

# Query Distribution Estimation and Predictive Caching in Mobile Ad Hoc Networks

Sheetal Gupta  
University of Maryland,  
Baltimore County  
1000 Hilltop Circle  
Baltimore, Maryland 21250  
sheetal4@cs.umbc.edu

Anupam Joshi  
University of Maryland,  
Baltimore County  
1000 Hilltop Circle  
Baltimore, Maryland 21250  
joshi@cs.umbc.edu

Justin Santiago  
Agnik, LLC  
8840 Stanford Blvd. Suite  
1300  
Columbia, Maryland 21045  
jsantiago@agnik.com

Anand Patwardhan  
University of Maryland,  
Baltimore County  
1000 Hilltop Circle  
Baltimore, Maryland 21250  
anand2@cs.umbc.edu

## ABSTRACT

The problem of data management has been studied widely in the field of mobile ad-hoc networks and pervasive computing. The issue addressed is that finding the data required by a device depends on chance encounter with the source of data. Most existing research has focused on specifying the required data by specifying the user or application intentions. These approaches take the semantics of data into account while caching data onto mobile devices from the wired sources. We propose a scheme by which mobile devices proactively increase the availability of data by pushing and caching the most popular data in the network. It involves a local distributed technique for estimating global query distribution in the network. The devices have a finite sized cache to store the pushed data and use their estimation of queries for prioritizing the data to cache. We implement this technique in the network simulator, Glomosim and show that our scheme improves data availability as well as the response latency.

## Keywords

Mobile ad hoc networks, query distribution estimation, increase data availability.

## 1. INTRODUCTION

In mobile ad hoc networks and pervasive computing environments, there are multiple independent sources of data. Mobile devices can be both producers and consumers of data. Thus the data is often distributed in these data intensive environments. The mobile devices have constrained

storage capacity, computing power and energy resources. They must forage for the information they seek in their neighborhood using the constrained resources available to them. Obtaining the data they seek depends on serendipitous meeting with the source of data. A lot of work has been done to improve data availability in mobile and pervasive environments. The constrained resources impose a limit on the number of messages that the device can exchange to get the required information. The latency of obtaining the information is also a concern.

We focus on being able to get the needed data on demand as well as maximizing the utility of available cache space. We propose a scheme for estimating the query distribution in the network. This helps in predicting which data is most popular and likely to be queried for in future. We push frequently queried data into the network, thus spreading the data that is most wanted in the network. This scheme enhances the availability of reliable data in MANETs by collaborative data exchanges with other devices and by ascertaining the reliability of the aggregated information using a validation process. We aim to use the scarce cache space available with the mobile devices efficiently, by using it to cache the most popular data. We illustrate the applicability of our technique in the following two real-life scenarios.

Vehicles fitted with computing devices having wireless capabilities cache relevant data from sensors stationed by the freeways that broadcast local conditions. The vehicular devices establish network connections with the neighboring vehicular devices that are passing by and form a vehicular ad hoc network. Common well-known information useful to such devices, are emergency related (e.g., police, medical, and fire department), traffic and road condition related, weather related, and maintenance related (e.g., gas station, towing service etc.). The local update about a closed lane is received by a device and must be propagated to other devices in a timely fashion. This must be done using the limited bandwidth and cache space available to the devices. The devices traveling in the opposite direction must cache this update with a high probability, so that it is received by other devices with low latency.

Similar functionality is achieved by the commercial ser-

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 2008 ACM 978-1-60558-221-4 ...\$5.00.

vice, “Dash Express, a two-way, Internet-connected GPS navigation system” [1] for automobiles. It works by sending each Dash driver’s location and speed to the Dash servers. This real-time traffic information is propagated to other local Dash drivers through Internet connectivity. The device can now decide on which route to take, to minimize time required to reach the destination based on actual traffic speeds. However it does not harness the powers of establishing mobile ad hoc links with its neighboring devices.

Soldiers on the battlefield carrying mobile hand-held devices with wireless capabilities is another scenario where it is useful to cache information based on the likelihood of getting queried for it. This includes information about supplies, enemy strength, strategic planning etc. In such tactical environments a central trusted authority is lacking and connectivity is volatile. The predictive caching technique improves data availability, reliability and the answering delay for the devices.

The remainder of the paper is organized as follows. Section 2 contains related work in the area of data management in mobile ad hoc networks and pervasive computing. Section 3 describes basic system model on top of which our system is built. The proposed algorithm for query distribution estimation and predictive caching is also described in this section. Section 4 contains results from the simulations that demonstrate the usefulness of our technique. We then conclude in section 5 with ideas for future work.

## 2. RELATED WORK

The problem of query result size estimation has been studied in traditional database systems. In [8] the range query result size is estimated based on data distribution and is fine tuned using the user query pattern. However this work does not deal with estimating the frequency of the queries themselves.

The problem of data management has been extensively researched in mobile and pervasive computing environments. Cherniak *et al.* [4] introduced the concept of data recharging based on user profiles. Mobile devices deal with disconnection from networked data sources by caching or “recharging” of data. The user profile consists of domain and utility function. The domain specifies the data objects of interest to the user. The integer utility function specifies the relative value of the data objects within a profile domain. The utility function is used to cope with limited storage, bandwidth and recharge time by prioritizing the data items to be cached. The paper describes the desirable properties of the language used to specify this highly expressive user profile and mention the limitations of the existing languages for this purpose. It mentions that the language must have reasoning capabilities over the metadata properties. The Cherniak approach relies on connectivity with servers on wired networks for data recharging. However we take the approach of also having peer mobile devices as our sources of data. We use a reputation management scheme to ascertain the reliability of the data acquired from peers.

Perich *et al.* [6] took this concept one step further by proposing that both the domains of data that a user needs and its utility will change depending on the context the user is currently in. They believe that modeling a user profile in terms of “beliefs,” “desires,” and “intentions” of the user, is a comprehensive way of modeling and for anticipating the future data requirements of the user. The “intentions” of a

user, modulated by the “beliefs” as well as contextual information, including location, time, battery power, and storage space, allow the information manager of each device to determine what data to obtain and its relative worth. Profiles are encoded using ontologies - DAML, a semantically rich language. Every device maintains a subset of the global information repository that it can provide to itself and possibly to others.

Both these approaches require a semantic description of the user needs. This requires the user to anticipate all future data requirements. They also focus on caching the data that is useful to the device. Whereas in this paper, we propose that after satisfying the device’s own requirements, we aim to increase data availability for the peer devices. This is done by estimating the global data requirement and then caching data accordingly. Our focus is not on inferring future data requirements of the user. Queries are known to the nodes and do not change over time.

Yin and Cao [10] propose a cooperative caching scheme for mobile ad hoc networks with finite cache using an efficient cache replacement policy. Popular data is cached, or the path to the data is cached or a hybrid approach is taken. They use information from the underlying routing protocol to minimize delays due to long communication paths. However their system model assumes the existence of a few data server nodes. The data consumer nodes know the identity of these nodes and know the mapping between the data they need and the data source. The nodes request data from those nodes and during transit the data is cached by the intermediate nodes. Either the data itself is cached or the path, which is the final requester node identifier along with the data identifier is cached. In contrast, our system model consists of devices that communicate with their one-hop neighbors for acquiring data. They estimate the popularity of data using a formula proposed in [7]. They show that even if this estimate is not very accurate, their cache replacement policy is effective. They verify this by introducing noise in their estimate and observing that the answering delay is not significantly affected by the error noise. We think that this is because their cache replacement policy is also a function of the size of data. That factor alone is sufficient to make their cache replacement policy effective. They do not focus on accurately estimating query distribution in the network. They do not have the concept of pushing popular data in the network to increase its availability. Also all nodes in the network are trusted and no validation is performed on the data acquired from peers.

Xu and Wolfson [9] examine database management for spatio-temporal resource information in mobile peer-to-peer networks. The database is distributed among the moving objects and they serve as routers of queries and answers. To address limited communication time, nodes prioritize the resource “reports” available to them. They adopt a hierarchical weighting priority structure that is set by the user unlike our approach where priorities are determined by the system. The paper also explores the concept of virtual currency to create incentive for peer-to-peer cooperation. The feasibility and performance of the proposal is not backed by actual simulations. Budiarto *et al.* [3] compare replication strategies for mobile databases. Consistency is the primary issue addressed by the paper. In our scenario the data is not updated once it has been generated by the source. More up-to-date data can be generated as time passes, however

the old data does not change. Thus maintaining consistency in mobile databases is not the focus of this paper.

### 3. PROPOSED APPROACH

#### 3.1 Basic Model

In prior work [5], we proposed a reputation management scheme that is used to validate the data acquired from peer mobile devices using accuracy and majority agreement in the provided data. We focus on mobile ad hoc networks where a small fraction of the nodes in the network are initially trusted, and we determine the reputation of the other nodes. In a vehicular ad hoc network, these trusted nodes can be the anchored sensor nodes that periodically broadcast current local conditions. In a battle space scenario, they can be the mobile devices carried by the higher ranked officers. Our query estimation and predictive caching technique is built on top of this reputation management scheme. This scheme [5] enhances the availability of reliable data in the MANET by collaborative data exchanges with other devices and by ascertaining the reliability of the aggregated information using a validation process. Thus the MANET consists of static trusted devices and mobile devices whose reputation must be determined based on data exchanges. The mobile devices receive the local conditions information from the trusted devices and also serve to propagate them further. A data packet received by a mobile device from a trusted anchored device is immediately trusted. A data packet received from a peer mobile device is cached, and a validation session is created for it. The data is validated when the same data is received from a trusted source later that happens at the serendipitous occurrence of the mobile device passing close to the actual source of the data. A packet is also validated if a threshold value of minimum agreement in the data received from its peer mobile devices is reached. If the session has been active for some time without validation being achieved, the data is timed-out and removed from the device cache. This scheme provides timely and accurate data to the consumer devices in a MANET and forms the starting point for our tests.

#### 3.2 Query Distribution Estimation

We use a local distributed algorithm for estimating the global query distribution in the network. Each device periodically broadcasts its list of queries to all other devices in range. This is done irrespective of whether the query has been answered. The broadcast message also contains the information of whether the answer is known to the querying node. The receiving devices will send the answer to those questions if they know the answer and the querying node indicated that it does not know the answer. In addition to answering, the receiving devices will process the query list message to extract counts of each type of query that its neighbor has. Initially a device only knows about its own queries. But as it encounters more devices and exchanges query lists with them, it knows more about the distribution of queries in its neighborhood. Gradually this knowledge of query distribution converges to that of the global query distribution in the network. This knowledge helps the nodes in predicting which queries it is likely to encounter in future.

Let  $c_i$  denote the count of queries of type  $i$  seen by a device so far. Let  $T_q$  denote the total count of all types of queries seen so far. Then the query frequency  $f_i$  of query type  $i$  is

calculated using the formula:

$$f_i = c_i/T_q$$

Note that the formula is entirely based on local cached data of a node. It does not assume any global knowledge.

Given the simple formula used, the computation overhead imposed by this algorithm is not significant in terms of energy cost as compared to the transmission costs. We follow an abstract query model, where each query is represented by a unique query identifier number. Thus processing the query list message to extract counts of each query type, consists of looking at the query identifier number and incrementing the corresponding query count. Answers are matched to queries by matching the answer identifier number to the query identifier number. The energy consumption is thus dominated by transmission energy required by the query estimation algorithm and pushing of data in the network.

#### 3.3 Predictive Caching

The knowledge about global query distribution in the network is utilized by the mobile devices to push data corresponding to the queries that have a query frequency greater than a user defined threshold frequency, into the network. This is done by broadcasting the data packets that have been validated, either by receiving the data directly from the source node or by attaining a majority agreement in the data obtained from peers and has subsequently been cached by the mobile node. Only the data packets that satisfy the condition of having a requirement in the network above the threshold query frequency are broadcast by the mobile nodes. This pushing of popular data in the network facilitates the dissemination of required data and increases the availability of data in the network.

We experimented with different cache replacement policies for caching of the pushed data. Firstly the data that corresponds to the queries that the node has is cached. Other data they encounter is due to the seeding of current data in the network by anchor sensor devices and as a result of pushing of popular data by the mobile nodes. This data is either cached or discarded based on the cache replacement policy. We studied the performance of the algorithm with FIFO and priority-based cache replacement policy. With FIFO the mobile nodes cache the latest data that has been pushed towards them. With priority-based cache replacement scheme, the priority of data is determined by the corresponding query frequency calculated for that data by the node. Thus the data that is predicted to be queried for the most, is cached with a higher priority and the data that is least queried is removed from the cache when more popular data is pushed towards it.

We compare the performance of these caching schemes with the performance using a simple cache for the nodes wherein they only cache the data corresponding to the queries that they have. Thus there is no predictive caching of the data in the hope that they will be asked for in future. We also studied the comparison of the performance of these caching schemes with the performance of the system when nodes have an infinite sized cache.

We also experimented with the system by modifying it so that there is no pushing of data in the network based on the query estimation. This is to determine the effect of pushing on the network performance. In this scenario the nodes still perform query estimation and use priority-based caching to

**Table 1: Simulation Parameters**

Spatial Dimensions	700 m x 900 m
Simulation period	30 min
Mobiles devices	50,100,150,200
Stationary devices	38
Transmission range	99.472 m
Routing Protocol	AODV
Mobility pattern	Vehicular trace

cache the data that they encounter. However they do not use the query distribution estimate to broadcast popular data in the network, thus being less effective in increasing the availability of data.

The baseline for performance comparison is the case where the nodes have a simple cache and do not push data. This represents a typical mobile ad hoc network environment where answers are obtained only on serendipitous encounters with the data source. The results are presented in the simulation section.

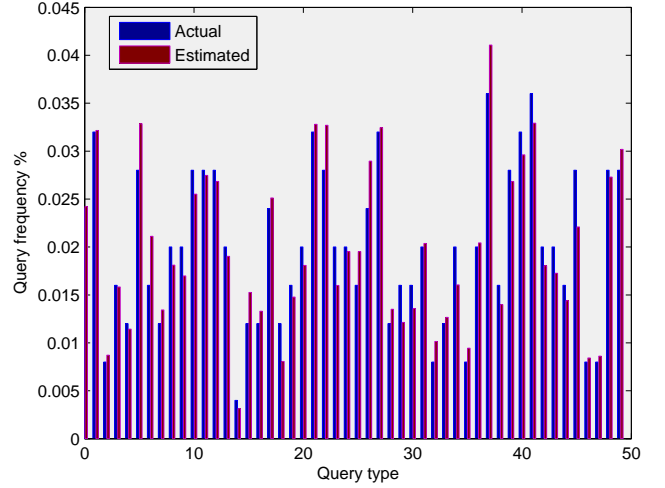
#### 4. SIMULATION

We implemented the query estimation and predictive caching algorithm using the Glomosim [11] simulator. The information in this section was generated using the simulation parameters mentioned in Table 1.

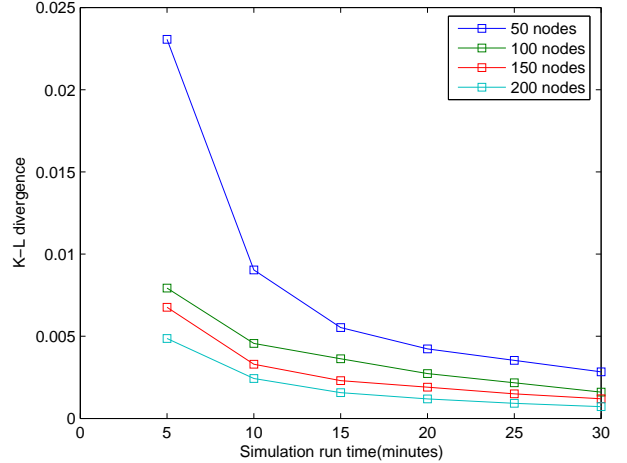
The experiment was run with 50, 100, 150 and 200 mobile nodes and 38 "pre-trusted" anchor nodes, for a duration of 30 minutes. A mobility pattern of vehicular movement was chosen, with speeds ranging from 15 m/s to 25 m/s and pause times of 0 to 30 s. Most existing work use random waypoint motion in their simulations. We chose to use vehicular movement since it is more representative of real-life scenarios. We modeled the actual road network around the Dupont Circle area in Washington DC as described in [5]. Each anchor node has a list of 5 answers that it will seed into the network. Every 2 minutes one of the 5 answers at each node is chosen and broadcast to all mobile nodes within range. Each mobile node has a set of 5 queries that it wishes to get answered and broadcasts to all nodes in range after every 1 minute. The mobile nodes are assigned queries and anchor nodes are assigned answers in a random uniform distribution pattern. The exchange of query list is used in building an estimate of the global query distribution. The mobile nodes that receive the query will send the answer to the querying node if they have it. The mobile nodes will also send out cached, validated data packets they have once every minute, in accordance with the minimum query threshold frequency that is set up at the beginning of the simulation. The mobile devices have a cache size of 10, so that they can cache 5 answers in addition to the answers that they need. There are 50 unique queries and answers in the network. The experiment was run 5 different times using a different vehicular trace path for each node in the simulation every time it was run.

Figure 1 shows a plot of the average estimated query distribution by the mobile nodes after a simulation run time of 5 minutes and the actual query distribution. We see that the estimated distribution has almost converged with the actual distribution.

More formally we measure the Kullback-Leibler divergence [2] or relative entropy between the two actual and estimated



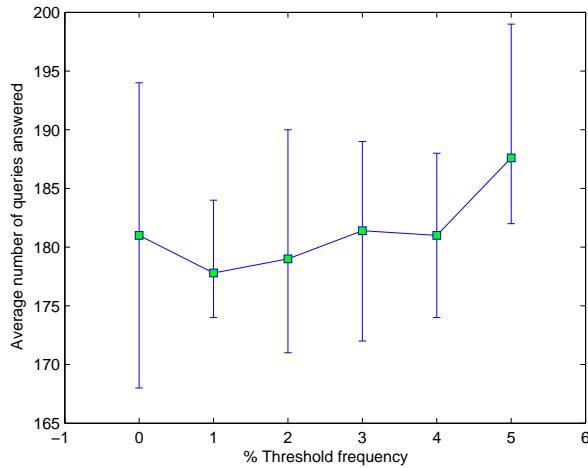
**Figure 1: Query distribution plot of the actual and estimated distributions after 5 minutes of simulation run time**



**Figure 2: Kullback-Leibler divergence between the average estimated and actual query distributions**

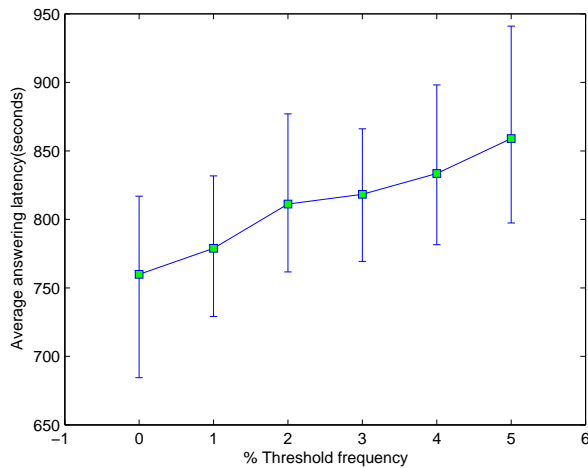
query distributions. We see in figure 2 that the divergence reaches a low value of 0.023 for a network size of 50 nodes after only 5 minutes of simulation run time from a value of infinity at time  $t=0$ . We also observe that the convergence of the query estimation algorithm is faster in case of a denser network. So the convergence is sooner for a network size of 200 nodes than that for a network size of 50 nodes.

Figure 3 shows the results for a simulation run with a network size of 50 nodes and with a priority based caching scheme, where priority is determined by the query estimation algorithm. The average number of queries answered is shown along with the variations. The y-axis scale is expanded to see the exact nature of the graph. A threshold frequency of 0% implies that all cached data is pushed in



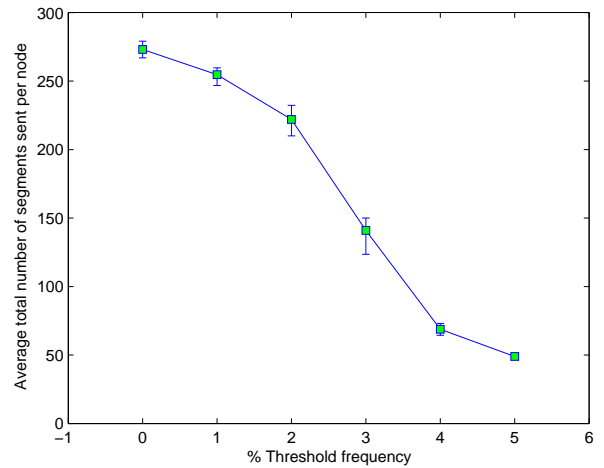
**Figure 3: Variation of number of queries answered as the threshold frequency is varied from 0-5%**

the network. Thus the query estimation is not being used to push the required data. It is only being used for cache replacement decisions. We observe that at threshold frequencies 0%,1%,2%,3% and 4% the total number of queries answered remains about the same. At a low threshold frequency of say 1%, the nodes do not distinguish between data that has a query frequency of 1% and data with a query frequency of 4% while pushing it. The receiving nodes filter out the data by only caching the 4% frequency data and discarding the 1% frequency data. Thus the effect of pushing is canceled by filtering by cache. Hence the number of queries answered remains the about the same at threshold frequencies of 0%,1%,2%,3% and 4%. The number of queries answered seems to increase slightly with increase in the threshold frequency to 5%.



**Figure 4: Variation of average query answering delay as the threshold frequency is varied from 0-5%**

The average time to answer a query clearly increases as the percentage threshold frequency increases as seen in figure 4. This is because since at high threshold frequencies nodes only push the most accessed data, proper dissemination of data does not take place. Thus the available node cache space is not utilized completely in case of high threshold frequency. On the other hand, at low threshold frequencies, the nodes do a better job of disseminating required data in the network resulting in increased availability. Thus answering delay is least in case of a threshold frequency of 0% and worst in case of a threshold frequency of 5%.

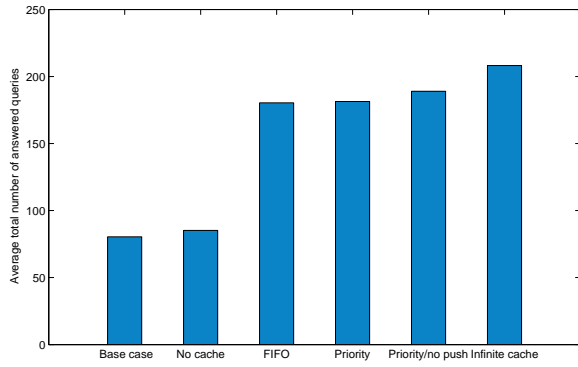


**Figure 5: Variation of average total segments sent per node as the threshold frequency is varied from 0-5%**

Figure 5 shows that the number of segments or answers transmitted per node decreases significantly with increase in threshold frequency. This is because as the threshold frequency increases, fewer cached data satisfy the condition of having a frequency above the threshold frequency and hence do not qualify for being pushed in the network. This decreases the overall traffic in the network at higher threshold frequencies. With decrease in number of transmissions per node, the energy consumption per node is also minimized.

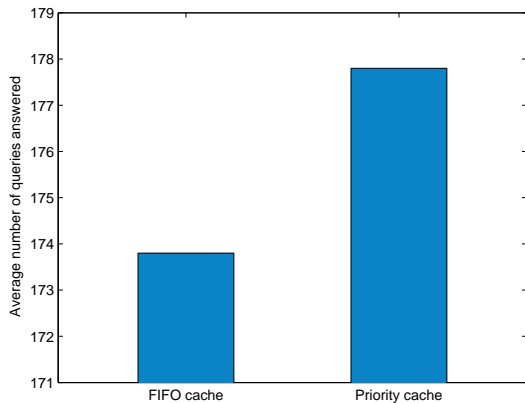
From the graphs we see that a threshold frequency of 5% works best, in terms of number of queries answered and energy consumption per node if the answering delay is not a concern. Selecting a threshold frequency midway like 3% seems to be a good compromise if answering delay is also an important factor.

Figure 6 characterizes the behavior of the query estimation and caching algorithm for different caching schemes for a threshold frequency of 3% and a network size of 50 nodes. In the no cache case nodes only have space to store answers to their own questions. As expected the total number of queries answered is the largest in the case of infinite cache and least in the base case. The base case consists of no cache and no pushing of data. In the infinite cache case, nodes have unlimited cache space to store all the answers that were pushed to it. This increases the data availability in the network. Conversely in the base and no cache case, nodes do not cache any data that is pushed to it in the net-



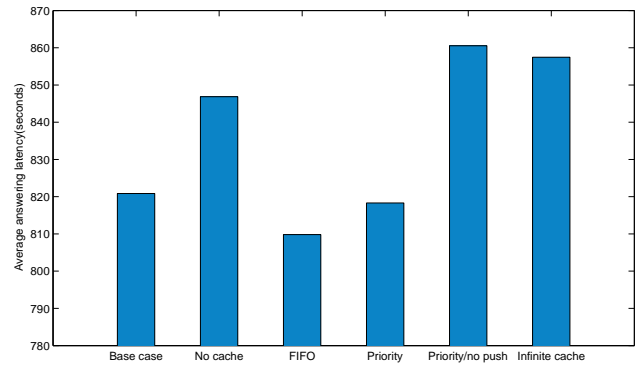
**Figure 6: Average total number of queries answered for different caching schemes for a threshold frequency of 3%**

work. This results in decreased total number of answered queries. The number of answered queries increases slightly for the priority-based caching scheme as compared to that for FIFO caching scheme. This can be attributed to the fact that at 3% threshold frequency, most of the data that is pushed in the network is highly popular data and thus even FIFO caching results in higher availability of required data and thus good performance. In the case of no pushing of answers in the network by the mobile nodes based on query estimation, the only means of data dissemination are the data broadcast by the source anchor nodes. The nodes use a priority-based caching scheme based on its query estimation. Here the only way of getting a query answered is by a chance encounter with the source anchor node having the answer. It is observed that this case performs as good as the FIFO cache and priority-based caching schemes in terms of total number of queries answered. However it performs badly in terms of query answering delay as seen in figure 8.



**Figure 7: Average total number of queries answered for priority and FIFO caching schemes for a threshold frequency of 1%**

Figure 7 shows that the priority-based cache replacement scheme results in greater number of queries getting answered than using a FIFO cache for a threshold frequency of 1%. This is because although a lot of the data qualifies for pushing with a low threshold frequency of 1%, the priority-based caching scheme is better able to cache only the most popular data than the FIFO caching scheme which discards old data, even if it was estimated to be more popular than the new data. We also observed that the answering delay was 11s less in case of priority caching scheme than for FIFO cache. However the number of segments/answers sent per node was greater in case of priority caching scheme than FIFO caching scheme by 18 segments since most of the highly popular data in its cache qualified for pushing and so greater number of answers were pushed in the network.



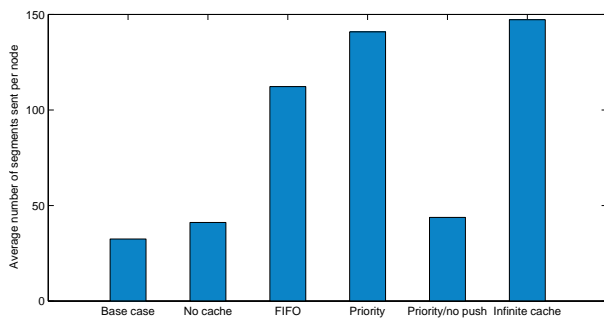
**Figure 8: Average query answering delay for different caching schemes for a threshold frequency of 3%**

As seen in figure 8 the average time to answer a query is the worst in the case of no pushing of answers in the network because of its dependence on chance encounter with the source nodes to have its query answered. The no cache case also performs poorly as expected since there is no cache in the mobile nodes for storage of the pushed data in the network. The performances of the FIFO and priority cache schemes are about the same due to the same reason that the data dissemination is almost the same for a threshold frequency of 3%.

Figure 9 shows that the number of segments/answers transmitted per node is least in the base case, no pushing of answers and in the case where nodes do not have any extra cache space. The number of segments sent per node is slightly more in priority cache case than FIFO cache case, since the priority cache will lead to more in-cache data to qualify for pushing in the network. The number of segments sent is the highest in infinite cache case, since it can cache more data that qualifies for pushing in the network. This graph indicates the traffic overhead caused due the query estimation algorithm and pushing of data in the network.

## 5. CONCLUSION AND FUTURE WORK

Data management in mobile ad hoc networks is an important issue, with each mobile device carrying only a fraction of the data. Instead of relying on chance encounter to get the



**Figure 9: Average total number of segments sent per node for different caching schemes for a threshold frequency of 3%**

answers to its queries, we proposed that the mobile devices dynamically estimate the global query distribution. They use this estimate of global query distribution to predict and cache the most popular data in the hope of being able to provide it to other devices when asked by them. This increases the data availability in the network and decreases latency of obtaining required data.

From our preliminary simulation results we observed that the threshold frequency should be set low enough for data dissemination to take place and high enough to limit the traffic and thus energy consumption in the network. Using this query estimation technique, we are able to make better utilization of available cache space to increase the data availability in the network.

For future work we are investigating how to assign a confidence level to each answer obtained by majority agreement in a mobile ad hoc network. We will use statistical techniques to achieve this. We will improve on [5] by taking into account the reputations of the source devices while obtaining majority agreement. If an application knows the certainty it desires, and its trust in the neighbors, it can decide what number of neighbors must agree on the information to achieve the given certainty bound. We will use Chernoff's bound to determine the lower bound on the number of neighbors that must agree on the information.

## 6. ACKNOWLEDGMENTS

This project has been supported by US Army Contract W15P7T-07-C-P447. The authors would like to thank Dr. Hillol Kargupta, UMBC for his help during discussions.

## 7. REFERENCES

- [1] <http://dash.net>.
- [2] <http://mathworld.wolfram.com/RelativeEntropy.html>.
- [3] Budiarto, S. Nishio, and M. Tsukamoto. Data management issues in mobile and peer-to-peer environments. *Data Knowl. Eng.*, 41(2-3):183 – 204, June 2002.
- [4] M. Cherniack, M. J. Franklin, and S. Zdonik. Expressing user profiles for data recharging. *Personal Communications, IEEE*, 8(4):32–38, August 2001.
- [5] A. Patwardhan, A. Joshi, T. Finin, and Y. Yesha. A data intensive reputation management scheme for vehicular ad hoc networks. *3rd Annual International Conference on Mobile and Ubiquitous Systems - Workshops*, pages 1–8, July 2006.
- [6] F. Perich, S. Avancha, D. Chakraborty, A. Joshi, and Y. Yesha. Profile driven data management for pervasive environments. *Proceedings of the 13th International Conference on Database and Expert Systems Applications*, pages 361 – 370, September 2002.
- [7] J. Shim, P. Scheuermann, and R. Vingralek. Proxy cache algorithms: Design, implementation, and performance. *IEEE Trans. Knowledge and Data Eng.*, 11(4):549–562, July/August 1999.
- [8] Y.-L. Wu, D. Agrawal, and A. E. Abbadi. Query estimation by adaptive sampling. *Proceedings of the ICDE Conference*, pages 639–648, February 2002.
- [9] B. Xu and O. Wolfson. Data management in mobile peer-to-peer networks. *2nd International Workshop on Databases, Information Systems, and Peer-to-Peer Computing (DBISP2P'04)*, August 2004.
- [10] L. Yin and G. Cao. Supporting cooperative caching in ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(1):77–89, January 2006.
- [11] X. Zeng, R. Bagrodia, and M. Gerla. Glomosim: A library for parallel simulation of large-scale wireless networks. *Workshop on Parallel and Distributed Simulation*, pages 154–161, July/August 1998.