An Improved Quorum Selection Algorithm

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

1. 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.

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 an iPod.

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 provides a background. Section 3 presents the epidemic quorum algorithms. Section 4 discusses the improved quorum selection algorithm architecture. Section 5 concentrates on the algorithms used in our approach. Section 6 provides the performance evaluation and section 7 presents the conclusion.

2. Background

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 remains a challenge 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 [7] provide an excellent overview of the various mobile database models, the most popular of which, are presented in [7] and Kumar [5].

With a comprehension of the workings of these various models, recent publications by Holliday et al [3], [4] and Baretto Ferrero [1], seem to agree that quorum systems are the best suited for a mobile environment.

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, [6] by Peysakhov et al. that provides a general quorum availability

978-0-7695-3190-8/08 $25.00 © 2008 IEEE
DOI 10.1109/CTRQ.2008.14

evaluation and Baretto Ferrero [1] that deals more specifically with the performance evaluation of epidemic quorum algorithms.

Other works were also studied, pertaining to quorum placement and congestion management, but the findings, although very interesting, were left as future improvements.

3. Epidemic Quorum Algorithms

Epidemic Quorum Algorithms (which will be referred to as eQuorums from hereon), are a particular breed of epidemic algorithms with some very 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. Transactions in eQuorums are serialized in a causal fashion, so that one and only one transaction will commit through consensus. Vote results are stored in a log and then propagated to other sites through eQuorum messages until all sites have received the vote results. When a particular site receives a positive vote for a particular transaction, it automatically commits the data for that transaction aborting all other conflicting transactions at that site.

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. 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. 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:

- $rep_\varepsilon(n) + dec_\varepsilon(n) \leq 1$
- 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 in (1):

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

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. The goal is to minimize the probability of failure p_f to maximize availability.

Sensing and incorporation of network states in quorums is a very recent topic of discussion, studied, most notably, by Peysakhov et al. [6] and Gupta et al. [2]. However, instead of using the standard client/server model, [6] uses migrating agents applied to standard quorums. As for the metrics evaluated in [6], 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.

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

and

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

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, defined as values lying outside of C, is reached. Although [6]'s method enhances the general confidence in a quorum, the presented approach deals only with the quality of the host as measured prior to quorum selection. Furthermore, an agent

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

4. 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.
- Host Priority: Depending on the signal strength of the MH, a priority number is given directly proportional to the signal strength.
- Host Trend: Defined as a derived metric based on the performance of a MH's signal strength with time. The trend is calculated using a standard weighted linear regression.

The BS, once it receives this data from a MH, will classify it in a heap structure that allows picking the best performing MHs when a quorum is being built. The heap structure on the BS is constantly maintained as a max heap. The max heap keeps a real-time record of all registered MHs ordered on the metrics mentioned prior, from most reliable to least reliable.

The architecture also introduces the means to migrate MHs from one BS to another. Since the MH is essentially a mobile unit, the 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 cell he/she's currently in.

5. 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. Figure 1 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.

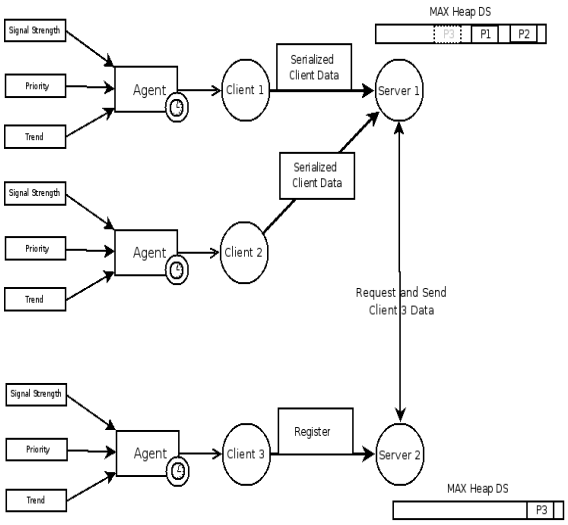


Figure 1. IQSA high level modules interaction

5.1 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 notable, are the Time vector structure and the PrioSig vector structure. The Time structure holds a timestamp for every measure of both the SignalStrength and Priority performed by the agent, at specific intervals metrics. 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 or to default values. The trend metric is initially null due to the unavailability of data to perform regression calculation. 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 be controlled by assigning to it a variable weight factor. This calculation is performed every time the client agent gets an updated measure on the basic metrics: signal strength and priority.

5.2 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. The following procedures take place to insure proper information addition/update.

Step1: The BS would first check if the record for that MH is registered with it, if not it will retrieve that data from the last BS that MH was registered with and deleting the entry of that MH from the previous BS's heap. The BS will then 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 for a newly connected MH. In this case the server will determine that this is a first time sign on by a new client and proceed as outlined in step 1.

6. Performance Evaluation

The algorithm performance is measured in the standard Big O notation. 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.

The performance of the heap building and sorting algorithms are well-known. 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. 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.

Evaluating the performance of the main procedures involves a breakdown of the execution on every site, where 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 $O(n^3.\log^2(n))$. Generalizing this to *m* sites, would yield a worst case global performance described in Expression (1).

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

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.

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 on the overall availability through a variable p_f 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. 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 3 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):

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

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 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 reliable hosts, reducing both t_{create}, and t_{repeat}.

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.

7. 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. Work done in [2] 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 of future studies on this topic, would include, building historical track record of mobile hosts based on more elaborate regression models, 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.

References

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

[2] 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.

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

[4] 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.

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

[6] 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.

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

Appendix 1

Table 1 Availability Chart with dec=1 & ep=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

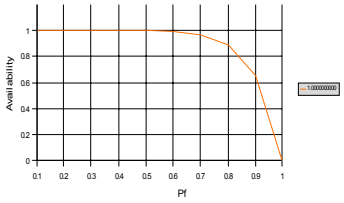


Table 2 Availability Chart with dec=0.25 & ep=0

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

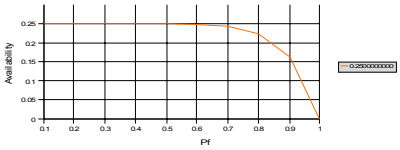
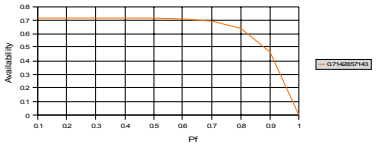


Table 3 Availability Chart with dec=0.25 & rep=0.65

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



Acknowledgements:

This work was funded by a grant (grant number: URC-T-2008-06) from the Lebanese American University.