APPROVAL SHEET

Title of Thesis: Trust-based Data Management in Mobile Ad Hoc Networks

Name of Candidate: Sheetal Gupta
Master of Science, 2008

Thesis and Abstract Approved: ________________________
Anupam Joshi
Professor
Department of Computer Science and
Electrical Engineering

Date Approved: ______________________________

Curriculum Vitae

Name: Sheetal Gupta

Permanent Address: 4760 Drayton Green, Halethorpe, MD 21227

Degree and date to be conferred: Master of Science, August 2008

Date of Birth: September 18, 1982

Place of Birth: Pune, India

Secondary Education: Vidya Niketan, Pimpri, Maharashtra, India, 1998

Collegiate institutions attended:

University of Maryland Baltimore County, Master of Science, Computer Science, 2008

University of Pune, India, Bachelor of Engineering(Computer Engineering), 2004

Major: Computer Science

Professional publications:

Sheetal Gupta, Anupam Joshi, Justin Santiago and Anand Patwardhan, ”Query Distribution Estimation and Predictive Caching in Mobile Ad Hoc Networks” in Proceedings of Seventh International ACM Workshop on Data Engineering for Wireless and Mobile Access (MobiDE’08), June 2008.

Mohamed Younis, Sookyoung Lee, Sheetal Gupta and Kevin Fisher, ”A Localized Self-healing Algorithm for Networks of Moveable Sensor Nodes” in Proceedings of IEEE Global Communications Conference (IEEE GLOBECOM 2008), December 2008.

Professional positions held:

Graduate Research Assistant, University of Maryland Baltimore County (Jan’08 - June’08).

Graduate Teaching Assistant, University of Maryland Baltimore County (Aug’07 - Dec’08).

Summer Intern, Cougaar Software, Inc (June’07 - Aug’07).

Graduate Teaching Assistant, University of Maryland Baltimore County (Aug’06 - May’07).

Software Engineer, Sybase Software, India Pvt. Ltd (June’04 - July’06).

Project Intern, Cradle Technologies, Pune, India (July’03 - March’04).

ABSTRACT

Title of Thesis: Trust-based Data Management in Mobile Ad Hoc Networks

Sheetal Gupta, Master of Science, 2008

Thesis directed by: Dr. Anupam Joshi, Professor
Department of Computer Science and
Electrical Engineering

The problem of data management has been studied widely in the field of mobile ad-hoc networks and pervasive computing. An issue is that finding the required data depends on chance encounter with the source of data. Most existing research take the semantics of data into account while caching data onto mobile devices from the wired sources. In this work, we propose that the mobile devices decide what to cache based on the queries they encounter. The scheme involves a distributed technique for estimating the global query distribution in the network. The devices proactively increase the availability of popular data in the network. They use their estimate of query frequencies to push popular data and to guide their caching decisions. We also address the issue of data tampering in MANETs. The answers obtained from peer devices may not be reliable. This propagation of incorrect data may be either intentional or out of ignorance. We propose a Bayesian approach to infer the correct answer. The suggested answers and the reputation values of the sources themselves are used to determine the most likely answer. We implement these techniques in the network simulator, Glomosim and show that our scheme improves data availability, response latency and data accuracy.

Trust-based Data Management in Mobile Ad Hoc Networks

by

Sheetal Gupta

Thesis submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
Master of Science
2008

To my beloved parents and doting brother

ACKNOWLEDGMENTS

I am very grateful to Dr. Anupam Joshi for his constant support, encouragement and timely advice. I am grateful to him to give me the opportunity to present my work at MobiDE'08. I would also like to thank Dr. Anand Patwardhan for his valuable comments and suggestions. I am grateful to Dr. Suresh Purini for his insightful thoughts and ideas. I thank Dr. Yun Peng for his inputs. Finally I would like to thank all my labmates, family and friends for their encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 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.

The data provided by peer mobile devices may not be reliable. This could be due to the presence of malicious devices in the network or simply due to their ignorance. Such peer-provided data cannot benefit from the security mechanisms available in a client-server environment. We devise a new technique to infer the most accurate data from the different versions of the same data provided by peers. Our approach is based on Bayes theorem. We associate a reputation value with each neighbor, which denotes the likelihood of a device to provide correct data. The reputation value is an indicator of both the trustworthiness and capability of the device. We choose the data which is most likely to be correct using the versions of the provided data along with the reputations of their sources. This technique aims to reduce the risk of propagating incorrect data in the network. Our focus is not on how the reputation values evolve. Rather it builds on top of a reputation evolving mechanism like in (Patwardhan *et al.* 2006) and (Perich *et al.* 2004).

We illustrate the applicability of our technique in two real-life scenarios. Vehicles fitted with wireless computing devices 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. Malicious devices may tamper with the data, providing false alarms or dropping genuine alarms.

Similar functionality is achieved by the commercial service, “Dash Express, a two-way, Internet-connected GPS navigation system” (Dash 2008) 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 battlefront 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 and answering delay for the devices. The approach to arrive at the correct data using Bayes theorem ensures credibility of data.

Chapter 2

RELATED WORK

2.1 Query Distribution Estimation

The problem of query result size estimation has been studied in traditional database systems. In (Wu, Agrawal, & Abbadi 2002) 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.* (Cherniack, Franklin, & Zdonik 2001) 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 trust-based data validation scheme to ascertain the reliability of data acquired from peers.

Perich *et al.* (Perich *et al.* 2002) 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.

Yin and Cao (Yin & Cao 2006) 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. Yin *et al.* estimate the popularity of data using a formula proposed in (Shim, Scheuermann, & Vingralek 1999). 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 (Xu & Wolfson 2004) 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.* (Budiarto, Nishio, & Tsukamoto 2002) 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. Thus the data cached my mobile devices continues to remains valid long after it was obtained from the source sensor devices. Maintaining consistency in mobile databases is not the focus of this paper.

Acharya *et al.* (Acharya *et al.* 1995) proposed the concept of ”broadcast disks”. It is applicable in a client-server environment, where the downstream communication dominates upstream communication. Instead of the server transferring data on request from client viz. ”pull” model, the server ”pushes” data out to the clients. The broadcast channel becomes a disk that clients can read from. The server broadcasts different items with differing frequency emphasizing the popular items. The broadcast program and the caching policy are considered together. Clients replace the page having the lowest ratio of access probability to broadcast frequency, rather than just replacing the least accessed page. We are considering a peer-to-peer network with no classification of devices into server roles. Each node acts as a server for the data it has and pushes only the data it thinks is popular. Contrary to their approach, the less popular data is not pushed at all. The data being pushed and the caching decisions made by the clients are based on the same factor viz. the global query distribution. All the data that a node needs fits into its cache. In addition to its own data requirements, it caches data likely to be asked for by it peers. The cache replacement policy affects the caching decisions for this additional cached data.

2.2 Reputation Management

Jonker *et al.* (Jonker & Treur 1999) propose a formal framework for trust evolution. They propose a mathematical model for trust management in multi-agent systems. The trust function is based on intial trust, experiences and trust dynamics. Perich *et al.* (Perich *et al.* 2004) propose a distributed, mathematical model for trust and belief management in mobile ad hoc networks. The model categorizes devices as reliable and unreliable. Several trust learning functions are described based on experience and recommendations from peers. The devices perform information source discovery and combine the suggested answer accuracy degree and reputation of sources to decide on the final answer. The devices accept the answer whose accuracy level is above a threshold value and is the highest among the received answers. Simple ways of combining accuracies of different versions of an answer are used like taking the maximum, minimum and average accuracy. Whereas we use a Bayes theorem based approach to find the correct answer. Our focus is not on how trust relations evolve, rather our approach works on top of a trust evolving mechanism. Moreover the model proposed by Perich *et al.* often concluded on the incorrect answer in highly dishonest environments. Our approach works reliably even in completely dishonest environments.

An approach is described by Patwardhan *et al.* (Patwardhan *et al.* 2006) in which a few nodes are trusted a priori and data is validated either using agreement among peers or direct communication with a trusted node. Collaborative propagation of reliable data helps in improving the timeliness of data. Bad nodes are detected when the data they provide is invalidated by the validation algorithm. Consensus is achieved when the number of copies agreeing is greater than a threshold value. The reputations of the devices are not considered when determining the consensus answer. We use this approach as a base line to compare the performance of our validation algorithm.

Chapter 3

QUERY ESTIMATION AND PREDICTIVE CACHING

3.1 Basic Model

In prior work (Patwardhan *et al.* 2006), Patwardhan *et al.* proposed a reputation management scheme that is used to validate the data acquired from peer mobile devices using majority agreement in the received data. They focus on mobile ad hoc networks where a small fraction of the nodes in the network are initially trusted, and the goal is to 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 (Patwardhan *et al.* 2006) 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 which 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 session is timed-out and corresponding data removed from the device cache. This scheme provides timely and reliable data to the consumer devices in a MANET and forms the starting point for our tests in query estimation and predictive caching algorithm.

3.2 Query Distribution Estimation

We use a 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 already 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 counts 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 consumed as compared to the energy loss due to transmission of messages. 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 costs 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 into the network. Cached data on a device qualifies for pushing if it is the answer to a query having a query frequency greater than the user defined threshold frequency. 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. The 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 it 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 cache the data that they encounter. However they do not use the query frequency 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.

3.4 Query Containment

Query containment is the problem of checking whether for every database, the result of one query is a subset of the result of another query (Libkin & Zhang 2008). It is useful for query optimization and materialized view selection in relational database systems and XPath queries over XML documents. Conjunctive query correspond to the select-from-where queries from SQL that only use and as a Boolean connective. The problem of conjunctive query containment is known to be NP-complete, but is polynomial time solvable for some special class of queries. The XPath query containment problem is easier in some sense than the classical relational conjunctive query containment (Schwentick 2008).

In our scenario if a device is asked a query that is contained in another query that it knows the answer to, the device can also answer the contained query. We do not focus on how query containment will be determined, but rather explore and try to quantify the effect that query containment will have on our algorithm performance.

Chapter 4

BAYESIAN APPROACH TO DATA VALIDATION

The data provided by peer mobile devices in a MANET may not be reliable. For example, we might receive differing versions of the same data from different peer devices. This could be either intentional or out of ignorance of the peers. It is thus necessary to validate data before accepting it as true and pushing it to other devices. We wish to infer the most accurate data from the different versions provided by our peers.

A simple approach would be to accept the answer from the highest reputation node and reject the other answers that were received from lower reputation nodes. However, this has the disadvantage of excessive reliance on the reputation values. For example if we received answer 0 from a node with reputation 0.9 and the answer 1 from three other neighbors with reputations only slightly less, 0.83, 0.85 and 0.87. In this case, it is intuitive to believe the answer provided by three highly reputed nodes rather than that provided by a single highly reputed node. That node may have been assigned a high reputation based on past experiences with it. But it may have now turned malicious or out of ignorance might be providing the incorrect answer.

Another approach is to find the answer based on majority agreement. An answer is accepted after the number of nodes that agree on an answer, become greater than a threshold value. For the example described above, this algorithm with a threshold value of 3 would choose the answer to be equal to 1 which intuitively seems the most reliable answer. However in a low trust environment, multiple low trust nodes can collude and each can provide the same wrong answer. This can cause the node to incorrectly assume the provided answer as the correct answer. This approach completely ignores the reputation values, leading to compromising on data reliability in low trust scenarios.

We propose an approach that takes into account the reputations as well as consensus to decide the most accurate answer. Simulation results show that the Bayesian approach for data validation presented here performs very well in terms of accuracy of answers in low trust scenarios. However this approach requires us to make certain assumptions.

1. A node's reputation value indicates its correctness probability value. This is a reasonable assumption, since the reputation evolution mechanism assigns reputation values based on how correct the node has been in the past. A higher value indicates a node having a positive history. Such a node is more likely to provide the correct answer in future.

2. All the received answers are equiprobable to be the right answer. In real life scenarios, typically the initial probabilities are not known and this is a reasonable assumption to make.

3. The nodes do not collude with each other, viz. they answer independently. Having a collusion with multiple participants is difficult to achieve in practise and is thus a rare occurrence. Also, the nodes validate data before propagating it further. Hence they will not propagate incorrect data obtained from malicious peers, which can be mistaken as collusion. Collusion is handled by our validation algorithm if the participating nodes have negative histories and thus low reputations.

4. A node can choose the answer from a finite set of possible answers. This assumption

may not be true for some class of queries. For example, in case of queries that have a real number as the answer, a device can choose the answer from the real number line, which has infinite points. Thus the answer can have an infinite range size.

Our approach is based on the Bayes theorem and builds on top of a reputation evolving mechanism. The mechanism associates a reputation value with each encountered mobile device. The reputation value is an indicator of the trustworthiness and the capability of the device. We choose the data which is most likely to be correct using the versions of the provided data along with the reputations of their sources.

Let the reputation value of node n_i be denoted by r_i. Let the actual answer be A_A and A_i be the answer returned by node n_i. Then we assume the source node reputation indicates the probability that the received answer A_i is equal to the actual answer, which has value equal to the constant c.

$$\Pr[A_i = c | A_A = c] \quad = \quad r_i$$

The probability with which node i gives the incorrect answer is equal to $1 - r_i$. There are two cases in calculating the probability that node i gives the particular incorrect answer A_i.

If the answer can take only binary values viz. either 0 or 1, then the probability that a node lies is given by $1 - r_i$. Thus the probability that node i gives answer A_i, where A_i is not the correct answer is equal to $1 - r_i$.

$$\Pr[A_i \neq c | A_A = c] \quad = \quad (1 - r_i)$$

The second case makes a closed world assumption for input, so that the node can choose the answer from a set of k distinct answers. The answer range is given by

$\{c_1, c_2, c_3, \ldots, c_k\}$. Then,

$$\Pr[A_i \neq c_x | A_A = c_x] \;=\; (1 - r_i)/(k-1)$$

However for most queries in practise, it is not possible to get a reasonable estimate for k. Moreover, a node can give the incorrect answer in unlimited number of ways. For a query like "What is the current temperature?", possible answers can lie anywhere on the real number line. Thus the answer range size, $k \to \infty$. So the probability with which a node gives a particular incorrect answer tends to zero.

$$\Pr[A_i \neq c | A_A = c] \to 0$$

We explored the first two cases of binary and finite answer range size and present the results in the simulation section. We assume that the possible answer space is discrete. They also belong to the same domain space since they are answers to the same question. Let A_1, A_2, A_3 be the answers returned by nodes n_1, n_2 and n_3. The reputations of the nodes are denoted by r_1, r_2 and r_3 respectively.

Using Bayes theorem, probability that the actual answer is equal to A_1 given the received answers from our neighbors is,

$$Pr[A_A = A_1 | (A_1, A_2, A_3)] \;=\; \frac{Pr[(A_1, A_2, A_3) | A_A = A_1] * Pr[A_A = A_1]}{Pr[(A_1, A_2, A_3)]}$$

Then the ratio of the probabilities of the actual answer to be equal to A_2 to it being equal to A_1, given the replies from our neighbors is,

$$\begin{aligned}\frac{Pr[A_A = A_2|(A_1, A_2, A_3)]}{Pr[A_A = A_1|(A_1, A_2, A_3)]} &= \frac{Pr[(A_1, A_2, A_3)|A_A = A_2]}{Pr[(A_1, A_2, A_3)|A_A = A_1]} * \frac{Pr[A_A = A_2]}{Pr[A_A = A_1]} * \frac{Pr[(A_1, A_2, A_3)]}{Pr[(A_1, A_2, A_3)]} \\ &= \frac{Pr[(A_1, A_2, A_3)|A_A = A_2]}{Pr[(A_1, A_2, A_3)|A_A = A_1]} \\ &\quad \textbf{(assuming equal initial probabilities)}\end{aligned}$$

We define relative likelihood that the actual answer is equal to A_1 as,

$$R[A_A = A_1|(A_1, A_2, A_3)] = Pr[(A_1, A_2, A_3)|A_A = A_1]$$

Thus we can find which of the received answers is most likely to be the true answer by computing the relative probabilities $R[A_A = A_1|(A_1, A_2, A_3)]$, $R[A_A = A_2|(A_1, A_2, A_3)]$ and $R[A_A = A_3|(A_1, A_2, A_3)]$ and choosing the answer with the maximum relative correctness probability value. Generalizing for $A_A = A_i$,

$$R[A_A = A_i|(A_1, A_2, A_3)] = Pr[(A_1, A_2, A_3)|A_A = A_i]$$

Assumption: Replies from the neighbors are conditionally independent i.e. they do not collude while replying. Then from above we have,

$$R[A_A = A_i|(A_1, A_2, A_3)] = Pr[A_1|A_A = A_i] * Pr[A_2|A_A = A_i] * Pr[A_3|A_A = A_i]$$

Case 1: $A_A = A_1$ and $A_1 \neq A_2 \neq A_3$. Of the returned answers only A_1 equals the actual answer.

$$
\begin{aligned}
&R[A_A = A_1|(A_1, A_2, A_3)]\\
&= Pr[A_1 = A_A|A_A = A_1] * Pr[A_2 \neq A_A|A_A = A_1] * Pr[A_3 \neq A_A|A_A = A_1]\\
&= r_1 * (1 - r_2) * (1 - r_3)
\end{aligned}
$$

Using the approach where the probability with which a node gives the incorrect answer as $(1 - r_i)/(k - 1)$, we get

$$
\begin{aligned}
&R[A_A = A_1|(A_1, A_2, A_3)]\\
&= Pr[A_1 = A_A|A_A = A_1] * Pr[A_2 \neq A_A|A_A = A_1] * Pr[A_3 \neq A_A|A_A = A_1]\\
&= r_1 * \frac{(1 - r_2)}{(k - 1)} * \frac{(1 - r_3)}{(k - 1)}
\end{aligned}
$$

Case 2: $A_A = A_1$ and $A_1 = A_2 \neq A_3$.Of the returned answers both A_1 and A_2 agree on the actual answer.

$$
\begin{aligned}
&R[A_A = A_1|(A_1, A_2, A_3)]\\
&= Pr[A_1 = A_A|A_A = A_1] * Pr[A_2 = A_A|A_A = A_1] * Pr[A_3 \neq A_A|A_A = A_1]\\
&= r_1 * r_2 * (1 - r_3)
\end{aligned}
$$

Using the second approach,

$$
\begin{aligned}
&R[A_A = A_1|(A_1, A_2, A_3)] \\
&= Pr[A_1 = A_A|A_A = A_1] * Pr[A_2 = A_A|A_A = A_1] * Pr[A_3 \neq A_A|A_A = A_1] \\
&= r_1 * r_2 * \frac{(1 - r_3)}{(k - 1)}
\end{aligned}
$$

Case 3: $A_A = A_4$ and $A_1 \neq A_2 \neq A_3 \neq A_4$ This represents the case where none of the answers returned by the current neighbors is the actual answer.

$$
\begin{aligned}
&R[A_A = A_4|(A_1, A_2, A_3)] \\
&= Pr[A_1 \neq A_A|A_A = A_4] * Pr[A_2 \neq A_A|A_A = A_4] * Pr[A_3 \neq A_A|A_A = A_4] \\
&= (1 - r_1) * (1 - r_2) * (1 - r_3)
\end{aligned}
$$

Using the second approach,

$$
\begin{aligned}
&R[A_A = A_4|(A_1, A_2, A_3)] \\
&= Pr[A_1 \neq A_A|A_A = A_4] * Pr[A_2 \neq A_A|A_A = A_4] * Pr[A_3 \neq A_A|A_A = A_4] \\
&= \frac{(1 - r_1)}{(k - 1)} * \frac{(1 - r_2)}{(k - 1)} * \frac{(1 - r_3)}{(k - 1)}
\end{aligned}
$$

We compare the probabilities thus computed and choose the answer having the maximum probability value. If the probability that none of the returned answers is the actual answer is the greatest, then we wait till we get the actual answer from our future neighbors.

4.1 Illustrative Examples

Example 1: The answers returned by our neighbors are $A_1 = 1$, $A_2 = 0$ and$A_3 = 0$. And their reputations are $r_1 = 0.25$, $r_2 = 0.75$ and $r_3 = 0.75$. Then we have,

$$\begin{aligned}
R[A_A = 0|(1,0,0)] &= (1-r_1) * r_2 * r_3 = 0.75 * 0.75 * 0.75 = 0.421875 \\
R[A_A = 1|(1,0,0)] &= r_1 * (1-r_2) * (1-r_3) = 0.25 * 0.25 * 0.25 = 0.015625 \\
R[A_A = x|(1,0,0)] &= (1-r_1) * (1-r_2) * (1-r_3) = 0.75 * 0.25 * 0.25 = 0.046875
\end{aligned}$$

Here x is a value other than 0 and 1. Thus $A_A = 0$ with a higher probability. Intuitively the answer that is agreed upon by greater number of trusted nodes is chosen above the answer given by fewer distrustful nodes.

We reach the same conclusion in the second case with $(k-1) = 2$.

Example 2: Values returned by neighbors are $A_1 = 0$, $A_2 = 1$ and$A_3 = 1$. And their reputations are $r_1 = 0.751$, $r_2 = 0.75$ and $r_3 = 0.75$, then

$$\begin{aligned}
R[A_A = 0|(0,1,1)] &= r_1 * (1-r_2) * (1-r_3) = 0.751 * 0.25 * 0.25 = 0.0469375 \\
R[A_A = 1|(0,1,1)] &= (1-r_1) * r_2 * r_3 = 0.249 * 0.75 * 0.75 = 0.1400625 \\
R[A_A = x|(0,1,1)] &= (1-r_1) * (1-r_2) * (1-r_3) = 0.249 * 0.25 * 0.25 = 0.0155625
\end{aligned}$$

Here $A_A = 1$ with a higher probability. Intuitively, when the reputations of the neighbors are comparable, then the answer is chosen by majority agreement among the neighbors. Here too we reach the same conclusion using the finite answer range size case.

Example 3: Values are $A_1 = 1$, $A_2 = 0$, $A_3 = 0$ and$A_4 = 0$. And their reputations are

$r_1 = 0.75$, $r_2 = 0.25$, $r_3 = 0.25$ and $r_4 = 0.25$, then

$$\begin{aligned}
R[A_A = 0|(1,0,0,0)] &= (1-r_1) * r_2 * r_3 * r_4 \\
&= 0.25 * 0.25 * 0.25 * 0.25 = 0.00390625 \\
R[A_A = 1|(1,0,0,0)] &= r_1 * (1-r_2) * (1-r_3) * (1-r_4) \\
&= 0.75 * 0.75 * 0.75 * .75 = 0.31640625 \\
R[A_A = x|(1,0,0,0)] &= (1-r_1) * (1-r_2) * (1-r_3) * (1-r_4) \\
&= 0.25 * 0.75 * 0.75 * .75 = 0.10546875
\end{aligned}$$

Thus when we have one highly trusted neighbor and several low reputation neighbors, the answer given by the highly trusted node is chosen i.e. $A_A = 1$.

Note that getting an answer from a distrustful node, that contradicts the answer given by a highly trusted node, actually makes it easier to converge on the correct answer given by the highly trusted node. To illustrate, if in the above example another node with a low reputation of $r_5 = 0.25$ gave the answer $A_5 = 1$ matching that from the highly reputed node 1. Then, the probabilities would be,

$$\begin{aligned}
R[A_A = 0|(1,0,0,0,1)] &= (1-r_1) * r_2 * r_3 * r_4 * (1-r_5) \\
&= 0.25 * 0.25 * 0.25 * 0.25 * 0.75 = 0.0029296875 \\
R[A_A = 1|(1,0,0,0,1)] &= r_1 * (1-r_2) * (1-r_3) * (1-r_4) * r_5 \\
&= 0.75 * 0.75 * 0.75 * 0.75 * 0.25 = 0.0791015625 \\
R[A_A = x|(1,0,0,0,1)] &= (1-r_1) * (1-r_2) * (1-r_3) * (1-r_4) * (1-r_5) \\
&= 0.25 * 0.75 * 0.75 * 0.75 * 0.75 = 0.0791015625
\end{aligned}$$

This makes the probability of $A_A = 1$ become equal to the probability that the actual answer is not received yet. Thus getting the correct answer from a low reputation node can delay converging on the correct answer. But will eventually lead to increasing its reputation, after the answer is proved correct. Thus a contradicting answer from a low reputation node helps in converging on the correct answer sooner.

On the other hand, with $(k-1)=2$, we still converge on the correct answer in this case.

$$\begin{aligned} R[A_A = 0|(1,0,0,0,1)] &= \frac{(1-r_1)}{2} * r_2 * r_3 * r_4 * \frac{(1-r_5)}{2} \\ &= (0.25/2) * 0.25 * 0.25 * 0.25 * (0.75/2) = 0.0007324 \\ R[A_A = 1|(1,0,0,0,1)] &= r_1 * \frac{(1-r_2)}{2} * \frac{(1-r_3)}{2} * \frac{(1-r_4)}{2} * r_5 \\ &= 0.75 * (0.75/2) * (0.75/2) * (0.75/2) * 0.25 = 0.00988 \\ R[A_A = x|(1,0,0,0,1)] &= \frac{(1-r_1)}{2} * \frac{(1-r_2)}{2} * \frac{(1-r_3)}{2} * \frac{(1-r_4)}{2} * \frac{(1-r_5)}{2} \\ &= (0.25/2) * (0.75/2) * (0.75/2) * (0.75/2) * (0.75/2) = 0.002471 \end{aligned}$$

Thus the answer chosen is $A_A = 1$. Here in contrast to the previous case where $(k-1) = 1$, the probability that we do not know the answer yet, is less than that for $A_A = 1$. Intuitively an agreeing answer even from a low reputation node, contributes in choosing that answer.

Example 4: Values are $A_1 = 1$, $A_2 = 0$ and $A_3 = 0$. And their reputations are $r_1 = 0.4$, $r_2 = 0.5$, $r_3 = 0.3$, then

$$\begin{aligned} R[A_A = 1|(1,0,0)] &= r_1 * (1-r_2) * (1-r_3) = 0.4 * 0.5 * 0.7 = 0.14 \\ R[A_A = 0|(1,0,0)] &= (1-r_1) * r_2 * r_3 = 0.6 * 0.5 * 0.3 = 0.09 \\ R[A_A = x|(1,0,0)] &= (1-r_1) * (1-r_2) * (1-r_3) = 0.6 * 0.5 * 0.7 = 0.21 \end{aligned}$$

Here the probability of the actual answer having a value other than those returned by its neighbors so far is highest. So we conclude that we have not got the actual answer yet and wait for it. Intuitively it means that we are in a low trust neighborhood and must wait till we get an answer from a more trusted neighbor.

With $(k-1)=2$, we converge on the correct answer as $A_A=0$ since it was provided by two nodes of reputations 0.5 and 0.3. So the requirement that we get an answer from a good node with $r>0.5$ is not essential anymore.

Example 5: Values are $A_1=1$ and $A_2=0$. And the reputations are $r_1=0.5$ and $r_2=0.5$ respectively, then

$$\begin{aligned} R[A_A=1|(1,0)] &= r_1*(1-r_2)=0.5*0.5=0.25 \\ R[A_A=0|(1,0)] &= (1-r_1)*r_2=0.5*0.5=0.25 \\ R[A_A=x|(1,0)] &= (1-r_1)*(1-r_2)=0.5*0.5=0.25 \end{aligned}$$

All nodes initially have a reputation of 0.5 viz. undecided and evolve as they encounter positive and negative experiences. When all current neighbors have 0.5 reputation, it means we are in a neutral neighborhood. The probabilities calculated are the same for all returned answer values. We conclude that we have not got the actual answer yet and wait for it. Intuitively we do not accept an answer from a neighbor whose reputation is in the initial stage.

We reach the same conclusion even in the second case.

Example 6: To illustrate how the validation algorithm behaves in general, we note that the difference in the reputations of the good device and the other devices needed for the good guy to win is a function of how high the reputations of the other devices in consideration are. If we are in a high trust neighborhood, viz. all answers are from devices having reputations >0.5, then the higher the other devices' reputation, the greater must be difference in reputations between the correct guy and the other guys.

For example with $r_1 = 0.7$, $r_2 = 0.6$ and $r_3 = 0.6$, and $A_1 = 1$, $A_2 = 0$ and $A_3 = 0$, node 1 wins.

However for reputations $r_2 = 0.7$, $r_3 = 0.7$ and values $A_1 = 1$, $A_2 = 0$ and $A_3 = 0$, r_1 must be as high as 0.845 for node 1 to win. Thus the difference in reputations required increases in a high trust neighborhood.

By using $(k - 1) = 2$, with $r_2 = 0.6$ and $r_3 = 0.6$, and $A_1 = 1$, $A_2 = 0$ and $A_3 = 0$, r_1 is required to be even higher, viz. $r_1 = 0.82$, for the answer $A_1 = 1$ to win. Intuitively having a k factor increases the importance of majority agreement.

Example 7: In a low trust neighborhood, even a single answer from a good guy, viz. with reputation > 0.5 is sufficient for the good guy to win. For the border case of $r_1 = 0.51$, $r_2 = 0.49$ and $r_3 = 0.49$, and $A_1 = 1$, $A_2 = 0$ and $A_3 = 0$, we have,

$$\begin{aligned}
R[A_A = 1|(1,0,0)] &= r_1 * (1 - r_2) * (1 - r_3) = 0.51 * 0.51 * 0.51 = 0.132651 \\
R[A_A = 0|(1,0,0)] &= (1 - r_1) * r_2 * r_3 = 0.49 * 0.49 * 0.49 = 0.117649 \\
R[A_A = x|(1,0,0)] &= (1 - r_1) * (1 - r_2) * (1 - r_3) = 0.49 * 0.51 * 0.51 = 0.127449
\end{aligned}$$

Thus even in this border case of reputations, the good guy wins i.e. $A_A = 1$, without having a huge difference in reputations above the bad guys.

Using $(k - 1) = 2$, the answer $A_A = 0$ wins because it was returned by two nodes having reputations only slightly less than the third node. Thus the emphasis on getting an answer from a node having $r > 0.5$ in order to believe it, is not true anymore. This can be good strategy to tolerate error in reputations. On the other hand the same answer from two low reputation nodes, wins over an answer from a high reputation node. This might lead to compromising on the credibility of data.

Thus we will accept an answer as the actual answer once its calculated probability becomes greater than probabilities of all other answers obtained. It must also be greater than the probability that we have not found the answer yet. We will wait to obtain the answer to a question from at least three neighbors before running the above calculation. We run the validation scheme to arrive at the correct answer for the questions that a node has. We also need to run the validation algorithm for other answers received, that do not correspond to any of the questions that the node has. This is because we are going to push this other data in the network if it is estimated to be popular in the network. We do not want to push data until we have validated it.

The answer validation scheme presented above relies on an reputation evolving scheme to provide the input reputation values. The reputation evolving scheme must give a fairly accurate estimate of peer reputations. As we saw in example 7, the proposed validation algorithm with $(k-1)=1$, is sensitive to accuracy of reputation values around 0.5. Thus a correct distinction between devices as good $(r > 05.)$ or bad $(r < 0.5)$ is sufficient to yield the correct answer. In a high reputation environment, if the reputation values are incorrect and the peers provide different versions of answers, it might take longer to converge on the correct answer. It might even lead to concluding on the incorrect answer if a sufficient number of incorrectly, highly reputed nodes agree on the wrong answer.

Chapter 5

SIMULATIONS

5.1 Query Distribution Estimation

We implemented the query estimation and predictive caching algorithm using the Glomosim (Zeng, Bagrodia, & Gerla 1998) simulator. The information in this section was generated using the simulation parameters mentioned in Table 5.1.

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 (Patwardhan *et*

Table 5.1. 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

al. 2006). 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.

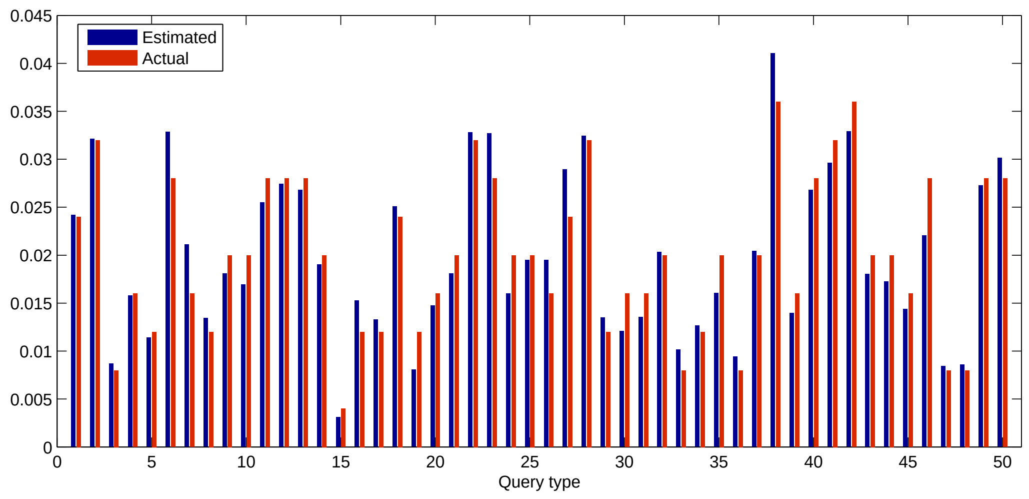

FIG. 5.1. Query distribution plot of the actual and estimated distributions after 5 minutes of simulation run time

Figure 5.1 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.

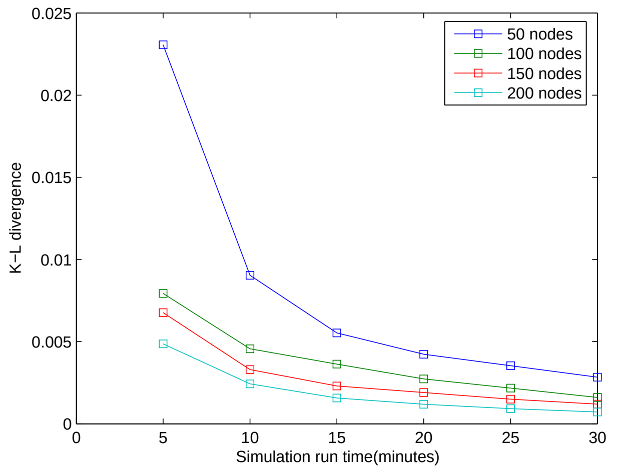

FIG. 5.2. Kullback-Leibler divergence between the average estimated and actual query distributions

More formally we measure the Kullback-Leibler divergence (kld 2008) or relative entropy between the actual and estimated query distributions. We see in figure 5.2 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 5.3 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.

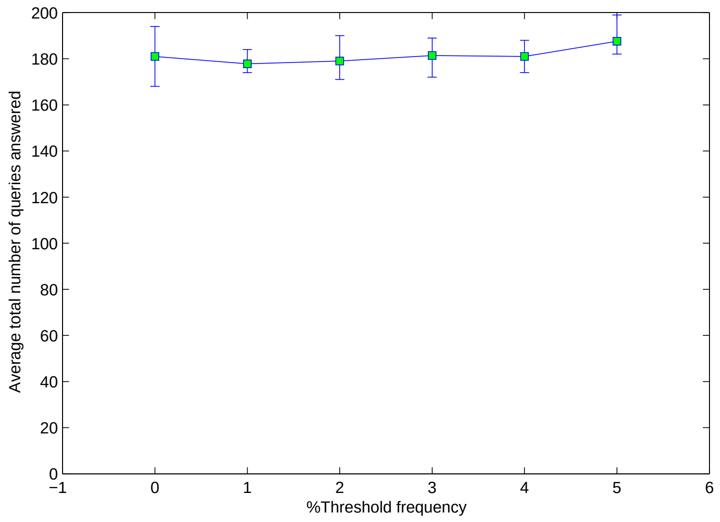

FIG. 5.3. Variation in number of queries answered as the threshold frequency is increased from 0-5%

A threshold frequency of 0% implies that all cached data is pushed in 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%.

The average time to answer a query clearly increases as the percentage threshold fre-

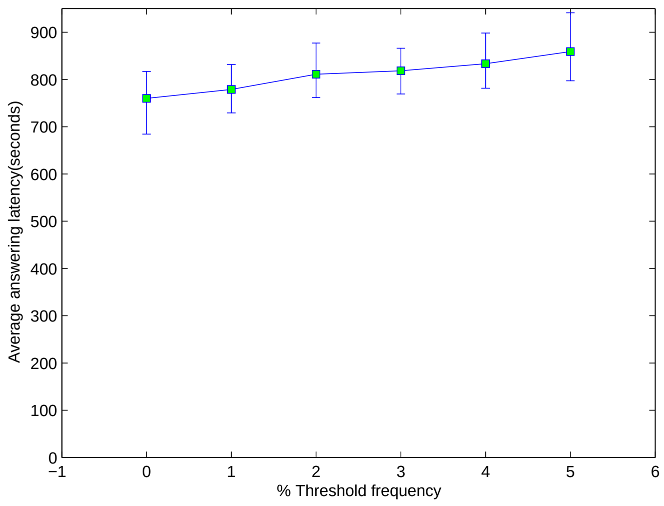

FIG. 5.4. Variation in average query answering delay as the threshold frequency is increased from 0-5%

quency increases as seen in figure 5.4. 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 5.5 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.

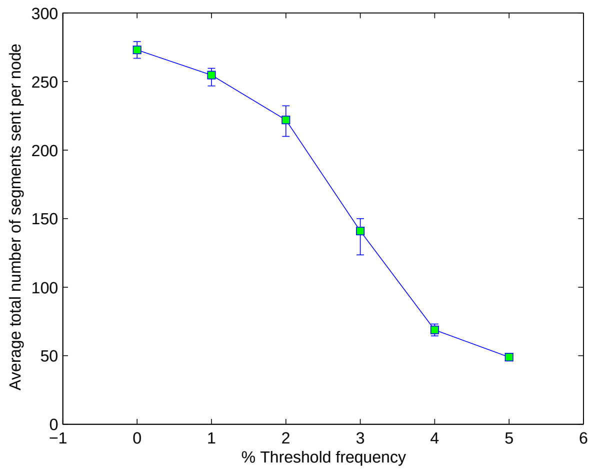

FIG. 5.5. Variation in average total segments sent per node as the threshold frequency is increased from 0-5%

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 5.6 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 network. This results in decreased total number of answered queries. The number of answered queries in-

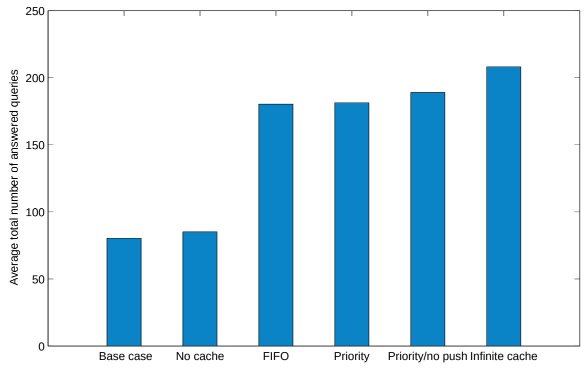

FIG. 5.6. Average total number of queries answered for different caching schemes for a threshold frequency of 3%

creases slightly for the priority-based caching scheme as compared to that for FIFO caching scheme. This can by 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 5.8.

Figure 5.7 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

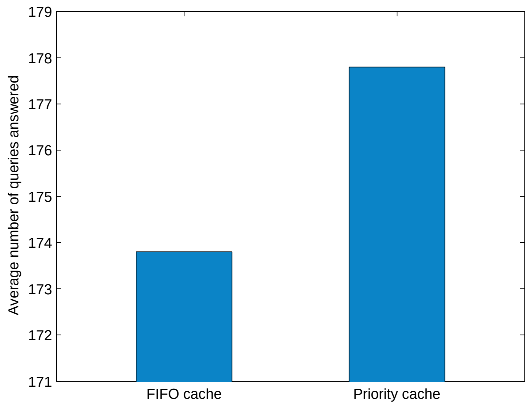

FIG. 5.7. Average total number of queries answered for priority and FIFO caching schemes for a threshold frequency of 1%

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.

As seen in figure 5.8 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 or peer nodes that know the answer, to have its query answered. Note that while calculating answering latency, we have not posed any penalty for unanswered queries. So for the schemes where more questions were answered, it seems that the answering latency was more. This is because more the number of answers obtained, greater the chance that many answers were obtained during the later parts of the simulation. This boosts the average answering latency. Hence it turns out to be greater for cases where more answers were

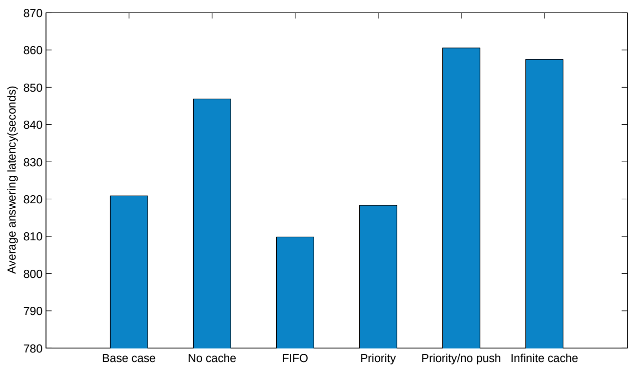

FIG. 5.8. Average query answering delay for different caching schemes for a threshold frequency of 3%

obtained.

The no cache case performs poorly in terms of answering latency as expected since there is no cache in the mobile nodes for storage of the pushed data in the network. The base case, appears to perform better than the no cache case. However this is due to the reason explained above, as more answers were obtained in no cache case, the latency appears greater in that case. 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%. As the maximum number of answers were obtained in the infinite cache case, the latency appears to be the greatest.

Figure 5.9 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

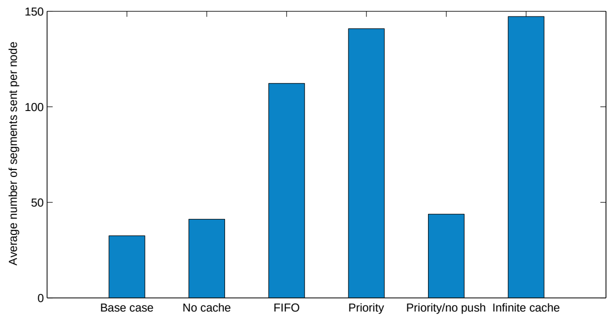

FIG. 5.9. Average total number of segments sent per node for different caching schemes for a threshold frequency of 3%

the traffic overhead caused due the query estimation algorithm and pushing of data in the network.

5.2 Query Containment

We experimented with having some of the queries contained in other queries. Query containment affected the working of the algorithm in three ways. The nodes can now answer queries that are contained in other queries that they have the answer to. The nodes will further filter the answer to return only the data requested for by the contained query. While collecting statistics about the query distribution, the queries will now also increment frequency of the query containing the encountered query. This is to increase popularity of the containing query, so that devices cache the corresponding data with a higher probability. Thirdly, this will lead nodes to now also push answers to queries that contain popular queries, even if the containing query itself is not that popular, thus increasing the availability of popular data.

Figure 5.10 shows that the estimated query frequency is much greater than the actual

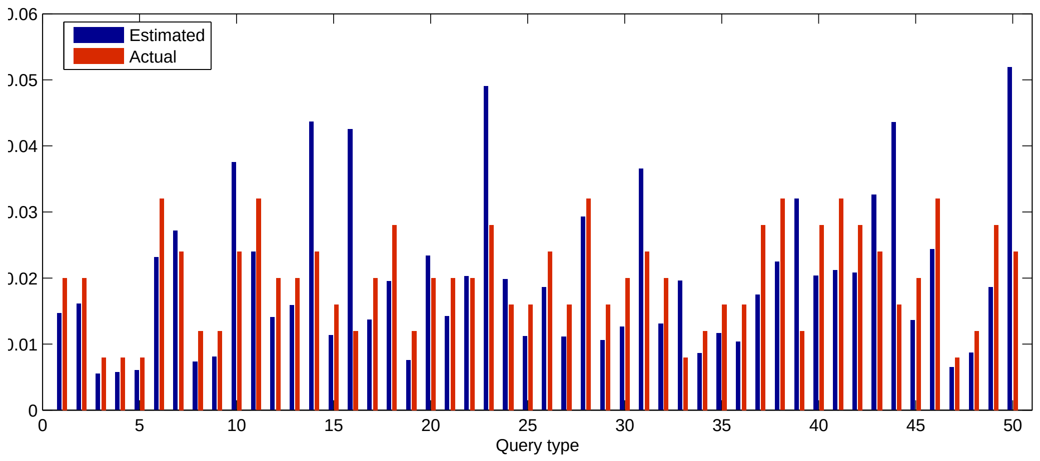

FIG. 5.10. Query distribution plot of the actual and estimated distributions after 30 minutes with 50% query containment in the network

frequency for some queries. This happens because these queries contain other queries, so their estimated popularity increases beyond the value that would have been obtained if we had considered the queries independently. For example in this simulation run, the query 49 is important because it contains two other queries, 0 and 22. So are the queries 13, 15, 22 and 43. These queries have a higher estimated frequency than their actual frequencies.

Figure 5.11 shows that the total number of queries answered remains almost the same as the percentage query containment is increased from 0% to 50%. However this is accompanied with a decrease in the answering latency as seen in figure 5.12.

The answering latency decreased from 818 seconds to 709 seconds as query containment was increased from 0% to 50%. The decrease in answering latency is expected since now even the nodes having answers to the containing query can answer the contained queries posed to them.

The total traffic in the network seems to increase as seen in figure 5.13. This increase is because even the data corresponding to queries containing popular queries are now pushed

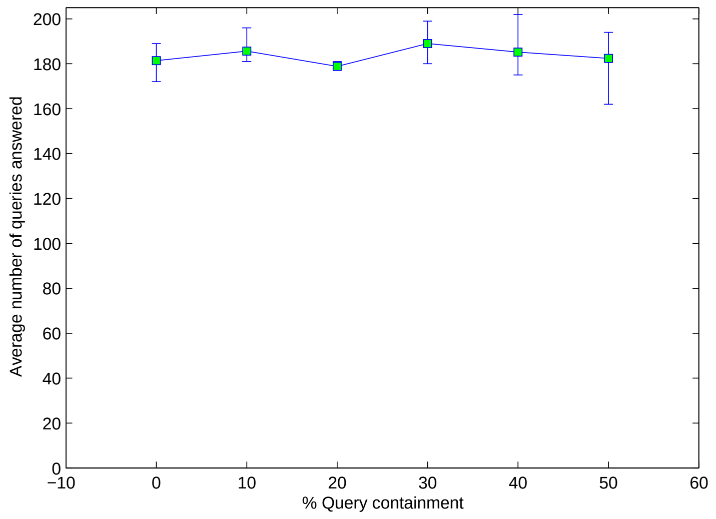

FIG. 5.11. Average total number of queries answered for varying percentages of query containment

in addition to pushing the popular data. Thus more data qualifies for pushing with query containment.

5.3 Bayesian Approach to Data Validation

We ran simulation experiments to test the performance of our Bayes theorem based validation algorithm. The percentage of bad nodes in the network was increased from 0% to 100% in steps of 10%. Reputation values are assigned to the nodes initially and do not change during the simulation. The good nodes are assigned reputation values > 0.5 and bad nodes are assigned values < 0.5. The bad nodes also run the validation algorithm to obtain the correct answer. However they push incorrect answers into the network. The good nodes too perform the validation step, but only push validated answers into the network. The graphs show the results for 3% popularity threshold level. The effect on the

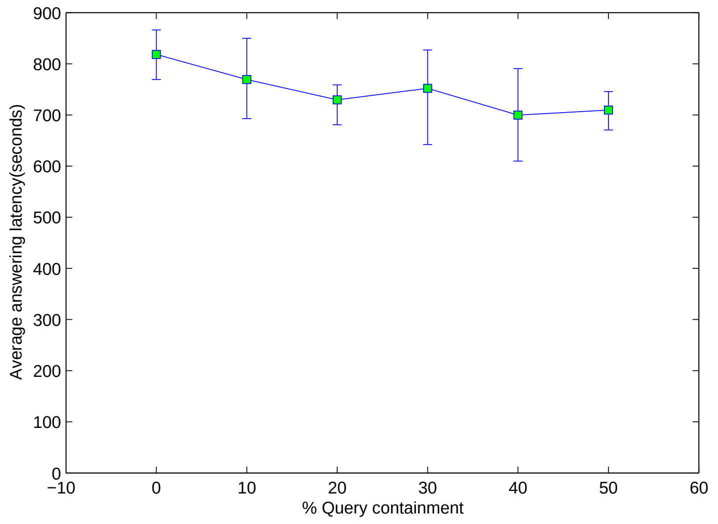

FIG. 5.12. Average answering latency for varying percentages of query containment

number of correct answers obtained in the network, the answering latency and total traffic in the network was observed as the fraction of bad nodes is increased in the network. The performance of this validation algorithm is compared to that of the validation algorithm described in (Patwardhan *et al.* 2006). This algorithm serves as the base case for performance comparison. The good nodes having reputation say 0.7 also gives the incorrect answer with a 0.3 probability. Similarly bad nodes also return the correct answer with a small probability. This setup might give a better representation of a real-world scenario where good nodes cannot be expected to be correct at all times.

Figure 5.14 shows the average total number of correct and incorrect answers obtained in this setup. It shows that our validation algorithm performed consistently better in terms of total number of correct answers obtained. The number of correct answers obtained dropped significantly as the percentage of bad nodes was increased beyond 60%. This is because in a low trust neighborhood, most of the received answers come from distrustful

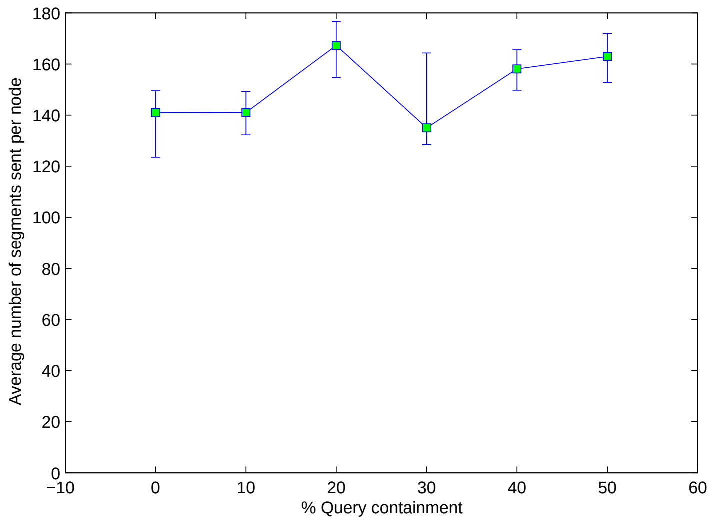

FIG. 5.13. Traffic generated per node for varying percentages of query containment.

sources. Thus the validation algorithm calculated that probability of the answer not being known as the highest. Hence most queries remained unanswered though they received multiple copies/versions of answers from the bad nodes.

It concluded on the wrong answer more often in high reputation environments, than the base case algorithm. This is because when a high reputation node gives the incorrect answer, the proposed algorithm will choose it as the correct answer, since it relies on correctness of reputation values. The base case algorithm does not consider the reputations at all, but relies on reaching a threshold agreement. So it results in fewer incorrect answers in high-trust scenarios. However, our algorithm performed better in low-trust environments. The number of incorrect answers obtained was fewer than the base case in such environments and reached a value of 0 at 100% bad nodes.

The mean answering time shows an increasing trend, on increase in the number of bad nodes as seen in figure 5.15. It increased from 817 seconds to 870 seconds from 0% to

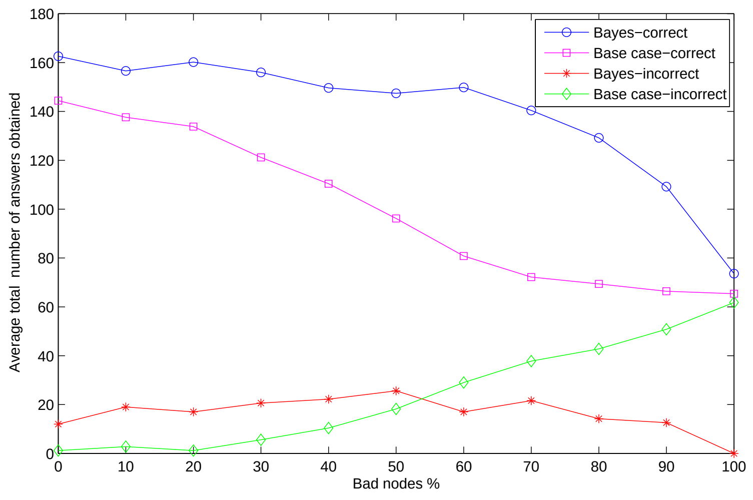

FIG. 5.14. Number of correct and incorrect answers obtained as the percentage of malicious nodes in the network is increased.

100% bad nodes. This is due to the same reason that it takes longer to encounter and get an answer from a trusted node as the number of bad nodes in the network is increased. Hence it takes longer to decide on the correct answer, thus increasing the answering latency.

Figure 5.16 shows that the total traffic in the network decreases for greater than 70% bad nodes. This follows from our observation for the number of validated answers in the network which also decreases after 70% bad nodes. As the devices have fewer validated answers in its cache, the amount of data that lies above the popularity threshold and thus pushed in the network also decreases.

We changed the nodes behavior so that the good nodes $(r > 0.5)$ always return the correct answer and the bad nodes $(r < 0.5)$ always give the incorrect answer. As figure 5.17 shows the number of correct answers obtained using our validation algorithm remained almost constant at around 174 answers till the percentage of bad nodes was increased from 0% to as much as 60%. The number of correct answers decreased after 70% and more

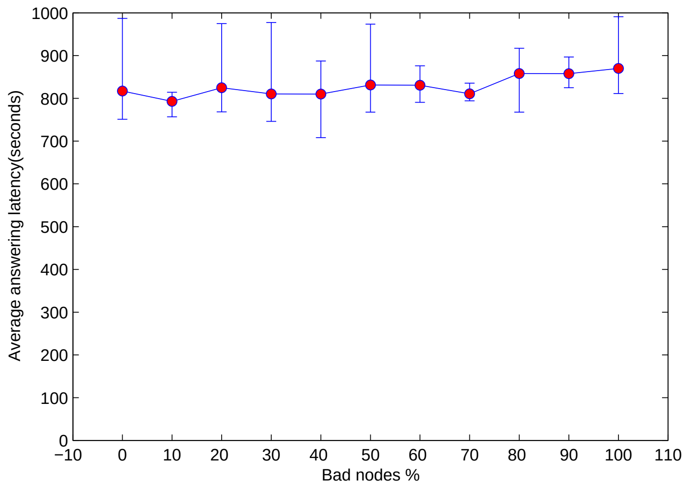

FIG. 5.15. Average answering latency for our validation algorithm as the percentage of bad nodes is increased

malicious nodes were introduced in the network. Yet the total number of correct answers obtained remained consistently greater than that obtained using the base case algorithm. The number of answers incorrectly assumed to be true was 0 in case of our validation algorithm even as the percentage of bad nodes in the network was increased to 100%, which indicates very good performance of the algorithm in terms of accuracy. At 100% bad nodes, the only way to get an answer is by direct encounter with the source of data and not through the cache of any peer devices. On the other hand, in the base case, where only consensus is used to determine the correct answer and the reputations of the providing sources are not considered at all, the number of incorrect answers was non-zero starting at 20% bad nodes. This increased as the fraction of bad nodes was increased. The number of incorrect answers became greater than the number of correct answers at 80% bad nodes in the network.

We implemented the second approach where the probability with which a node gives

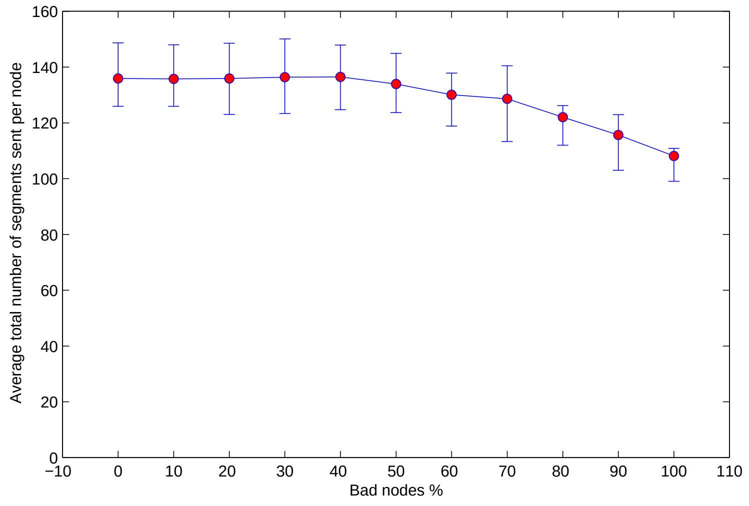

FIG. 5.16. Traffic generated per node for the Bayes theorem based validation algorithm

the incorrect answer is equal to $(1-r_i)/(k-1)$. Figure 5.18 shows the number of incorrect and correct answers obtained when there were 50% bad nodes in the network, for different values for $(k-1)$. We varied $(k-1)$ from 1 which is the same as in previously mentioned experiments, to 10. We also tried to estimate k by counting the number of unique answers we obtain for a given query. We observe from the figure that the number of correct answers is the maximum at $(k-1)=2$, and the number of incorrect answers is also the least in this case. Otherwise the performance for all other values of $(k-1)$, is worse than the base case where we divide by 1, with more incorrect answers and fewer correct answers.

Figure 5.19 shows the number of correct and incorrect answers obtained using both the approaches as the number of bad nodes in the network are increased. We observe from figure that the performance is almost the same up to around 50% bad nodes. Greater number of correct answers are obtained using $(k-1)=2$ in highly distrustful environments. This is because in low trust environment, the bad nodes return the correct answers with

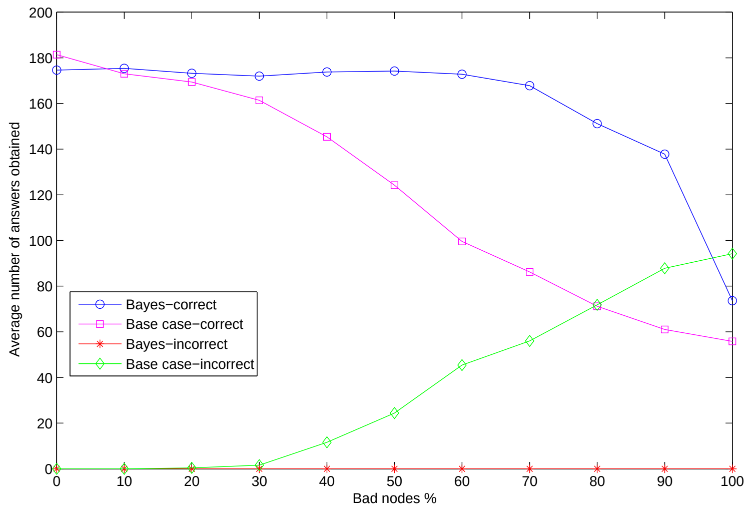

FIG. 5.17. Number of correct and incorrect answers obtained using the new setup.

a small probability. With $(k-1)=2$, the algorithm is able to conclude on the correct answer returned by bad nodes if sufficient number of nodes agree on the correct answer. With $k-1=1$, the algorithm rejects the right answer if it comes from a low reputation node. However, the number of wrong answers also increases for $(k-1)=2$ as compared to $(k-1)=1$. This is due to the same reason that if enough number of nodes agree on the wrong answer, the algorithm takes it to be the correct answer.

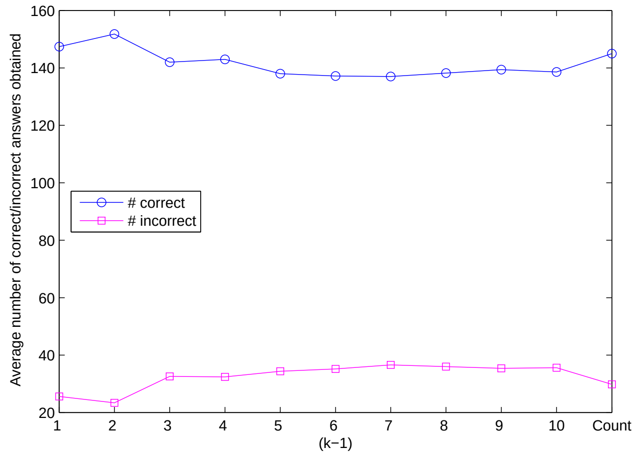

FIG. 5.18. Average number of correct and incorrect answers obtained for 50% bad nodes in the network for different values for the heuristic (k-1) factor

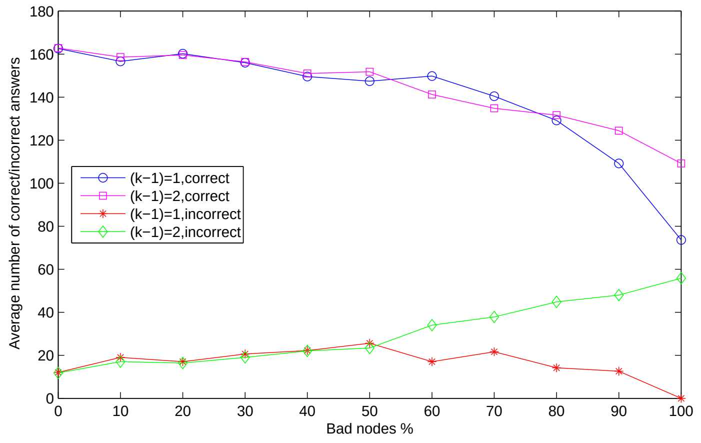

FIG. 5.19. Comparison of the number of correct/incorrect answers obtained at (k-1)=1 viz. base case and when (k-1)=2.

Chapter 6

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 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 better utilize the available cache space to increase the data availability in the network.

We proposed a data validation scheme based on Bayes theorem. The nodes consider the data and the data source reputation to determine the most correct answer. We looked at how the algorithm works for different cases of data values and reputations. We observed the performance for the two cases of binary and finite answer range size. Simulation results showed that the algorithm using $k - 1 = 1$ performs very well in terms of reliability and accuracy of data in low trust environments ($> 50\%$ bad nodes). In high trust environments($< 50\%$ bad nodes), majority voting works best in terms of accuracy of data.

For future work we are investigating how to avoid the computation involved in validating other answers received by a device i.e. the answers that do not correspond to any of the queries that the device has. We can do that by assigning an accuracy level to each answer received without going through the validation algorithm. The accuracy a_X can be calculated by using a trust-weighting function to pro-weight a suggested accuracy degree of an answer sa_X provided by a device X with the trust in that device r_X. We plan to use the formula given by Perich *et al.* (Perich *et al.* 2004),

$$a_X(i) = sa_X(i) * r_X$$

An answer received directly from an anchored source is cached with a confidence level of 1. For answers received from peer devices, the above formula is used to determine the confidence in the answer. When pushing answers in the network, devices also push their confidence values with the answers. The validation algorithm is used to determine the correct answer corresponding to questions that a node has. After validating an answer, it is cached with a confidence level of 1.

This would lead to saving of computation for answers that we receive but are not actually interested in. This scheme of associating a confidence level with each answer pushed in the network will help in making the right decisions about reputation evolution of devices based on the answers suggested by them. The reputation evolution mechanism will not only consider the answer value but also its suggested accuracy while updating reputation values.

Another interesting future work area would be in reputation evolution. Assuming that nodes know the correct answer beforehand, they can use the answers provided by peer devices to correct their estimates of peer reputations. They may use the Bayes theorem approach or some other technique to find the correct reputations.

REFERENCES

[1] Acharya, S.; Alonso, R.; Franklin, M.; and Zdonik, S. 1995. Broadcast disks: data management for asymmetric communication environments. 199–210.

[2] Budiarto; Nishio, S.; and Tsukamoto, M. 2002. Data management issues in mobile and peer-to-peer environments. *Data Knowl. Eng.* 41(2-3):183 – 204.

[3] Cherniack, M.; Franklin, M. J.; and Zdonik, S. 2001. Expressing user profiles for data recharging. *Personal Communications, IEEE* 8(4):32–38.

[4] Dash. 2008. Website. `http://dash.net`.

[5] Jonker, C. M., and Treur, J. 1999. Formal analysis of models for the dynamics of trust based on experiences. In *MAAMAW '99: Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, 221–231. London, UK: Springer-Verlag.

[6] 2008. Relative entropy. `http://mathworld.wolfram.com/RelativeEntropy.html`.

[7] Libkin, L., and Zhang, Z. 2008. Topics in DB: Foundations of XML (CSC2538) XPath Query Containment. `http://www.cs.toronto.edu/~libkin/csc2538/zhengzhang.pdf`.

[8] Patwardhan, A.; Joshi, A.; Finin, T.; and Yesha, Y. 2006. A data intensive reputation management scheme for vehicular ad hoc networks. *Proceedings of the Second International Workshop on Vehicle-to-Vehicle Communications* 1–8.

[9] Perich, F.; Avancha, S.; Chakraborty, D.; Joshi, A.; and Yesha, Y. 2002. Profile driven data management for pervasive environments. *Proceedings of the 13th International Conference on Database and Expert Systems Applications* 361 – 370.

[10] Perich, F.; Undercoffer, J. L.; Kagal, L.; Joshi, A.; Finin, T.; and Yesha, Y. 2004. In Reputation We Believe: Query Processing in Mobile Ad-Hoc Networks. In *International Conference on Mobile and Ubiquitous Systems: Networking and Services*.

[11] Schwentick, T. 2008. XPath Query Containment. `http://www.cs.utoronto.ca/~libkin/dbtheory/thomas.pdf`.

[12] Shim, J.; Scheuermann, P.; and Vingralek, R. 1999. Proxy cache algorithms: Design, implementation, and performance. *IEEE Trans. Knowledge and Data Eng.* 11(4):549–562.

[13] Wu, Y.-L.; Agrawal, D.; and Abbadi, A. E. 2002. Query estimation by adaptive sampling. *Proceedings of the ICDE Conference* 639–648.

[14] Xu, B., and Wolfson, O. 2004. Data management in mobile peer-to-peer networks. *2nd International Workshop on Databases, Information Systems, and Peer-to-Peer Computing (DBISP2P'04).*

[15] Yin, L., and Cao, G. 2006. Supporting cooperative caching in ad hoc networks. *IEEE Transactions on Mobile Computing* 5(1):77–89.

[16] Zeng, X.; Bagrodia, R.; and Gerla, M. 1998. Glomosim: A library for parallel simulation of large-scale wireless networks. *Workshop on Parallel and Distributed Simulation* 154–161.