World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:4, No:1, 2010

A Delay-Tolerant Distributed Query Processing Architecture for Mobile Environment

T.P. Andamuthu and Dr. P. Balasubramanie

***Abstract*—**The intermittent connectivity modifies the "always on" network assumption made by all the distributed query processing systems. In modern- day systems, the absence of network connectivity is considered as a fault. Since the last upload, it might not be feasible to transmit all the data accumulated right away over the available connection. It is possible that vital information may be delayed excessively when the less important information takes place of the vital information. Owing to the restricted and uneven bandwidth, it is vital that the mobile nodes make the most advantageous use of the connectivity when it arrives. Hence, in order to select the data that needs to be transmitted first, some sort of data prioritization is essential. A continuous query processing system for intermittently connected mobile networks that comprises of a delay-tolerant continuous query processor distributed across the mobile hosts has been proposed in this paper. In addition, a mechanism for prioritizing query results has been designed that guarantees enhanced accuracy and reduced delay. It is illustrated that our architecture reduces the client power consumption, increases query efficiency by the extensive simulation results.

***Keywords*—**Broadcast, Location, Mobile host, Mobility, Query.

I. Introduction

CURRENTLY, one of the most important and attractive research issues in computer networks is the development of ubiquitously-accessed networks, where users can access computer resources at anytime from anywhere through movable computers. A substantial amount of research has already been conducted on how to support mobile communications in existing network environments. These approaches were directed at the realization of low cost location management of mobile hosts. In addition to the packet transmission required in conventional mobile communication protocols, the mobile computing environments currently under development must resolve several data management issues such as query processing [1].

Nodes (clients and servers) may not remain connected to the MANET throughout their life. To be connected to the network, a node must be able to hear the transmission of at least one other node on the network and have sufficient power to function. Network nodes (clients and servers) may operate in any of the three modes that are designed to facilitate the reduction in power used.

T.P. Andamuthu, Assistant Professor, Department of CSE, Maharaja Engineering College, Coimbatore, Tamilnadu, ph: +91 9842286262, E-mail: andamuthu@rediffmail.com, andamuthu_me@yahoo.co.in
Dr. P. Balasubramanie, Professor, Department of CSE, Kongu Engineering College, Perundurai, Tamilnadu.

These are: transmit – this mode uses the most power, allowing transmission and reception of messages, receive – this mode allows the processing of data and reception of transmissions, and standby – in this mode, the CPU does no processing, transmitting or receiving.

In many existing and emerging application domains the downstream communication capacity from servers to clients is much greater than the upstream communication capacity from clients back to servers.

The emergence of new lightweight web technologies and the development of mobile devices leads to a situation, where the users can consume the same web services regardless of the place, time and device. Mobile devices are equipped with networking capabilities and sensors that provide versatile context and user-community information. This information enhances the user experience as it can be used to compensate the limited means of input.

In our previous work [2], we have proposed an efficient architecture for query processing in mobile environment. The proposed architecture utilizes maximum of downstream communication capacity of server to provide the client with the needed information with minimum need for data pull, because pull-based systems are a poor match for asymmetric communications environments, as they require substantial upstream communications capabilities. The proposed approach allows mobile clients to maintain a service execution closest to their location context and to thematically locate the mobile. The paper also contains an Adaptive Updating Algorithm for handling host mobility.

The intermittent connectivity modifies the "always on" network assumption made by all the distributed query processing systems. In modern- day systems, the absence of network connectivity is considered as a fault. Since the last upload, it might not be feasible to transmit all the data accumulated right away over the available connection. It is possible that vital information may be delayed excessively when the less important information takes place of the vital information. Owing to the restricted and uneven bandwidth, it is vital that the mobile nodes make the most advantageous use of the connectivity when it arrives. Hence, in order to select the data that needs to be transmitted first, some sort of data prioritization is essential.

A continuous query processing system for intermittently connected mobile networks that comprises of a delay-tolerant continuous query processor distributed across the mobile hosts has been proposed in this paper. In addition, a mechanism for prioritizing query results has been designed that guarantees

enhanced accuracy and reduced delay. The conventional continuous query processors send the results of continuous queries instantaneously over the network, whereas the proposed query processor stores them in an output buffer.

The paper is organized as follows. Section II presents the related work. Section III presents the model for query processing. In Section IV, we describe our proposed architecture. Simulation results are presented in Section V and the conclusion is given in section VI.

II. Related Work

Dorian C. Arnold Barton P. Miller [3] have proposed a scalable failure recovery model for data aggregations in large scale tree-based overlay networks (TBONs). They have formalized the TBON model and its fundamental properties to prove that our state compensation model properly preserves computational semantics across TBON process failures.

Huilong Huang et al. [4] have proposed CNFS, an algorithm for efficient and robust query processing for mobile wireless sensor networks. CNFS is a walk–based algorithm that is biased to visit nodes close to the source first. This bias is accomplished by collecting topology information about the network as the search progresses.

Dragan Stojanovic et al. [5] address the problem of processing continuous range queries over mobile objects, whose motion is constrained by a spatial network. The query range represents the user-selected area, the map window, the polygonal feature, or the area specified by the distance from a reference point of interest. In contrast to regular queries that are evaluated only once, a continuous query remains active over a period of time. A major challenge for this problem is how to provide efficient processing of continuous queries with respect of CPU time, I/O time and main memory utilization.

Wei-Shinn Ku et al. [6] proposed two novel algorithms for processing k nearest neighbor queries and range queries on spatial networks with privacy protection. The main idea is to hide the exact mobile user location with a cloaked region. The cloaked region covers the query requester and at least $K - 1$ other users based on the K-anonymity concept. The spatial queries are executed based on both the cloaked region and the underlying networks.

Giuseppe Amato et al. [7] proposed approach the WSN can be programmed using a query language (MW-SQL), which offers constructs, specialized for sensor networks. The query language is offered by a JDBC driver which is encapsulated within the OSGi framework.

Wen-Chih Peng et al. [8] explored three asymmetric features of a mobile environment. Then, in light of these features, we devised query processing methods for both join and query processing. Explicitly, according to those asymmetric features of a mobile computing system, we examined three different join methods and devised some specific criteria to identify MI/SI profitable semi joins.

J. M. Hellerstein et al. [9] attempting to architect and implement a continuously adaptive query engine suitable for global-area systems, massive parallelism, and sensor networks. To set the stage for our research, we present a survey of prior work on adaptive query processing, focusing on three characterizations of adaptivity: the frequency of adaptivity, the effects of adaptivity, and the extent of adaptivity.

A. Y. Seydim et al. [10] proposed a general view of location relatedness in the queries. We distinguish between location awareness and location dependence and provide a formalization of the approach.

F. Perich et al. [11] described new challenges that this scenario presents to the distributed database framework, and present the design of a framework for serendipitous querying and query response in an ad-hoc mobile environment.

R. Avnur et al. [12] proposed a query processing mechanism called an eddy, which continuously reorders operators in a query plan as it runs. It characterize the moments of symmetry during which pipelined joins can be easily reordered, and the synchronization barriers that require inputs from different sources to be coordinated. By combining eddies with appropriate join algorithms, we merge the optimization and execution phases of query processing, allowing each tuple to have a flexible ordering of the query operators.

Jim Smith and Paul Watson [13] have described how a publicly available distributed query processing system for the grid can be enhanced to support fault-tolerance. It describes the implementation of a rollback-recovery protocol and presents measurements of the cost of both protocol overheads and recovery. These results shows that the implementation can exhibit low overhead and can yield significant performance improvements through recovering and continuing after failure.

Jeong-Hyon Hwang et al. [14] have studied various recovery guarantees and pertinent recovery techniques that can meet the correctness and performance requirements of stream-processing applications. They have discussed the design and algorithmic challenges associated with the proposed recovery techniques and described how each can provide different guarantees with proper combinations of redundant processing, check pointing, and remote logging. They have analyzed how the knowledge of query network properties can help decrease the cost of high availability.

III. Query Processing For Location Sensitive Queries

A. Model

A mobile host, which can move across cells, can keep pieces of information that may be accessed by other systems. A client is a system, which invokes queries about mobile hosts. A query server (QS) is a system, which can handle mobile query processing according to the requests from clients. Each client is directly or indirectly connected to at least one query server through a global network. A mobile host server (simply, server hereafter) is a system, which can directly communicate with all mobile hosts in the same cell through its wireless interface, and furthermore can communicate with all query servers through global networks. In Fig 1, we show an example of a network, which includes clients, query servers, mobile host servers, and mobile hosts.

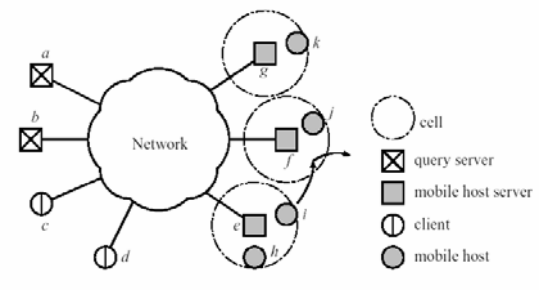

Fig. 1 An example of a mobile database network.

The following types of location sensitive queries are considered for mobile hosts.

1. Location (L) query: A query to obtain the location of a mobile host. Note that an L query is not concerned with whether the mobile host is active or not.

2. Data (D) query: A query to obtain a piece of information from a mobile host.

Now, we propose an fault-tolerant architecture for handling the queries.

IV. Proposed Fault-Tolerant Query Processing Architecture

A. Fault-Tolerant Continuous Query Processing

It is vital that mobile nodes make the best possible utilization of connectivity when it arrives owing to the limited and variable bandwidth. Accordingly, in order to permit the nodes to choose what the data to be transmitted first, some sort of data prioritization is necessary. In this paper, a simple buffering mechanism and a protocol to manage the staging and delivery of query results from mobile hosts to the clients and of queries from the clients to the mobile hosts have been proposed. Even in presence of poor or no connectivity, a local query processor in the proposed scheme collects stores and processes the data constantly. When connectivity resumes, in accordance to the query priority, the matched data as specified by the query is sent. Therefore, queries are continuous. Nevertheless, intermediate results are stored locally, from which the results of the continuous queries are streamed.

The mobile host server runs a delay-tolerant distributed query processor (DDQP). Clients use their query server (QS) to express declarative queries. These queries are distributed by FDQP to the "local" query processors running on the mobile hosts in such a manner that the same queries are shared by all the nodes. The required data is gathered and locally processed by the nodes and when the network connectivity is available, it is delivered to the server. In these opportunistic connections, queries and control messages too flow from the server to the mobile hosts. Then the continuous and snapshot queries sent from the server are executed over their database.

The main difference between DDQP and traditional continuous query processors is that the conventional continuous query processors send the results of continuous queries instantaneously over the network, whereas the DDQP stores them in an output buffer. The size of the node's durable storage restricts the total size of the output buffer. The subsequent section describes the prioritization of queries and the policy used to remove the oldest data from the lowest prioritized buffers initially. A network layer tuned for intermittent and short duration wireless connections is employed to drain the buffers. In accordance with the name of the source query and the client node ID, the rows are partitioned into tables as the results reach the server. A buffer name is added to the continuous queries by the clients to populate these result buffers and to specify a named output buffer. As and when the result rows from the continuous and ad hoc queries into the output buffer, they are positioned into separate named buffers.

B. Prioritizing Query Results

It is crucial that mobile nodes make the best possible utilization of connectivity when it arrives owing to the limited and variable bandwidth. Accordingly, in order to permit the nodes to choose what the data to be transmitted first, some sort of data prioritization is necessary.

Every node generates additional rows compared to the number of rows it can transmit to the server at any give time. The mobile host can choose the order in which it should transmit (when the connectivity is available) the data from the currently available results instead of simply transmitting the generated queries in the original order. This assists the fetching of the most relevant results from the buffer to the server and thus the priorities of the old results are reduced.

A collective score is assigned to each query in its output buffer every time a continuous query enqueues results for delivery. Ultimately, the results are classified in the ascending order of the score. All the pending higher-priority query results are delivered before delivering all the lower priority results. In case a conflict arises for queries running on equal priority levels, the results are delivered on basis of the collective weight value. The distance from the client and the downstream bandwidth required to transmit the results to the client are employed to calculate the collective weight value. Obviously, the queries with minimum weight values are served initially.

The least prioritized buffers are removed to avoid the buffer over flow.

(i) Let $R_i, i = 1,2,...n$ be the results of n queries which are enqueued in the respective named output buffers.

(ii) Let $\text{Pri}, i = 1,2,...n$ be the priorities of these queries specified by the clients at the time of submitting their queries.

(iii) Sort Pri in the descending order.

(iv) Then server transmits the results according to the sorted order of Pri .

(v) If $\text{Prj} = \text{Prj} + 1$, for any $j < n$, (i.e. two queries of same priority) then

(vi) Find the weight Wi of Qi such that $Wi = Dti + Bwi$

Where Dti is the distance between the mobile host and the client (which can be determined using the client pull stage [2]) and Bwi is the required downstream bandwidth from the server to the client.

(vii) Then server transmits the results in the ascending order of *Wi* .

In case of Intra-Query Prioritization, when results enqueued by continuous query enqueues for delivery, DDQP assigns a score to every row in a query's output buffer. In order to permit the previously scored rows to be rescored, DDQP delivers results in ascending score order. Usually, in order to deliver the rows in FIFO order, they are scored according to their insertion time.

V. Experimental Results

This section deals with the experimental performance evaluation of our FDQP architecture through simulations. In order to test our protocol, The NS2 simulation software [15] is used. NS2 is a general-purpose simulation tool that provides discrete event simulation of user defined networks.

The network nodes were placed uniformly at random within a square of 1000 meters. We varied the average speed of the mobiles from 10, 20, …, 50 in order to study the impact of server mobility. The data broadcast size is varied from 1000, 2000… 5000.

In all the experiments, we used the following evaluation criteria:

A. Simulation Parameters

Average Power Consumption The average power consumed by clients and the average power consumed by servers are calculated.

Query Efficiency The data pull section will rely on the measurement of query efficiency. This is a measure of the percentage of data queries that get served during an entire simulation.

Delivery Ratio which is the ratio of results received by the clients and the no. of queries sent.

B. Simulation Results

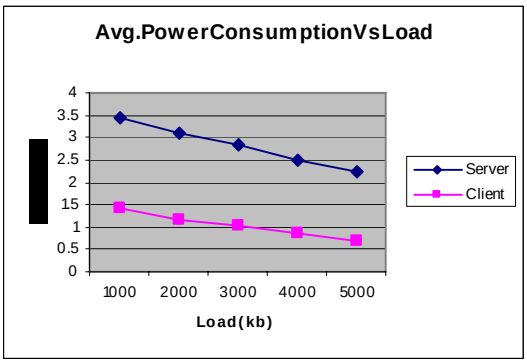

Fig.2 Avg.PowerConsumption Vs Load

In Fig2, when the broadcast size or server load increases, the average power consumption for server decreases as less time is spent for transmitting. The average power consumption for clients is universally low, because of the high level of disconnection due to the nodes movement.

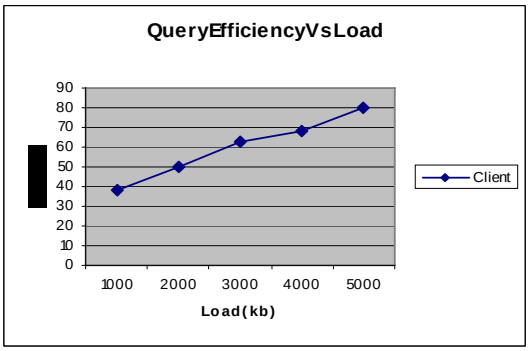

Fig.3 Query Efficiency Vs Load

From Fig.3, we observe that the query efficiency increases when the broadcast size increases.

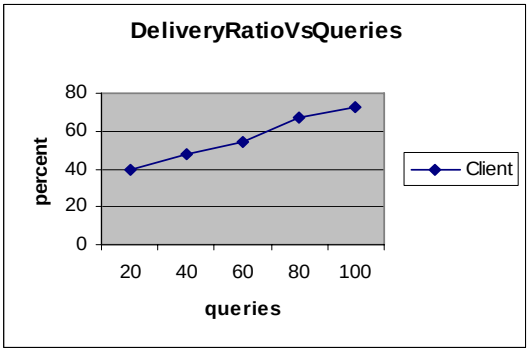

Fig.4 Delivery Ratio Vs Queries

In Fig.4, the delivery ratio increases significantly when the number of queries is increased.

VI. Conclusion

In this paper, an efficient query processing system for intermittently connected mobile networks that comprises of a delay-tolerant continuous query processor distributed across the mobile hosts has been proposed. In addition, a mechanism for prioritizing query results has been designed such that enhanced accuracy and reduced delay is ensured.

Even in presence of poor or no connectivity, a local query processor in the proposed scheme collects stores and processes the data constantly. When connectivity resumes, in accordance to the query priority, the matched data as specified by the query is sent. Intermediate results are stored locally, from which the results of the continuous queries are streamed. The conventional continuous query processors send the results of continuous queries instantaneously over the network, whereas the proposed query processor stores them in an output buffer. It is illustrated that our architecture reduces the client power consumption, increases query efficiency by the extensive simulation results.

References

[1] M. Tsukamoto et al. "Strategies for query processing in mobile computing," in Mobile Computing, ed. T. Imielinski, and H.F. Korth, pp.595–620, Kluwer Academic Publishers, 1996.

[2] T.P.Andamuthu and Dr.P.Balasubramanie "An Efficient Architecture for Query Processing in Mobile Environment", *IJCSNS International*

Journal of Computer Science and Network Security, VOL.8 No.9, September 2008.
[3] Dorian C. Arnold Barton P. Miller "A Scalable Failure Recovery Model for Tree-based Overlay Networks", 2007 *ACM*.
[4] Huilong Huang et.al. "Efficient and Robust Query Processing for Mobile Wireless Sensor Networks".
[5] D. Stojanović et al. "Continuous Range Query Processing for Network Constrained Mobile Objects", *Proceedings of the 8th International Conference on Enterprise Information Systems (ICEIS 2006)*, May 24-27, Paphos, Cyprus, 2006, pp. 63-70.
[6] Wei-Shinn Ku "Privacy Protected Query Processing on Spatial Networks", 2007 IEEE 23rd International Conference on Data Engineering Workshop, April 2007.
[7] Giuseppe Amato et al. "Enabling Context Awareness through Distributed Query Processing in Wireless Sensor Networks", *2nd International Workshop on Requirements and Solutions for Pervasive Software infrastructures*, September 16, 2007, Innsbruck, Austria.
[8] Wen-Chih Peng et al. "Query Processing in a Mobile Computing Environment: Exploiting the Features of Asymmetry" *IEEE Transactions on Knowledge and Data Engineering,* Publication Date: July 2005.
[9] J. M. Hellerstein et al. "Adaptive query processing: Technology in evolution" *IEEE Data Engineering* Bulletin, pages 23(2):7–18, 2000.
[10] A. Y. Seydim et al. "Location dependent query processing". In *MobiDE*, pages 47–53, 2001.
[11] F. Perich et al. "Query routing and processing in mobile ad-hoc environments" Technical report, UMBC, November 2001.
[12] R. Avnur et al. "Eddies: continuously adaptive query processing" SIGMOD Rec., 29(2): 261–272, 2000.
[13] Jim Smith, Paul Watsonm "Fault-Tolerance in Distributed Query Processing", 18th February 2005.
[14] J. Hwang, M. Balazinska, A. Rasin, U. Cetintemel, M. Stonebraker, and S. Zdonik. High-availability algorithms for distributed stream processing. In Proc*.21st International Conference on Data Engineering (ICDE)*, 2005.
[15] http://www.isi.edu/nsnam/ns/

T.P.Andamuthu received the B.E. degree in Electronics and Communication Engg in 1996 and M.E. degree in Computer Science and Engineering in 2002 from Bharathiar University, Coimabatore, India. Now he is working as a assistant professor in the department of computer science and Engineeringing in Maharaja Engineering College, Coimbatore, Tamilnadu. He is currently doing is Ph.D degree in the field of mobile computing in Anna University, Chennai, India. His area of interest are Mobile computing, operating systems and Computer Networks. He has presented many papers in national Conferences in various fields.

Dr.P.Balasubramanie has obtained his Ph.D degree in theoretical computer science in the year 1996 from Anna University, Chennai. He was awarded junior research fellow by CSIR in the year 1990. Currently he is a professor in the department of Computer Science and Engineering, Kongu Engineering College, Perundurai, Tamilnadu. He has published more than 50 research articles in International/National Journals. He has also authored six books. He has guided 3 Ph.D scholars and guiding 15 research scholars. His area of interest include theoretical computer science, data mining, image processing and optimization Techniques.