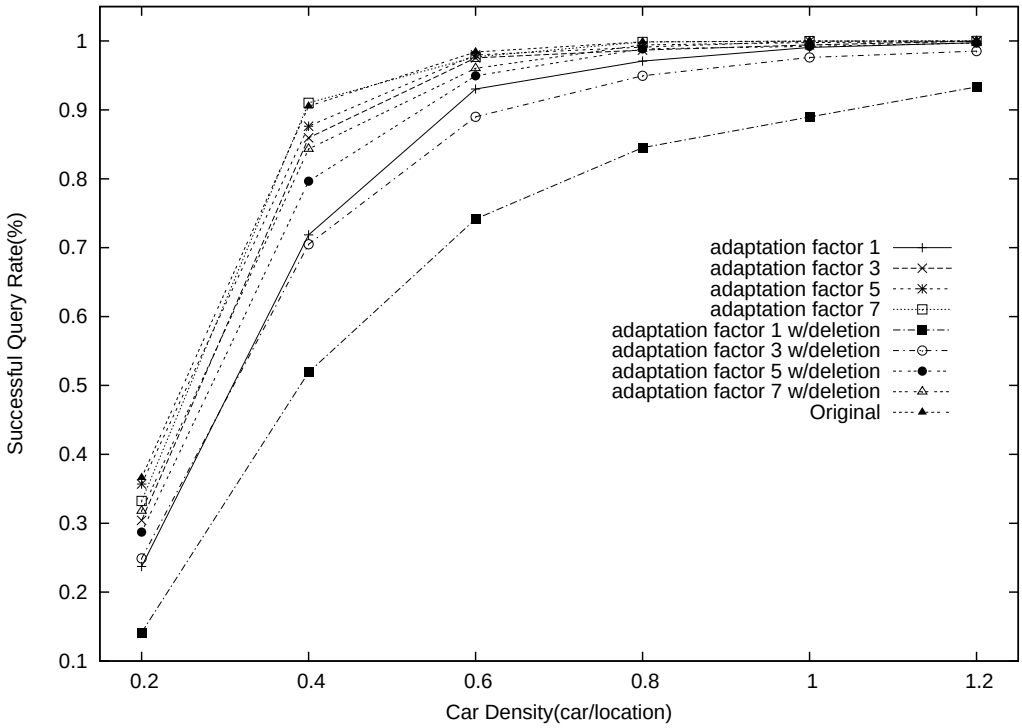

Figure 15: **Car density vs. successful query rate for AR and ARw/D**

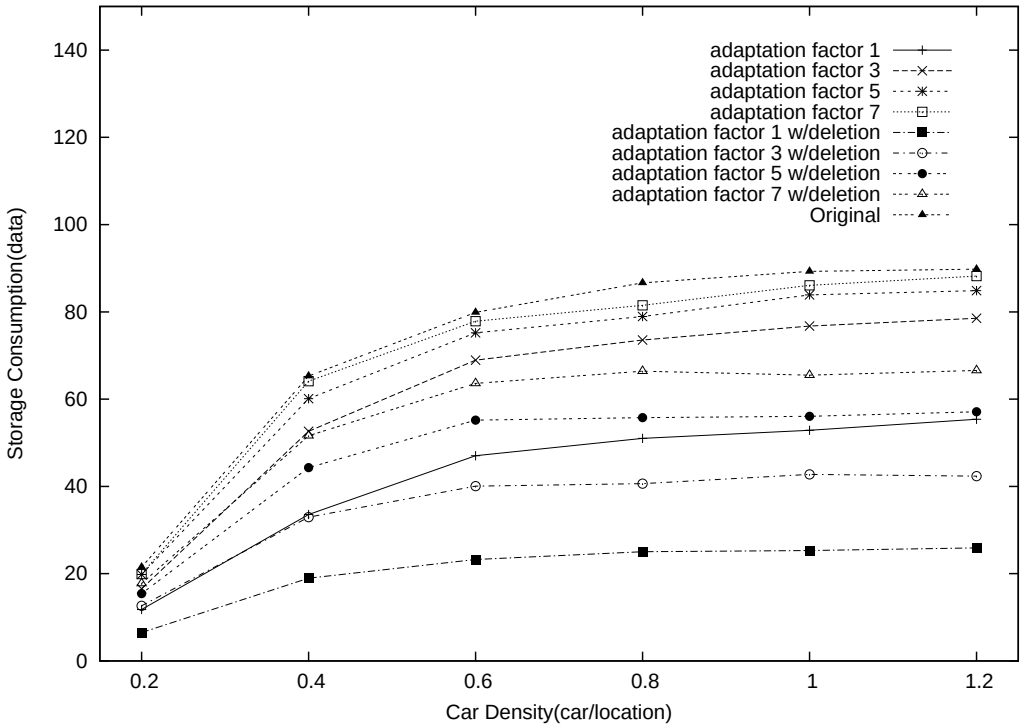

Figure 16: **Car density vs. storage consumption for AR and ARw/D**

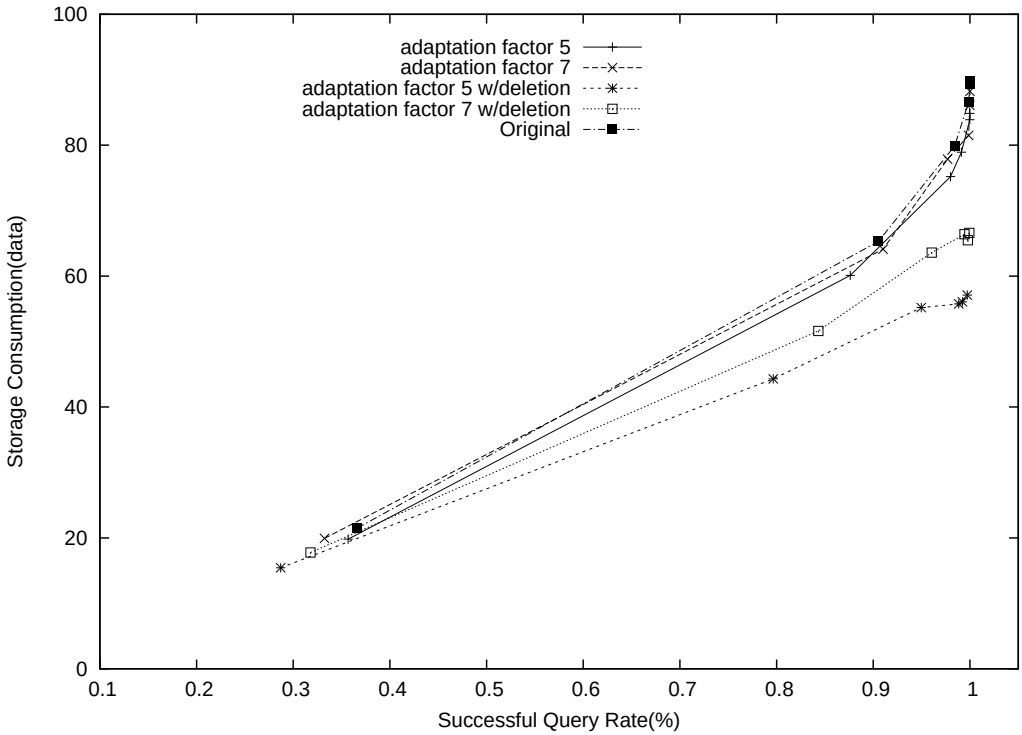

Figure 17: **Successful query rate vs. storage consumption for AR and ARw/D**

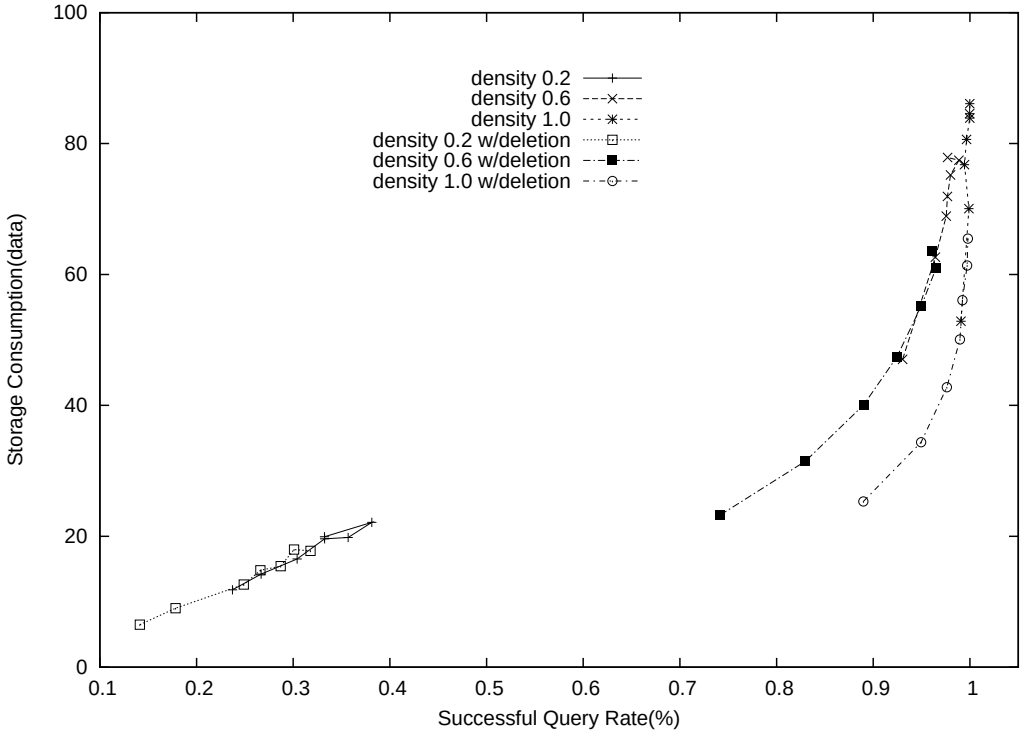

Figure 18: **Successful query rate vs. storage consumption at various car densities**

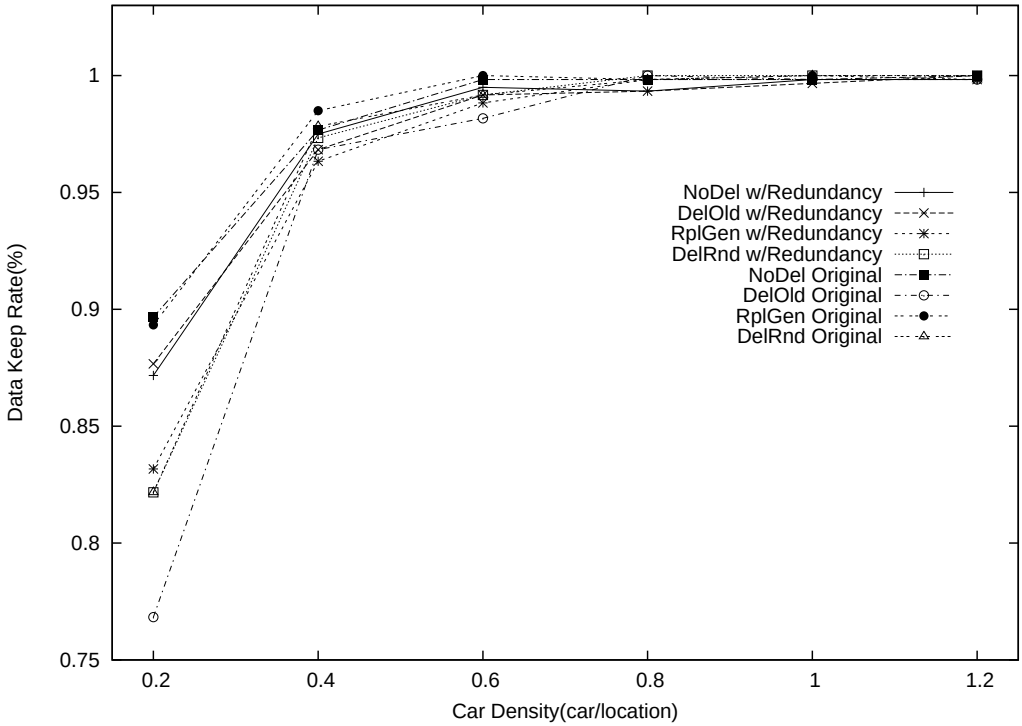

Figure 19: **Car density vs. data keep rate at storage size = 20**

sity 0.6" plots show the storage consumption of AR at the car density is 0.6, and so on. "density 0.2w/deletion" plots show the storage consumption of ARw/D at the car density is 0.2,"density 0.6w/deletion" plots show the storage consumption of ARw/D at the car density is 0.6, and so on. As we travel each plot from the left to right, the adaptation factor α increases from 1 to 7. This graph shows that ARw/D with the adaptation factor α of 5 is most cost-effective, that is, it achieves higher SQR with lesser storage consumption.

5.3 Effect of Storage Management Protocols

The relationship between the car density, DKR, SQR, and the storage consumption is captured by Figure 19-21, Figure 22-24 and Figure 25-27. In this simulation, we vary the adaptation factor α from 1 to 7 and we use two protocols, AR and ARw/D.

Figure 19-21 plot the impact of varying the car density on DKR. Figure 19 shows the result when the storage size of each peer is 20, Figure 20 shows the result when the storage size of each peer is 40, and Figure 21 shows the result when the storage size of each peer is 60. In these graphs, "NoDel w/Redundancy" plots show the DKR of ARw/D using Nodel, "DelOld w/Redundancy" plots show the DKR of ARw/D using DelOld, "RplGen w/Redundancy" plots show the DKR of ARw/D using RplGen, "DelTnd w/Redundancy" plots show the DKR of ARw/D using DelRnd, "NoDel Original" plots show the DKR of the data storage without distribution using Nodel, "DelOld Original" plots show the DKR of the data storage without distribution using DelOld, "RplGen Original" plots show the DKR of the data storage without distribution using RplGen, and "DelTnd Original" plots show the DKR of the data storage without distribution using DelRnd.

The graphs show that there is no significant difference among the storage management protocols when the car density is higher(greater than 0.4). In the case of very low density(0.2), there is no clear trends of the protocols. The reason of this is the storage management protocol does not work well because the number of storing data at a peer is very small.

Figure 22-24 plot the impact of varying the car density on SQR. Figure 22 shows the result when the storage size of each peer is 20, Figure 23 shows the result when the storage size of each peer is 40, and Figure 24 shows the result when the storage size of each peer is 60. In these graphs, "NoDel

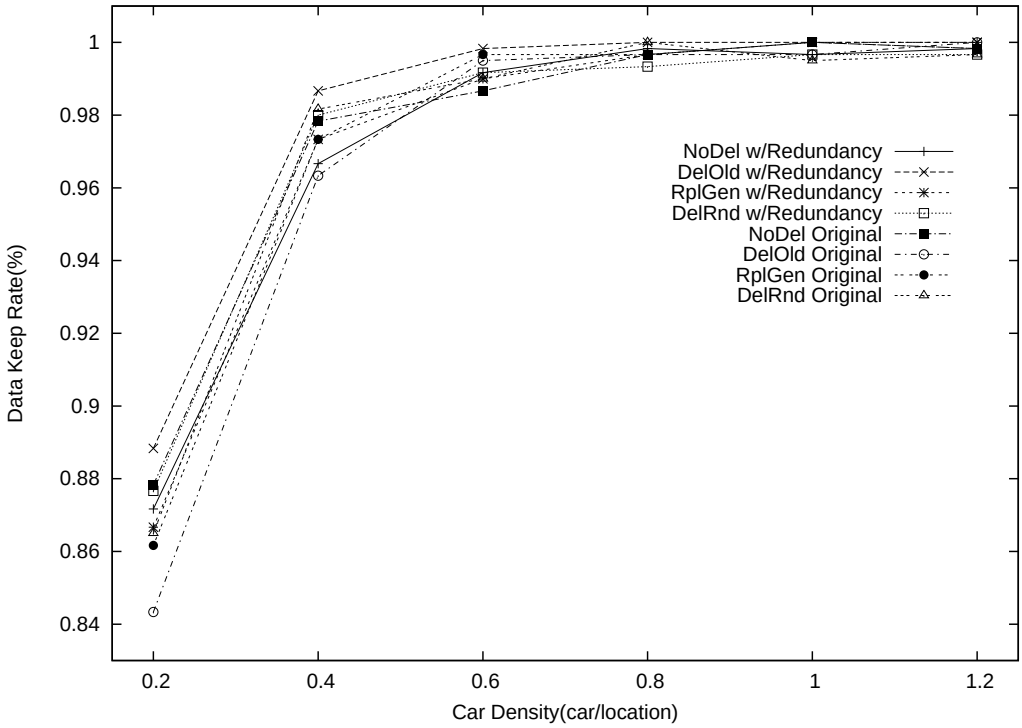

Figure 20: **Car density vs. data keep rate at storage size = 40**

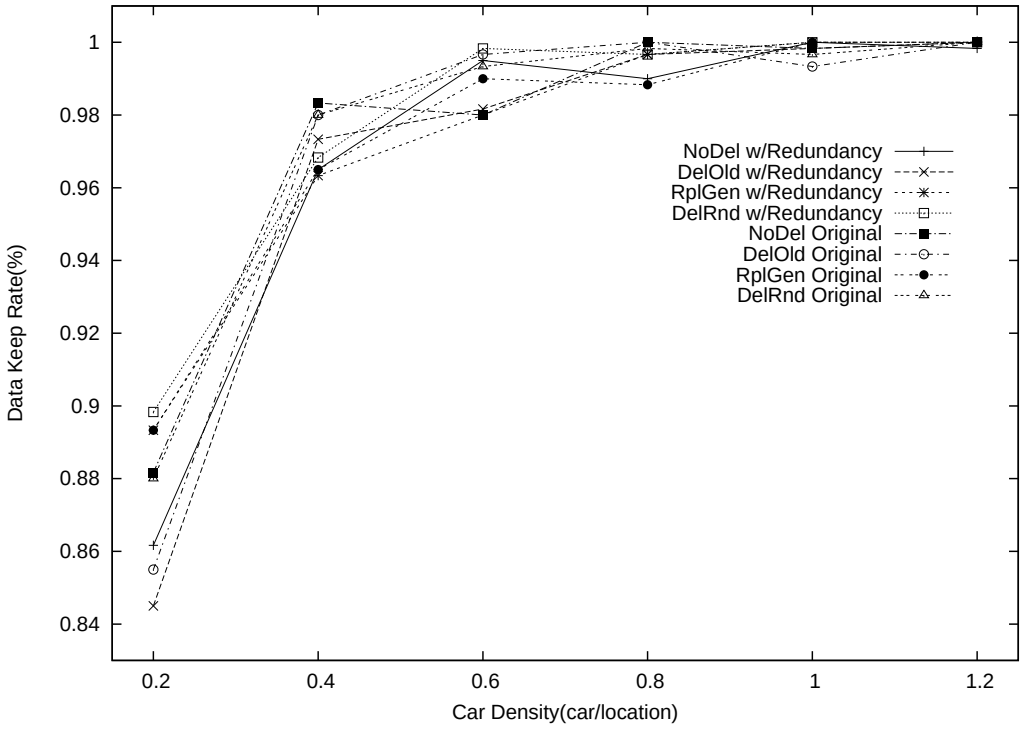

Figure 21: **Car density vs. data keep rate at storage size = 60**

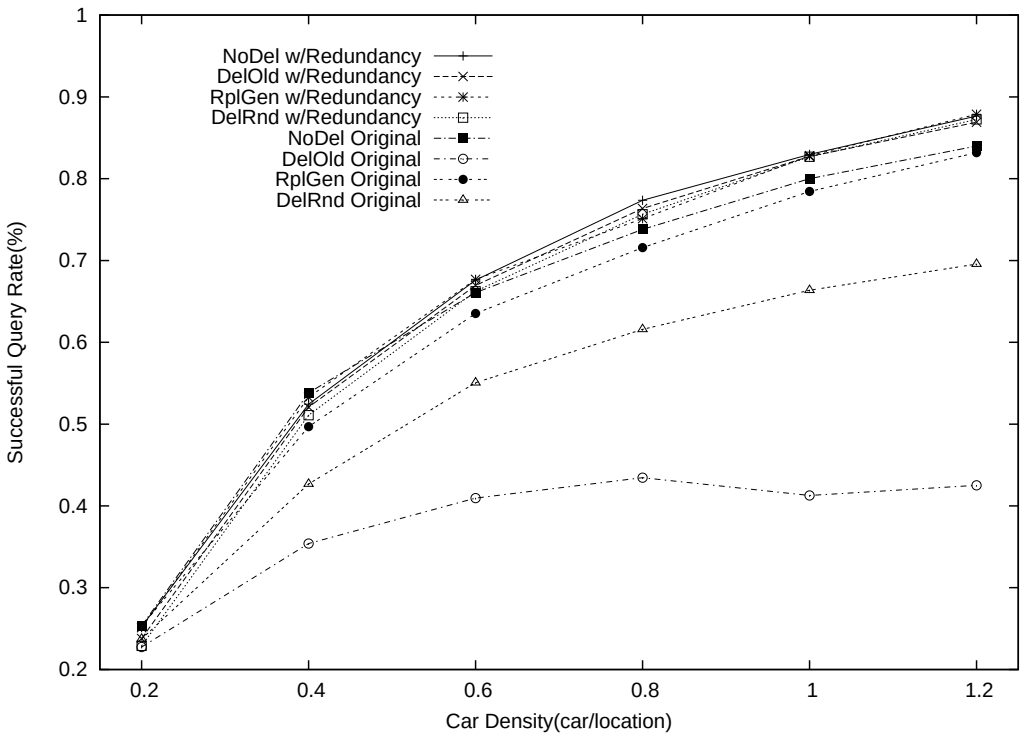

Figure 22: **Car density vs. successful query rate at storage size = 20**

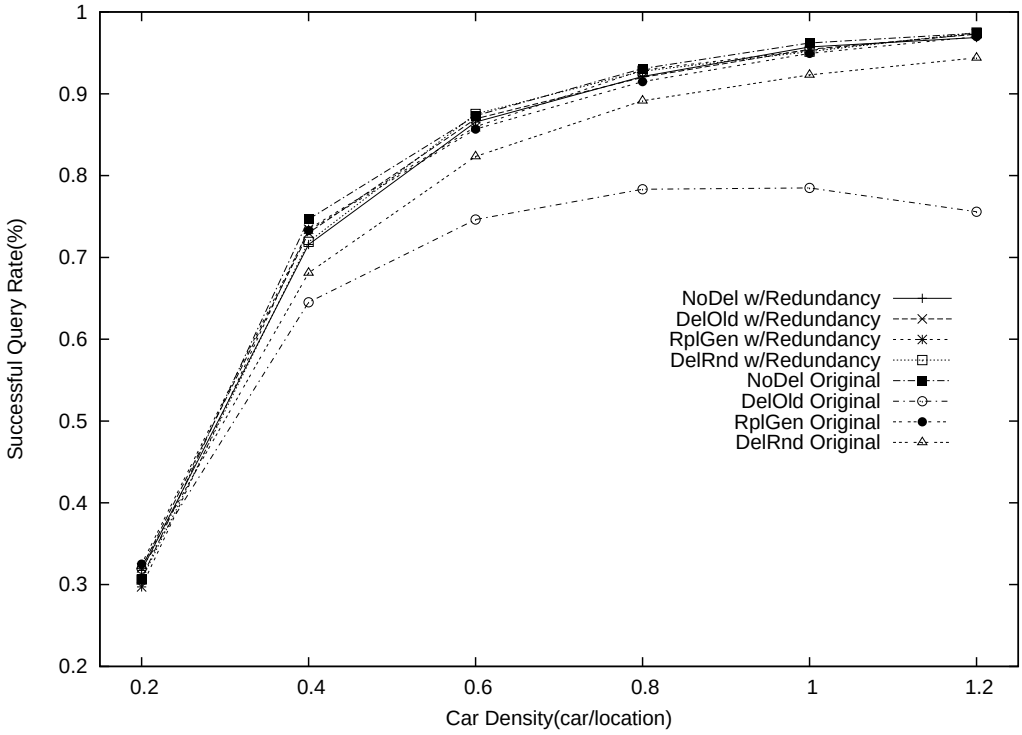

Figure 23: **Car density vs. successful query rate at storage size = 40**

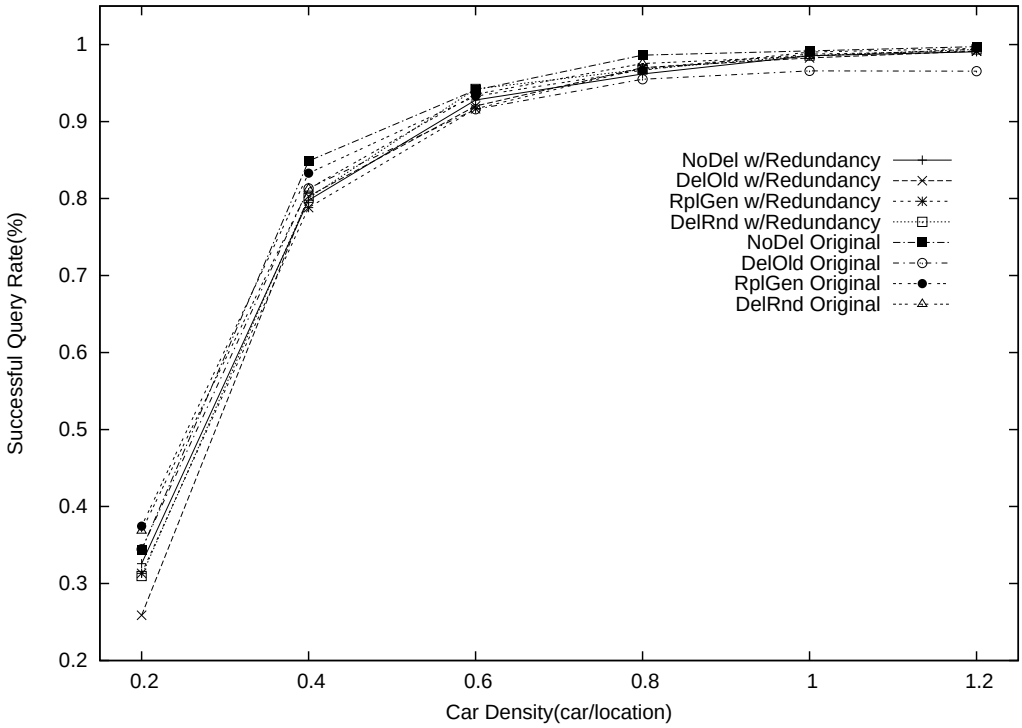

Figure 24: **Car density vs. successful query rate at storage size = 60**

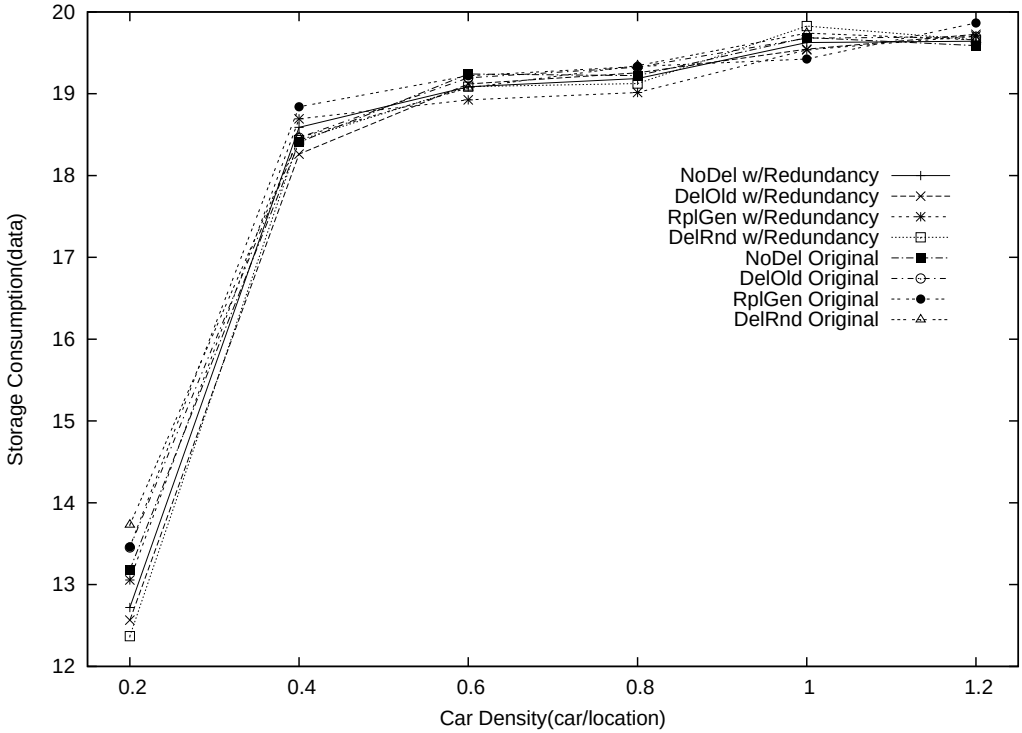

Figure 25: **Car density vs. storage consumption at storage size = 20**

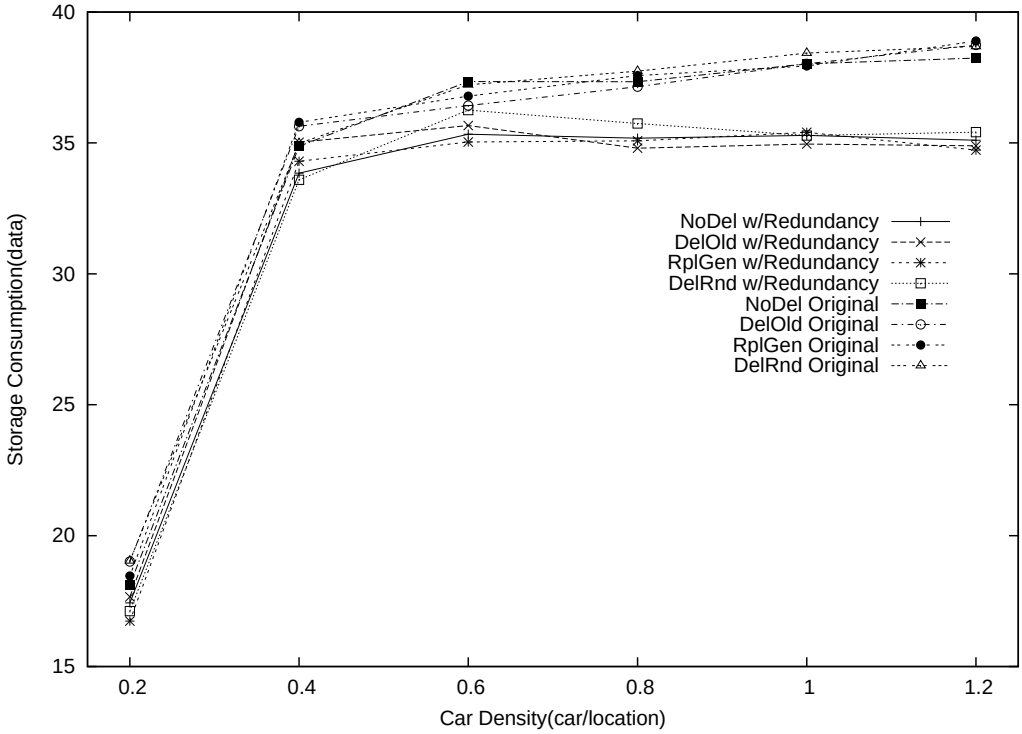

Figure 26: **Car density vs. storage consumption at storage size = 40**

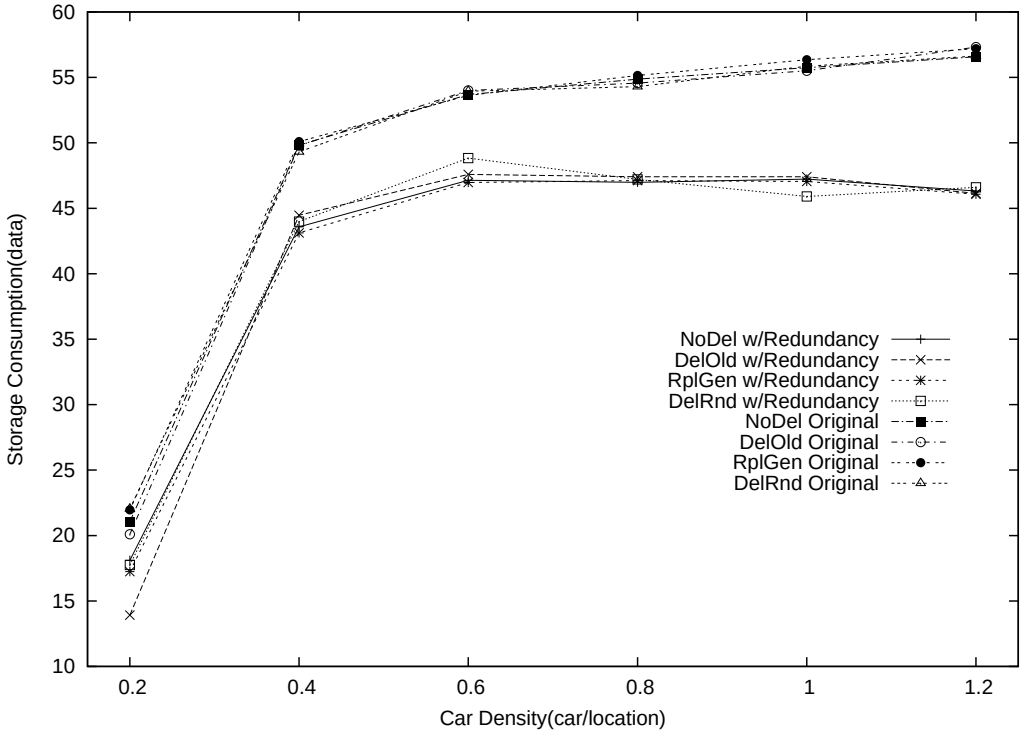

Figure 27: **Car density vs. storage consumption at storage size = 60**

w/Redundancy" plots show the SQR of ARw/D using Nodel, "DelOld w/Redundancy" plots show the SQR of ARw/D using DelOld, "RplGen w/Redundancy" plots show the SQR of ARw/D using RplGen, "DelTnd w/Redundancy" plots show the SQR of ARw/D using DelRnd, "NoDel Original" plots show the SQR of the data storage without distribution using Nodel, "DelOld Original" plots show the SQR of the data storage without distribution using DelOld, "RplGen Original" plots show the SQR of the data storage without distribution using RplGen, and "DelTnd Original" plots show the SQR of the data storage without distribution using DelRnd.

The graphs show that there is no significant difference among the storage management protocols when the storage size is large and that that the data storage without distribution using DelRnd and the data storage without distribution using DelOld do not work well when the storage size is small.

The reason why there is no difference when we use ARw/D is that the data item belonging to a lower priority group is dropped in most of the case so the storage management protocol does not use the properties of data item.

Using the DelOld, a car drops the data item which was stored earliest. And the neighbors of the car are likely to move together because a car is more likely to move in the same direction that it moved during the last interval. This means that most of the neighbors of the car are likely to have run out of SR count for that item. So, once the data item was deleted, there are few chances to receive the data again.

The bad result of the DelRnd shows that the storage management using the properties of data item is effective.

Figure 25-27 plot the impact of varying the car density on storage consumption. Figure 25 shows the result when the storage size of each peer is 20, Figure 26 shows the result when the storage size of each peer is 40, and Figure 27 shows the result when the storage size of each peer is 60. In these graphs, "NoDel w/Redundancy" plots show the storage consumption of ARw/D using Nodel, "DelOld w/Redundancy" plots show the storage consumption of ARw/D using DelOld, "RplGen w/Redundancy" plots show the storage consumption of ARw/D using RplGen, "DelTnd w/Redundancy" plots show the storage consumption of ARw/D using DelRnd, "NoDel Original" plots show the storage consumption of the data storage without distribution using Nodel, "DelOld Original" plots show the storage consumption of the data storage without distribution using DelOld, "RplGen Original" plots show the storage consumption of the data storage without distribution using RplGen, and "DelTnd Original" plots show the storage consumption of the data storage without distribution using DelRnd.

The graphs show that that the ARw/D consumes less storage than the data storage without distribution when the storage size is larger.

As shown above, when the storage size is small, the redundancy control does not work well because too many data are dropped to utilize priority. Even though it does not work well, the storage consumption is not larger than that of the data storage without distribution. When the storage size is large enough, the redundancy control can save the storage consumption.

As shown above, the redundancy control ARw/D can achieve high SQR with less storage consumption.

6 Future Work

In this section, we show future work relating to this study.

To realize no data loss more perfectly, we need an algorithm which considers the boundary of a data sharing area. For example, transmitting stored data when approaching the boundary may increase DKR.

We need the data management mechanism which works well at very low car density. We think we need new adaptation algorithm in data dissemination to solve this problem.

We would like to study the coordination of the data storage mechanism and the data dissemination mechanism. For example, if the suppression function utilize the information of data storage, data can be disseminated more effectively.

We would also like to do a detailed study on the retrieval phase, especially numerical analysis of the limited search and the wide search.

There is another possible method of query processing, in which a querier declare where and when it wants to receive the answer and data holders send the answer to the appointed location at the appointed time. We think that adapting the dissemination depending on the declaration realizes effective data management in such a case.

7 Conclusion

We propose a data management mechanism for peers to share data in a mobile P2P system. This mechanism consists of the data storing method, the query processing method, and the data dissemination method. We presented a data distribution algorithm, redundancy control algorithm, and storage management algorithm for data storing method and simulated data sharing using them. Our results show that the data distribution algorithm HID used with the redundancy control algorithm ARw/D performs significantly better than the data storage without distribution.

References

[1] M. Kan, R.Pande, P.Vinograd, and H. Garcia-Molina. Event Dissemination in High-Mobility Ad-hoc Networks. Submitted, 2005

[2] S. Mehrotra, C.T. Butts, D. Klashnikov and N. Venkatasubramanian et al. Project RESCUE: Challenges in Responding to the Unexpected. In IS&T/SPIE International Conference on Internet Imaging, 2004.

[3] A. Nandan, S. Das, B. Zhou, G. Pau, and M. Gerla. AdTorrent: Digital Billboards for Vehicular Networks. In Proceedings of Vehicle-to-Vehicle Communications Workshop, 2005

[4] K. Patel, S. Iyer, and K. Paul. RINGS: Lookup Service for Peer-to-Peer Systems in Mobile Ad-Hoc Networks. In IWDC, 2004

[5] K. Seada and Ahmed Helmy Rendezvous Regions: A Scalable Architecture for Service Location and Data-Centric Storage in Large-Scale Wireless Networks. In IEEE/ACM IPDPS 4th Int'l Workshop on Algorithms for WMAN, 2004