An Enhanced Quorum Selection Algorithm

Samer Younes and Ramzi A. Haraty
Department of Computer Science and Mathematics
Lebanese American University
Beirut, Lebanon
Email: {samer.younes@lau.lb, rharaty@lau.edu.lb}

Abstract---**As communication becomes more and more an integral part of our day to day lives, so our need to access information increases as well. Mobility is currently one of the most important factors to consider in our aim to achieve ubiquitous computing, and with it raises the problem of how to manipulate data while maintaining consistency and integrity. Recent years have seen tremendous interest in quorum systems adapted to mobile hosts; however, the more recent topic, of studying the effects of mobile networks on quorum systems, has also been the focus of interest for building quorums aware of their network surroundings. This paper presents a novel approach in selecting mobile hosts to form epidemic quorum coteries, based on metrics measured by mobile hosts and then transmitted to base station servers, which maintain a vigil on the state of these mobile hosts to provide higher quorum availability and ultimately higher data accessibility, better integrity and consistency.**

Index Terms---**quorum selection, mobile environments, and cost metrics.**

I. INTRODUCTION

Pervasive computing is a term loosely used to describe the current state of computer technology in modern life. Our reliance on computing mediums increases with the need for mobility, connectivity and data availability.

As an example, in a single day, the average individual can go through a minimum of three different devices to perform various everyday tasks such as checking his/her email account on the desktop computer, calling a family member on the cell phone, listening to some music in the background on his/her personal laptop and syncing all appointments from his/her palm-pilot to his/her email client. Although we take such things for granted, our daily interaction with data keeps increasing, and with it the need to keep our data accessible, well-organized and safe.

The quorum algorithm has been extensively studied since its earlier days, adapting it to mobile devices with connectivity issues and providing a solid quality of service for quorum members is still in its infancy. The adaptation of quorum consensus to mobile environments to insure a high level of service is one of the main challenges to tackle.

The architectures for mobile database systems have been varied and diverse; however, all these architectures still adhere to the ACID principals of standard databases systems. Serrano-Alvarado et al [21] provide an excellent overview of the various mobile database models, the most popular of which, according to [21] and Kumar [11], are the clustering model introduced by Pitoura and Bhargava in [18], the 2-Tier replication model introduced by Gray et al in [4], the HiCoMo model presented in [13] by Lee and Helal, and the Pro-Motion model by Walborn and Chrysanthis [23] [24].

With a comprehension of the workings of these various models, recent publications by Holliday et al [8], [9] and Baretto Ferrero [1], seem to agree that quorum systems are the best suited for a mobile environment. They present a study of how epidemic algorithms can help increase the reliability and availability of mobile database systems.

Very recently, the interest in studying the effects of mobile network environments on the performance and availability of quorum systems has spurred interesting publications in this area; most notable of which are, [17] by Peysakhov et al. that provides a general quorum availability evaluation and Baretto Ferrero [1] that deals more specifically with the performance evaluation of epidemic quorum algorithms.

Other work by Gupta et al. [7] and Golovin et al. [4] were also studied, pertaining to quorum placement and congestion management, but the findings, although very interesting, were left as future improvements on the architecture presented herein.

This paper presents a novel approach in selecting mobile hosts (MHs) to form epidemic quorum coteries, based on metrics measured by mobile hosts and then transmitted to base station servers, which maintain a vigil on the state of these mobile hosts to provide higher quorum availability and ultimately higher data accessibility, better integrity and consistency.

The remainder of this paper is organized as follows: section 2 presents the epidemic quorum algorithms. Section 3 discusses the improved quorum selection algorithm architecture. Section 4 concentrates on the algorithms used in our approach. Section 5 provides the performance evaluation, and section 6 presents the conclusion.

II. EPIDEMIC QUORUM ALGORITHMS

Epidemic Quorum Algorithms (which will be referred to as eQuorums from hereon), are a particular breed of epidemic algorithms which, like the normal quorum algorithm, operate on the same basis with some particular

features that make them fit for distributed environments. eQuorums are used as substitutes to the standard pessimistic epidemic algorithm (ROWA) in environments that require high system throughput, by allowing one transaction to commit from each set of conflicting transactions. As mentioned previously, transactions in eQuorums are serialized in a causal fashion, so that out of each pair of conflicting transactions, one and only one transaction will commit through a yes, no, vote by site quorums. Vote results are stored in a log that indicates the local site's time, vote result and the identification number of that site. This log entry is then propagated to other sites through eQuorum messages until all sites have received the vote results. Vote results are sent from all sites to all sites and are propagated according to availability, among other factors. When a particular site receives a positive vote for a particular transaction, it automatically commits the data for that transaction. When a particular transaction commits at a certain site, all other conflicting transactions at that site are aborted.

The property of eQuorums is the availability of quorums at various sites. The goal is to maximize the availability of quorums and to increase the probability of the system, eventually reaching a global consensus, and thus agreeing on a given value. [1] provides a good overview of the performance of eQuorums and provides tangible metrics through which performance can be measured. [1] also presents a unified framework to compare and measure various epidemic quorum algorithms. The main goal of this work is to ensure that the availability of quorums is maximized, providing improved overall system performance. eQuorums propagate data items based on per site quorum votes. However, given that some sites may be unavailable at times, or may require more information (in the case of uncertain vote outcome), eQuorums perform a finite number of voting rounds; the outcome of which may be a second round of votes (if uncertain) or a decision. The work refers to these two metrics as probabilistic values represented by rep_ε for the repeat probability condition and dec_ε for the decision probability condition.

The probabilistic properties of rep_ε and dec_ε are as follows:

- $rep_\varepsilon(n) + dec_\varepsilon(n) \le 1$
- and $\forall n : rep_\varepsilon(n) < 1$

Following the above workings of eQuorum votes and its probabilistic constraints, [1] has defined the availability of an eQuorum by the formula given in (1):

$$\sum_{n=0}^{y} \frac{\binom{y}{n}(1-p_f)^n p_f^{(y-n)} dec_\varepsilon(n)}{1-rep_\varepsilon(n)} \qquad (1)$$

where p_f is the probability of failure of a given host to vote, part of the numerator expression $\binom{y}{n}(1-p_f)^n p_f^{(y-n)} dec_\varepsilon(n)$ represents the probability of having n correct processes out of y, and $\frac{dec_\varepsilon(n)}{1-rep_\varepsilon(n)}$ represents the probability of consecutive repeat votes followed by a decide vote.

Although [1] assumes p_f to be constant and uniform, in reality, given the volatile nature of wireless networks, the probability of failure cannot be fixed or defined ahead of time, as disconnections may occur randomly and without prior precursors. However, a general behavioral pattern for p_f can be deduced and applied to equation (1). The goal is to minimize the probability of failure p_f to maximize availability. It is also worth mentioning that as rep_ε and dec_ε grow, availability grows.

Although the sensing and incorporation of network states in quorums is a very recent topic of discussion, the most notable work in this area of research has been done by Peysakhov et al. [17] and Gupta et al. [6][7]. The approach brought forth by [17] uses the same general principals. However, instead of using the standard client/server model, [17] uses migrating agents applied to standard quorums. As for the metrics evaluated in [17], they use a probability density function (equations (2) and (3)), similar to equation (1), of positive versus total number of votes, to calculate a confidence factor.

$$\binom{y}{n}(1-p_f)^n p_f^{(y-n)} \qquad (2)$$

and

$$F(C) = \sum_{k \in C} f(x=k) \ge 0.9 \qquad (3)$$

where C is a pre-selected confidence interval and F(C) denotes the combined probabilities of all the members of C. The process of measurement works by continually collecting votes until such time as the uncertainty threshold (in the above example 0.9 ~ 90%), defined as values lying outside of the interval C, is reached. On a given site these values would tend to indicate that a certainty for either a positive or negative outcome of the votes has been reached by the quorum. Although [17]'s method enhances the general confidence in a quorum, the presented approach deals only with the quality of the host as measured prior to

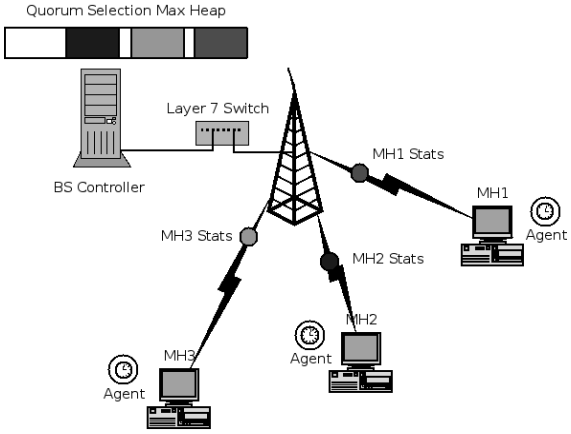

Figure 1. IQSA general architectural overview.

quorum selection. Furthermore, an agent approach to data collection suffers in weakly connected networks. The measures presented in this work would also help agents find better hosts to collect data from, reducing the amount of failed visits an agent may encounter.

III. IQSA ARCHITECTURE

This section explains the architecture and various modules used to achieve better quorum host selection, based on metrics measured by the mobile hosts pertaining to their state, and sent periodically to the base server (BS). We introduce the following metrics and variables to measure the performance, connectivity and health of a particular mobile host.

- Signal Strength: Defines the current signal strength of a MH, currently registered with a BS. In general, the signal strength can be expressed as a percentage of the maximum speed available on a particular wireless network.
- Host Priority: Depending on the current signal strength of the MH, a priority number is given to that MH. The higher the signal strength, the higher the priority. This can be set directly by the MH, or if selected as part of the quorum in a given BS, the BS may assign that MH a priority, based on measurements by the BS.
- Host Trend: We also define a derived metric based on the performance of a MH's signal strength with time. The trend of an MH's signal strength is calculated using a standard weighted linear regression, which shows the trend over time of the aggregates of both aforementioned metrics.

The BS on the other hand, once it receives this data from a MH, will classify it in a heap structure that allows the BS to pick the best performing MHs when a quorum is being built. The heap structure on the BS is constantly maintained as a max heap. Every new mobile host addition to the MH, will force the BS to restructure the max heap to take into account this new addition. Figure 1 provides more insight into the architecture, by showing how packets get arranged in the BS' max heap structure. The max heap keeps a record of all currently registered MHs with it, and orders the list, based on the metrics mentioned prior, from largest (most reliable host) to smallest (least reliable host).

The architecture also introduces the means to migrate MHs from one BS to another. Since the MH is essentially a mobile unit (cell phone, PDA, Laptop) with a moving user, for example, this user may travel through various BSs while going to work, and as such may travel and register with various base stations (similar again to the mobile phone network). So as not to clutter every BS with stale data, the MH will automatically be unregistered from a BS once that MH leaves its area of coverage. When a user registers with a new BS, the MH will automatically send the new BS the previous BS's address it was registered with.

This allows both BSs to communicate with each other and migrate the entry from the previous BS to the new BS. In some cases, mobile users may swing between two or more BSs, creating heavy migration traffic. In these cases, the MH may opt to remain registered with a particular BS as long as the cell that the user was registered in, is adjacent, in terms of area of coverage, to the the cell he/she's currently in. Figure 2 illustrates such a scenario, where MH7 migrates from a BS's coverage of cell C7 to a BS covering cell C6. The entry for MH7 is automatically migrated from C7 to C6 by communication between the BS's of both cells, as soon as the MH provides the new BS with the address of the previous BS.

The adjacency scenario can also be inferred from Figure 2 as well, where C7 has adjacent cells C6, C5 and C4 through which the BS, if present in any of the three, may use them as a bridge to communicate directly with C7, without re-registering with either C4, C5 or C6.

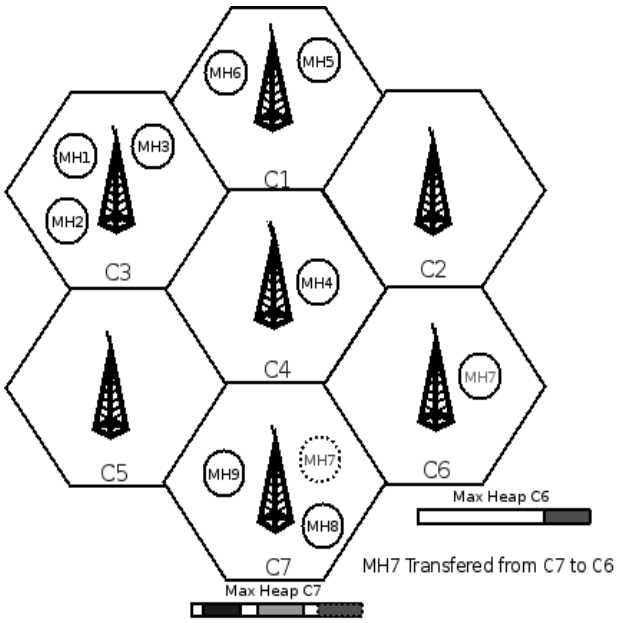

Figure 2. IQSA architecture mobile host migration process.

IV. THE ALGORITHMS

In this section, we discuss the high level algorithms that need to be implemented, in order to mimic the architecture discussed in the previous chapter. Although the core parts of it have been implemented in C++, this chapter presents the various modules in a more high level descriptive language.

Figure 3 shows the various high level parts and their high level interaction with one another.

The client module is responsible for serializing the data and sending it over the wire to the server. When a server receives a packet from a client, it stores and classifies the

client data into a max heap structure, based on the priority metric measured at the client end. The last module involves the inter-server data exchange system, which allows servers to exchange information about clients when they migrate from one server to the other and complete registration.

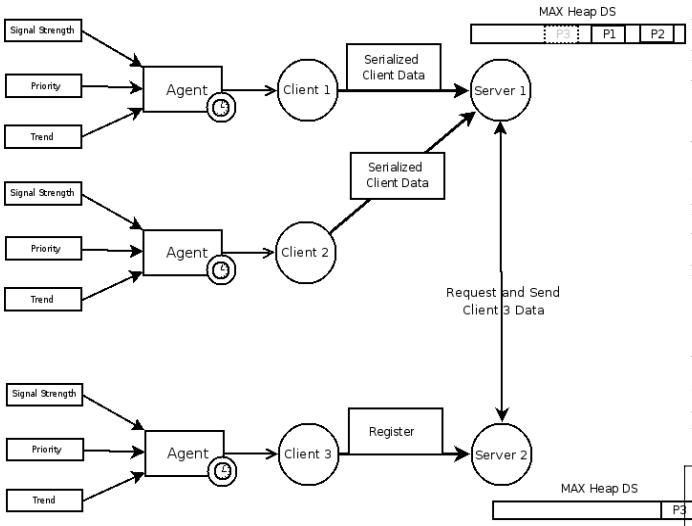

Figure 3. IQSA high level modules interaction.

A. Client Agent Procedures and Data Structures

The following is the Client data structure of metrics and measurements.

type: HostData
record
SignalStrength : int
Hostname : string
PreviousBS : string
Priority : float
Trend : double
Time : vector of double
PrioSig : vector of double

Of the aforementioned variables, most should be familiar from previous chapters, with the notable exceptions of the Time vector structure and the PrioSig vector structure. The Time structure, as the name indicates, holds a timestamp for every measure of both the SignalStrength and Priority performed by the agent, at specific intervals metrics. Every time these two metrics are measured, a timestamp is added to the dynamic array. This also holds true for the first time the structure is being initialized as well. The PrioSig structure on the other hand, holds an aggregate weighted measure of both the Priority metric and SignalStrength metric. These two vector structures are used to store historical data, needed by the linear regression function, which measures the host's trend metric.

When a MH first connects, it initializes all its metrics with either the most current measure taken (signal strength and priority), or to default values (trend metric). The trend metric is initially set to zero due to the unavailability of data to perform regression calculation. A minimum of two datum points are needed to perform a linear approximation, hence the greater than or equal to 2 condition in the calculateTrend procedure. Calculations of a weighted aggregate value combining both signal strength and priority metrics is also performed. The aggregation of these two metrics and its subsequent use in the linear approximation, would allow the environment to control which metric should have a higher importance, by assigning it a variable weight factor. The ensuing aggregate value is then associated with a timestamp in order to calculate the trend metric over time. This calculation is performed every time the client agent gets an updated measure on the basic metrics: signal strength and priority. The updates for the signal strength, priority metrics, and trend metric have been split into two procedures to make allow greater control over when and in which order these two procedures get called. Once all the data has been updated, the client serializes the host data and sends it over the wire to the BS.

```
calculateTrend(PrioSig, Time) :
   begin:
      if { sizeof(PrioSig) ≥ 2 } then
         Linear A = CalculateLinearReg(sizeof(Time),
PrioSig, Time);
         return getSlopeOf(A);
      else
         return 0;
   end;

calculateWeightedAggregate(SignalStrength, Priority) :
   begin:
      signalWeight ← signalImportanceFactor;
      priorityWeight ← priorityImportanceFactor;
      return
(SignalStrength*signalWeight)+(Priority*priorityWeight);
   end;

initialize(HostData) :
   begin:
      SignalStrength = measured(SignalStrength);
      Hostname ← gethostbyname();
      Priority ← measured(Priority);
      Time[] ← push { time(now) };
      PrioSig[] ← push {
weightedLinearDist(SignalStrength, Priority) };
      Trend ← calculateTrend(PrioSig, Time);
   end;

updateMetrics(SignalStrength, Priority) :
   begin:
      Time[] ← push { time(now) };
      PrioSig[] ← push {
weightedLinearDist(SignalStrength, Priority) };
   end;
```

```
updateTrend() :
   begin:
      trend ← calculateTrend(PrioSig, Time);
   end;
```

B. Server Side Procedures

On the receiving end, is the base station, running a server listening for registration and update requests sent by MH agents. The BS is responsible for maintaining an organized max heap structure of all currently registered MHs that fall under its zone of coverage. Once a registration or update packet is received, the following procedures take place to insure the careful addition/update of that MH's information into the BS's heap structure.

Step1: The BS would first check if the record for that MH is currently registered with it as a BS, if not the BS will retrieve that data from the last BS that MH was registered with and ask the previous BS to delete the entry of that MH in its heap. The BS will then send the MH its credentials to update its BS information and proceed to add the MH's record to its max heap structure.

Step2: If the MH's record already exists then the BS will consider the incoming data as an update and proceed to search and update its heap structure with the new data sent by the MH agent.

Step 3: The last case the server considers is when no previous BS record has been sent by the MH and no data for that MH has been found in the server's heap structure. In this case the server will assume that this is a first time sign on by a new client and proceed as outlined in step 1.

```
insertMaxHeap( heapStruct[], HostData Element ) :
   begin:
      if( Element.serverID neq serverID ) then
         migrateRecord ( Element.serverID,
Element.HostID );
      else if ( position ← searchForElement ( heapStruct[],
Element ) ≠ -1 ) then
         updateElement ( heapStruct[], Element,position
);
      else
         heapStruct[] ← push ( Element );
         buildHeap ( heapStruct [] );
         sortHeap ( heapStruct[], length[ heapStruct ] );
   end;
```

V. PERFORMANCE EVALUATION

The algorithm performance is measured in the standard Big O notation. Every essential part of the algorithm will be evaluated individually, and an aggregate performance measure will be derived from the parts. We first identify the following areas of the code that affect the performance of the algorithm, as presented in the previous section. We also distinguish between the preprocessing stage, which involves building and maintaining the heap structure, and the quorum selection process, which is a bounded function that depends on the size of the expected quorum. The amount of messages passed between the MH agent and the server are measured to insure that the least amount of needed messages are passed and to minimize network congestion problems that adversely affect the overall performance of the quorum process.

The performance of the heap building and sorting algorithms are well-known. Measures have been extensively detailed, so no formal proof will be given. Instead, only the final end result will be presented. Locating an element in the heap has linear performance $O(n)$. Although better search performance can be achieved using more efficient algorithms, it is not the focus of this work to tackle this issue, and is left as a future improvement. However, it is worth mentioning that *n* is bounded by the limited amount of hosts a BS can handle. As such we can represent the search procedure as $O(\max(MH))$ for a particular BS.

Taking the above performance measures into account, the following paragraph evaluates the performance of the main procedures that maintain and update the list of MHs currently registered with a particular BS.

Since this particular procedure involves code execution on two sites, a breakdown of the code execution on every site is measured, whereas network delays and other external factors have been ignored. The first step of the migration involves sending a request to a remote server, where a search and delete procedure is executed. The search procedure's performance is $O(n)$, with $O(1)$ performance for deletion once that record is found. When a record is received by the current server, the MH attempts to register using the combined insert and sort operations of the max heap structure. This adds a performance execution overhead of $O(n^2.\log^2(n))$, for a total combined worst case performance hit of $O(n^3.\log^2(n))$, which executes in polynomial time. Identical performance can be expected of the *insertMaxHeap* procedure in the case of an element update. Generalizing this to *m* sites, would yield a worst case global performance described in Expression (1).

$$\sum_{k=0}^{m} O(n^3.\log^2(n)) \qquad (1)$$

Ordinarily an insert operation on a heap data structure should be of the order $O(\log(n))$. However, due to the choice of the key (the Priority variable) chosen to sort the heap on, finding a host would require linear time instead,

based on the MH's identification string. This is one shortcoming that can be remedied in subsequent development of this architecture, and as mentioned prior, it is not the focus of this work to tackle all possible optimization aspects of the algorithm, but rather, to show that it is a viable architecture that can provide improved availability and reliability to epidemic quorum groups.

By selecting the best performing hosts from the max heap data structure to participate in an epidemic quorum, the BS insures that the probability of a process failure on the selected MH is kept to the minimum possible, based on the general state of the network. As such, going back to Eq. (1), we propose to see the effects of minimizing the process failure factor p_f on the overall availability of the epidemic quorum. Earlier, [1] sets p_f as a fixed value and measures availability based on the probabilities for vote repetitions and vote decisions. Whereas, the proposed mathematical model tackled here, measures how a variable p_f affects the overall availability for discrete values of *dec* and *rep*.

Appendix 1 tables and figures explain the mathematical results obtained, based on the mathematical framework presented in [1] for epidemic quorum availability measurement. The results also indicate that a minimum threshold should be respected when selecting MHs for quorums, below which, availability may suffer. This threshold would allow the MH selection procedure to set a cutoff point below which hosts would not be selected.

The results clearly show that based on the mathematical model presented in [1], the combination of a variable p_f, decision probability and repetition, although mainly affected by the probability of a quorum reaching a decision, clearly diminishes as the p_f factor increases. Selecting MHs with low probabilities of failures would ultimately lead to a far more stable and available quorum system. The probability for reaching a quorum decision on the other hand, is not directly related to p_f, but an indirect relation with the previous two factors may be inferred. Lesser available systems will most likely delay the outcome of the vote, if systems go off-line, forcing an increase in repetition cycles. Although the model does indicate that high availability is achieved, the convergence time to a consensus will increase with the amount of repetitions, leading to higher delays in up-to-date data acquisition. Table 4 shows the variance of p_f with high probability of repetition.

Trivially, with higher vote repetition, in the worst case scenario, the time t_{vote} it would take to create a quorum and reach a result can be expressed by equation (4):

$$t_{vote} = t_{create} + \sum_{n=0}^{\max(rep)} t_{repetition} \qquad (4)$$

where $t_{repetition}$ would vary with each repeat round, depending, among other factors, on network conditions as well. The repetition factor is not only calculated based on process failure, but also assumes the ability of a quorum to reach a decision based on the amount of information available to that quorum. The model in this thesis deals with the process failures due to network outages, rather than quorum failures, and as such, the repetition time and vote time factors expressed in Eq. (4) can be minimized by selecting highly reliable hosts, thus reducing both t_{create}, in the initial creation of the quorum, and t_{repeat} when subsequent quorum votes need to be carried.

Theoretically, the model provides insight into the availability of epidemic quorums, but given the difficulty to model real world network failures due to its mathematical complexity, only live system tests can verify the viability of such a model. The author is confident that by applying the presented theoretical model, this will unequivocally have a positive result on the availability of epidemic quorums. Since no live model was made available at the time of writing, a simulation was constructed to portray the availability of an epidemic quorum, by providing MH performance results with a discrete MH failure model.

VI. CONCLUSION

This work has presented an overview of the various mobile database models currently available, focusing mainly on the epidemic model, and more precisely on epidemic quorums. A novel approach to epidemic quorum selection based on an effort to minimize network disconnections, often experienced by wireless mobile hosts, was presented. The purpose of which, is to provide a more reliable quorum, with higher data availability. The classification of hosts according to measured and derived metrics, allows the model to be extended and incorporate other metrics which may be deemed important in later revisions, such as: database connection counts and performed transactions counts, to further refine the classification of mobile hosts. Although this primary study of the architecture was deemed satisfactory by its author, more work is needed to further refine the mathematical model and incorporate more complex network failure models that may further indicate the usefulness of this model. Work done in [6] and [7] can also be incorporated in the model to provide better quorum placement, minimizing network congestion and delays, instead of relying solely on traffic priority settings.

One of the main points to focus on in future studies on this topic, would include, building historical track record of mobile hosts based on more elaborate regression models (such as a Bayesian regression model), rather than the simplistic linear model used herein. This would require that real performance data be made available to the system in order to allow the Bayes engine's learning process to evaluate its current state, based on measurements that reflect the reality of the system. The author also recognizes that much improvement should also be done on the algorithm itself, allowing for much better performance, especially in the area of host lookup, where a hash lookup

table may be constructed in order to minimize lookup times. All in all the studies and results provided herein show how important a role the network environment plays in the performance of quorums in general and epidemic quorums in particular. Although this is still a very recent area of interest and study in mobility, it is however receiving just attention, especially in the wake of the wireless revolution, where reliable connectivity is considered whimsical in comparison to its wired counterparts.

REFERENCES

[1] Baretto, J., and Ferrero, P. (February 2007). The Availability and Performance of Epidemic Quorum Algorithms. INESC-ID Technical Report 10/2007.

[2] Bernard, G., Ben Othman, J., and Bouganim, L. (June 2004). Mobile Databases: A Selection of Open Issues and Research Directions. ACM SIGMOD record, 33(2) 78-83.

[3] Cormen, T., Leiserson, C. E., Rivest, R., L., and Stein, C., (2001). Introduction to Algorithms. Second Edition. Massachusets: MIT Press.

[4] Golovin, D., Gupta, A., Maggs, B.M., Oprea, and F., Reiter, M.K. (July 2005). Quorum Placement in Networks: Minimizing Network Congestion. Proceedings of the 24th Annual ACM Symposium on Principles of Distributed Computing (PODC'05), Denver, Colorado, USA, 16–25.

[5] Gray, J., Helland, P., O'Neil, P., and Shasha, D. (June 1996). The Dangers of Replication and a Solution. ACM SIGMOD Conference, Montreal, Canada.

[6] Greis, M. Tutorial. Retrieved from Marc Greis' Tutorial for the UCB/LBNL/VINT Network Simulator "ns" Web site: http://www.isi.edu/nsnam/ns/tutorial/index.html. 2007.

[7] Gupta, A., Maggs, B. M., Oprea, F., and Reiter, M. K. (July 2005). Quorum Placement in Networks to Minimize Access Delays. Proceedings of the 24th Annual ACM Symposium on Principles of Distributed Computing (PODC'05), Las Vegas, NV, USA, 87–96.

[8] Holliday, J., Agrawal, D., and El Abbadi, A. (July 2002). Disconnection Modes for Mobile Databases. Wireless Networks, 8(4) 391-402.

[9] Holliday, J., Steinke, R., Agrawal, D., and El Abbadi, A. (September 2003). Epidemic Algorithms for Replicated Databases. IEEE Transactions on Knowledge and Data Engineering, 15(5), 1218-1238.

[10] Kafri, N., and Janeček, J. (2002).Dynamic Behavior of the Distributed Tree Quorum Algorithm. Proceedings of the 22 International Conference on Distributed Computing Systems (ICDCS'02), Vienna, Austria, 517-524.

[11] Kumar, V. (2006). Mobile Database Systems. New Jersey: J. Wiley & Sons Inc.

[12] Kuruppilai, R., Dontamsetti, M., and J. Cosentino, F. (1997). Wireless PCS. New York: McGraw-Hill.

[13] Lee, M., and Helal, S. (2002). HiCoMo: High Commit Mobile Transactions. Kluwer Academic Publishers Distributed and Parallel Databases (DADPD), 11, 1.

[14] Madria, S. K., and Bhargava, B. (2001). A Transaction Model for Improving Data Availability in Mobile Computing. Kluwer Academic Publishers Distributed and Parallel Databases (DAPD), 10, 2.

[15] Mouly, M., and Pautet M. B. (1992). The GSM System for Mobile Communications. France: Cell and Sys Publications.

[16] Peleg, D., and Wool, A. (1995). The Availability of Quorum Systems. Information and Computation, 123(2), 210-223.

[17] Peysakhov, M., Dugan, C., Modi, P. J., and Regli, W. (May 2006). Quorum Sensing on Mobile Ad-Hoc Networks. Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi Agent systems (AAMAS'06), Hakodate, Japan, 1104-1106.

[18] Pitoura, E., and Bhargava, B., (1995). Maintaining Consistency of Data in Mobile Distributed Environments. Proceedings of 15th International Conference on Distributed Computing Systems.

[19] Pitoura, E., and Bhargava, B., (1999). Data Consistency in Intermittently Connected Distributed Systems. IEEE Transactions on Knowledge and Data Engineering (TKDE), 11, 6.

[20] Ross, K. A., and Wright, C. (1999). Discrete Mathematics Fourth Edition. New Jersey: Prentice Hall.

[21] Serrano-Alvarado, P. (2004). A Survey of Mobile Transactions. Distributed and Parallel Databases. 16, 193-230.

[22] VINT GROUP, from the Manual (formerly Notes and Documentation) Web site: http://www.isi.edu/nsnam/ns/doc/everything.html. 2007.

[23] Walborn, G. D., and Chrysanthis, P. K. (September 1995). Supporting Semantics-based Transaction Processing in Mobile Database Applications. Symposium on Reliable Distributed Computing Systems (SRDS), Bad Neuenahr, Germany.

[24] Walborn, G. D., and Chrysanthis, P. K., (March 1997). PRO-MOTION: Management of Mobile Transactions. ACM Symposium on Applied Computing, San Jose, USA.

[25] Walborn, G. D., and Chrysanthis, P. K., (February 1999). Transaction Processing in PRO-MOTION. ACM Symposium on Applied Computing, San Jose, USA.

Samer Younes a senior web developer and systems integrator at Solidere, one of the biggest real estate companies in the Middle East. His main role is developing a new web presence for the company as well as designing and implementing intranet web platforms and solutions for the company's use. Prior to this position he worked as a system administrator and application developer for the Lebanese American University where he also taught as a part time instructor. In his spare time Samer cultivates his avid interest in the Open Source movement and the Linux platform with particular interest in emerging web technologies and virtualization frameworks.

Ramzi A. Haraty is an associate professor of Computer Science and the assistant dean of the School of Arts and Sciences at the Lebanese American University in Beirut, Lebanon. He received his B.S. and M.S. degrees in Computer Science from Minnesota State University - Mankato, Minnesota, and his Ph.D. in Computer Science from North Dakota State University - Fargo, North Dakota. His research interests include database management systems, artificial intelligence, and multilevel secure systems engineering. He has well over 100 journal and conference paper publications. He is a member of the International Society for Computers and Their Applications.

APPENDIX A.

TABLE 1. AVAILABILITY CHART WITH DEC=0.6 AND REP=0

Dec	Rep
1	0
Pf	Sum(Pf)
0	1.0000000000
0.1	0.9999999999
0.2	0.9999998976
0.3	0.9999940951
0.4	0.9998951424
0.5	0.9990234375
0.6	0.9939533824
0.7	0.9717524751
0.8	0.8926258176
0.9	0.6513215599
1	0.0000000000

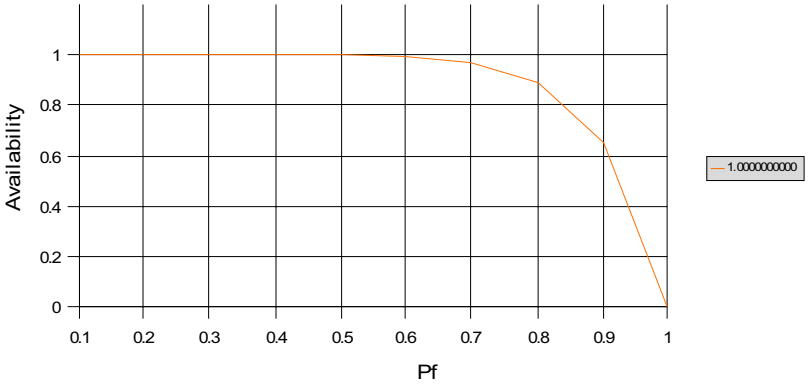

TABLE 2. AVAILABILITY CHART WITH DEC=0.25 AND REP=0

Dec	Rep
0.6	0
Pf	Sum(Pf)
0	0.6000000000
0.1	0.5999999999
0.2	0.5999999386
0.3	0.5999964571
0.4	0.5999370854
0.5	0.5994140625
0.6	0.5963720294
0.7	0.5830514851
0.8	0.5355754906
0.9	0.3907929359
1	0.0000000000

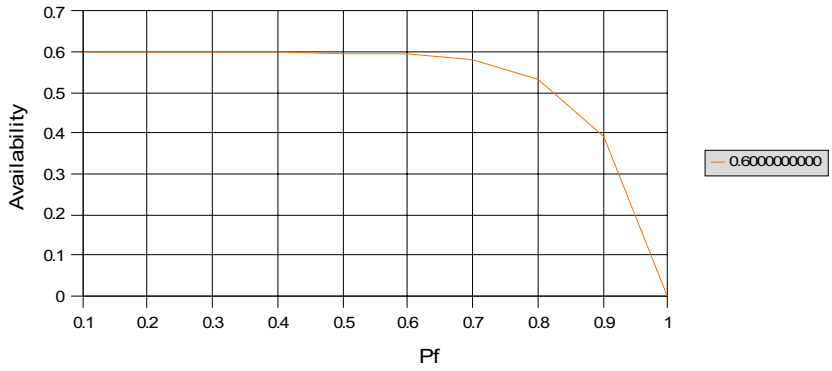

TABLE 3. AVAILABILITY CHART WITH DEC=0.25 AND REP=0

Dec	Rep
0.25	0
Pf	Sum(Pf)
0	0.2500000000
0.1	0.2500000000
0.2	0.2499999744
0.3	0.2499985238
0.4	0.2499737856
0.5	0.2497558594
0.6	0.2484883456
0.7	0.2429381188
0.8	0.2231564544
0.9	0.1628303900
1	0.0000000000

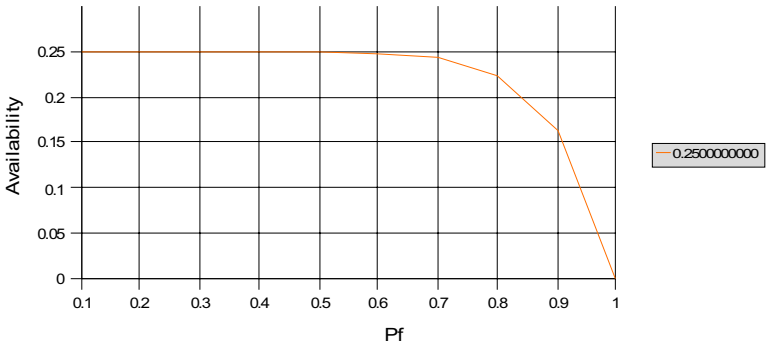

TABLE 4. AVAILABILITY CHART WITH DEC=0.25 AND REP=0.65

Dec	Rep
0.25	0.65
Pf	Sum(Pf)
0	0.7142857143
0.1	0.7142857142
0.2	0.7142856411
0.3	0.7142814965
0.4	0.7142108160
0.5	0.7135881696
0.6	0.7099667017
0.7	0.6941089108
0.8	0.6375898697
0.9	0.4652296856
1	0.0000000000