A COMPARATIVE STUDY OF REPLACEMENT ALGORITHMS USED IN THE SCALABLE ASYNCHRONOUS CACHE CONSISTENCY SCHEME

Ramzi A. Haraty and Lana Turk
Lebanese American University
P.O. Box 13-5053 Chouran
Beirut, Lebanon 1102-2801
Email: {rharaty, lana.turk}@lau.edu.lb

Abstract

The technology of PCs has been in progress in a fast rate for many years. Mobile computing is one of the technologies brought into the area of computers. Different problems have arisen from the narrow bandwidth and limited battery power of mobile clients. Therefore, algorithms have been proposed to provide cache consistency in mobile databases by using cache invalidation strategies. Scalable Asynchronous Cache Consistency Scheme (SACCS), a highly scalable, efficient and low complexity algorithm, is one of the cache consistency maintenance algorithms proposed. It counts on invalidation reports to maintain cache consistency between server databases and mobile user databases. Least-Recently-Used (LRU) is used as a cache replacement algorithm in SACCS. In this work, different cache replacement strategies are proposed to be applied in SACCS: MRU (Most-Recently-Used), MFU (Most-Frequently-Used), LFU (Least-Frequently-Used), FIFO (First-In-First-Out). A simulation of SACCS with these cache replacement algorithms is done and produces results that will be compared to show the variation of the performance of the system. Statistical results will point out the advantages and disadvantages of each algorithm concerning miss ratio, delay, total hit, and total miss.

KEY WORDS: Cache consistency, invalidation strategies, and mobile databases.

1. INTRODUCTION

In the past few years, mobile computing has been used profusely in computer technology. Many problems arise from this computer field because the wireless connection is expensive and the bandwidth is limited. To save time in exchanging data between base stations and mobile users and improve the performance of the system, data caching is imposed on mobile users in order to access the data recently used or most likely to be used in the future. To maintain cache consistency between mobile users and base stations, two approaches have been proposed: the stateful and the stateless approach.

A cache consistency maintenance algorithm called Scalable Asynchronous Cache Consistency Scheme (SACCS), gathering both stateless and stateful approaches, was proposed with the Least-Recently-Used (LRU) replacement algorithm [9]. In this work, different replacement algorithms are used instead of the LRU in the SACCS, producing different results concerning the system performance. A comparison is made between the LRU and the replacement algorithms used to show the advantages and disadvantages of each of these algorithms in system performance.

Mobile computing encounters numerous problems regarding wireless connection and limited bandwidth. Solutions have been proposed to such problems like data caching and invalidation reports. These methods have been used in the cache consistency maintenance algorithm SACCS, which maintains the cache consistency between the mobile users and the servers by gathering the features of both the stateful and stateless approaches. SACCS uses the cache replacement algorithm LRU as a base replacement algorithm in the mobile user caches.

In this work, the LRU is replaced by different cache replacement algorithms: LFU (Least-Frequently-Used), MRU (Most-Recently-Used), MFU (Most-Frequently-Used), and FIFO (First-In-First-Out). The purpose of this thesis is to show and compare between the results of these cache replacement algorithms used in the SACCS (including the LRU already used). The comparison between the different outputs of the SACCS algorithm will be built on experimental and statistical results to show the advantages and disadvantages of each cache replacement algorithm in different characteristics of the system performance, for example the miss ratio and the delay in data access.

The remainder of this paper is organized as follows: section 2 is a literature review of related techniques and approaches used in the field of mobile computing. Section 3 describes the cache consistency maintenance scheme – SACCS. Section 4 presents the experimental and statistical results using different replacement algorithms. And section 5 presents the conclusion.

2. LITERATURE REVIEW

Computer interconnection through wireless networks has been increasing at a high speed in mobile environments. Data is broadcasted repetitively from the server to the clients without the need for a client request. Thus, the server is not informed before a mobile transaction accesses a data object because the

data is accessed by the clients while it is being broadcasted. A distributed database system faces many challenges concerning data consistency, especially with a large number of mobile clients.

The server may broadcast data items during update transactions in its database. Thus, mobile clients observe data inconsistency through their transactions. During the past few years, different studies have been made on preserving data consistency in mobile distributed database systems [1][3][5][6][7][8][10]. This section concentrates on describing a number of cache invalidation strategies proposed to solve efficiently the problem of data inconsistency. These strategies lead to a better system performance by reducing bandwidth utilization and query latency, saving power and energy for mobile clients.

2.1 Cache Invalidation Model

The dynamic progress of mobile computing has led to a major concern regarding data consistency. Mobile clients want to connect to the server, send queries and receive data at different periods of time but they are facing different limitations, such as the limited battery power of the clients, and the narrow bandwidth of the network. In addition, the mobility and frequent disconnections of the mobile clients cause the data cached in the client side to become invalid. To solve this problem, different cache invalidation mechanisms have been proposed.

The IR-based cache invalidation approach was used to preserve cache consistency. In this approach, invalidation reports (IRs) are periodically broadcasted by the server to the clients. When a client receives a request for some data, it waits for the next IR to invalidate or validate the data in its cache accordingly. If the data is invalid, the client sends a request to the server for a valid copy. Otherwise, the client returns the requested data directly.

The latency for any client to answer a query depends on the length of the IR interval; as the IR interval gets longer, the answer to the query gets delayed. A solution to that delay was to replicate the IR m times within the IR interval [11]. The replicated IRs called updated invalidation reports (UIRs) contain only the data items updated after the broadcast of the last IR. In this way, the client should wait the maximum of 1/m of the IR interval to answer a query for any data. This proposition reduced the query latency.

2.2 Counter-based Cache Invalidation Algorithm

The counter-based cache invalidation approach is designed to reduce the query latency and the congestion on the bandwidth by following three schemes: First, prefetching data that is likely to be used in the future by the clients makes a better utilization of the broadcast bandwidth. In this case, when the server broadcasts data items, the clients download the data that is invalid in their caches and do not have to send additional requests to the server for valid data. Second, using a broadcast list saves power and bandwidth. The server broadcasts only the data that has been updated since the last IR (invalidation report). After the server broadcasts an IR, it broadcasts the id list of data being updated (Lbcast). Then it broadcasts the data itself. The clients, on their side, save the broadcast list and wait until the data arrives to wake up and download it. Third, relying on counters also helps in saving bandwidth and in identifying the frequently accessed data to be broadcasted. The broadcast list includes only the ids of the data most frequently accessed by the clients. To identify the data to be included in the id list, a counter is associated with each data item. This counter is incremented by one on every request of the data from the server and decremented by one if the data has been discarded by the client. The data with a counter that is equal to 0 is not included in the IR, unlike the IR-based approach, which includes all updated data in the IR. Hence, this counter-based scheme saves the broadcast bandwidth.

2.3 Cache Invalidation Strategy

In previous years, different cache invalidation strategies existed, such as the bit-sequence (BT), the timestamp (TS), the dual-report (DRCI) and others, to maintain data consistency between data items in the server database and data items in the mobile client caches.

A new cache invalidation strategy is proposed to reduce the invalidation report sizes and invalidate the necessary data items in the mobile client caches [2]. In this strategy, when the user sends a query to the mobile client, the queried data will be checked for its validity; if the data exists in the client cache and is valid, it will be used by the user. On the other hand, if the data does not exist in the client cache, it will be requested from the server. If the client is disconnected for a period of time and reconnected again, the data in its cache will be in an uncertain state. In this case, if data is requested by a user and exists in the client cache, the client should check for the validity of all the data in its cache by sending the server the value of the time when the last invalidation report was received by the client (T_{lb}) and the requested data. Then, the server broadcasts to the client the invalidation report containing only the ids of the updated data since T_{lb} (including the last updated timestamp T) and the requested data. Thus, the client answers the user's request and replaces its T_{lb} by T if T_{lb} is less than T. In addition, the other mobile clients, receiving the IR, will update their T_{lb} if its value is less than T.

As a result, the IRs broadcasted by the server will be smaller in size consisting only of the ids of the invalidated data since the T_{lb} and the data in the mobile client caches would not be invalidated unnecessarily.

Thus, this proposed cache invalidation strategy leads to better bandwidth utilization and less cache requests to the server.

2.4 Cache Consistency Strategy in a Disconnected Distributed Environment

As mentioned earlier, the mobile computing environment is prone to frequent disconnections of mobile clients or mobile hosts (MHs) from the server caused by low battery power. These disconnections make the cache consistency in the mobile hosts difficult to maintain. Several cache consistency schemes were proposed to solve this problem. Most of these schemes use the invalidation report approach (call-back mechanism).

An asynchronous invalidation reports scheme is proposed where invalidation reports are broadcasted only when data items are updated unlike periodic invalidation reports scheme where the server broadcasts invalidation reports periodically [4]. In the proposed caching scheme, an additional memory, called the Home Location Cache (HLC), is used for each mobile host in order to maintain each data item cached by the MH and the time-stamp of each data item last updated. The HLC of each mobile host is found at a Mobile Switching Station (MSS).

When a MH receives a query, it checks for the data in its cache; if the data is valid, then the MH answers the query immediately by using its local cache. Otherwise, the MH sends a request for the data to the MSS, which in its turn sends a request to the server for the data. When the MSS receives the data, it saves an entry of each data item in the HLC and forwards the data to the MH.

Each MH preserves a time-stamp for its cache, called the cache time-stamp, which is the time of the last invalidation report received by the MH. The cache time-stamp is used to decide which invalidation reports to discard or to resend to the MH just reconnected. The invalidation reports with a time-stamp less than the cache time-stamp of the MH are discarded and the ones with a time-stamp greater than the cache time-stamp in the HLC are resent to the MH. In fact, when a MH reconnects to the server after a disconnection for a period of time, it sends a probe message containing the cache time-stamp. The HLC responses to the MH message by sending an invalidation report. Thus, the MH can recognize which data was updated during its disconnection by using the cache time-stamp. This scheme handles frequent disconnections of the MHs by maintaining the consistency of their caches with the cost of extra memory used for HLCs.

3. THE SCALABLE ASYNCHROUNOUS CACHE CONSISTENCY SCHEME

For mobile computing environment, there are two types of cache consistency maintenance algorithms: stateless and stateful. In the stateless approach, the server is not informed about the client's cache content so it periodically broadcasts a data invalidation report (IR) to all mobile users (MUs). In such algorithms, mobile support station (MSS) does not maintain any state information about its mobile user caches (MUCs), thus allowing simple database management for the server cache (SC) but poor scalability and ability to support user disconnectedness and mobility. On the other hand, the stateful approach is used with large database systems at the cost of complex server database management. The communication in these approaches is reliable between MUs and the mobile support station (MSS) for IR broadcast, which means an MU has to send back an acknowledgement for each IR received from the server broadcast. Thus, if a mobile user is disconnected, it does not receive any IR broadcast and the server, which does not get any acknowledgement, has to resend the IR again.

SACCS combines the positive features of both stateless and stateful approaches [9]. Under unreliable communication environments, SACCS provides small stale cache hit probability; A stale cache hit for a data entry occurs at a connected MU when the MU misses the IR of the data entry and when the update time for the data entry is during its TTL (time-to-live period).

SACCS was proposed to maintain the mobile user cache (MUC) consistency for systems with read-only transactions. Maintaining minimum state information, SACCS has the positive features of both the stateless and stateful approaches. The advantage of SACCS over the asynchronous stateful algorithm, where the mobile support station (MSS) has to identify all data objects for every MUC, is that the MSS needs to recognize only the valid state of MUC data objects in SACCS. On the other hand, the advantage of SACCS over the stateless approach is that in SACCS, the server does not need to periodically broadcast IRs to all MUs, thus reducing the number of IR messages sent through the network. In addition, the broadcast channel efficiency progresses in SACCS, which has uncertain and ID-only states in MUC in order to handle sleep-wakeup patterns (or disconnection-reconnection patterns) and mobility of all MUs.

SACCS consists of four key features which make it a highly efficient, scalable and low complexity algorithm. Flag bits are used at SC and MUC, an ID is used for every data entry in MUC after its invalidation. All valid data entries in MUC become in an uncertain state when an MU reconnects (or wakes up), and each cached data entry has a TTL (time-to-live).

When an MU retrieves a data object from the MSS, the flag bit in SC is set so the MSS has to broadcast the IR to all MUs. On the other hand, if the flag bit is not set, the MSS does not have to send an IR to the MUs. In

this case, bandwidth utilization is reduced by avoiding unnecessary IRs. When a data entry in an MUC becomes invalid, it is removed and only its ID is kept (set to ID-only state), making the management of sleep-wakeup pattern simple. When an MU wakes up, all valid data items in its cache are set to uncertain state. TTL is estimated by an MSS for each data item depending on its update history. If the last update time of a data item added to its TLL equals to the current time, the data item is set to an uncertain state. This process helps in preventing a stale data object from being accessed in an MUC for a long time due to an incorrect IR arrival to the MU (IR loss).

The LRU (Least Recently Used) based replacement algorithm was used with the SACCS for the management of MUC. With the LRU algorithm used, a data object is moved to the head of the cache list when it is hit or newly cached.

4. EXPERIMENTAL RESULTS

The simulation of SACCS is running under fixed parameters and conditions such as fixed number of documents exchanged and mobile users, fixed mobile cache sizes, specified periods of time, etc for each of LRU, which is the based cache replacement algorithm for SACCS mobile user cache management, and the proposed cache replacement algorithms: LFU, MRU, MFU, and FIFO. Using these cache replacement algorithms with SACCS affects the system performance with different factors such as the simulation results of the cache miss ratio, the total miss, the total hit in MUCs and the delay of data during certain periods of time. A comparison between the cache replacement algorithms results on SACCS is done to show the advantages and disadvantages of every cache replacement algorithm used. The statistical results obtained are based on eight simulation time units with an interval of 50000 microsecond of simulation time.

4.1 Cache Miss Ratio

The cache miss ratio is the ratio of the number of unfound data items in the cache over the number of all requested data. The cache miss ratio simulation results are presented in figure 1, where the statistical results of each cache replacement algorithm miss ratio versus specific periods of simulation time are shown.

Among all the cache replacement algorithms, it is shown that the FIFO has the lowest miss ratio with the average of 0.69053, while the MFU has the highest miss ratio with average of 0.750494. With the SACCS, the first in first out replacement strategy ensures the highest cache hit ratio (i.e. the lowest cache miss ratio) among the other strategies so it handles the mobile users' queries for data faster than the other strategies. Thus, using the FIFO with SACCS improves the performance of the system because it has a lower miss ratio result than the based LRU with a miss ratio average of 0.696047.

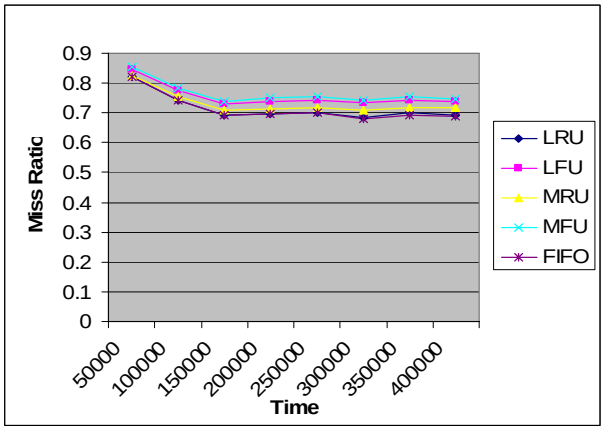

Figure 1 Miss ratio versus simulation time

4.2 Total Delay

The total delay is defined as the period of time between the time the request is issued and the time the result is received by the mobile user application. The total delay simulation results are presented in figure 2, where the statistical results of each cache replacement algorithm total delay time versus specific periods of simulation time are shown.

Among all the cache replacement algorithms, it is shown that the FIFO has the lowest total delay with the total average of 14893.7, while the MFU has the highest total delay with the total average of 21953. When used with the SACCS, the first in first out replacement strategy ensures the lowest period of delay time between a request and its answer among the other strategies because of different factors. The miss ratio is one of the factors that affects the delay time. When a requested data is invalid or not found in the mobile user cache, it is time-consuming for the MU to get the data from the server. And in this case, the FIFO is previously shown to have the lowest miss ratio among other replacement algorithms, which makes the total delay lower. Thus, using the FIFO with SACCS improves the performance of the system because it has a lower total delay result than the based LRU with a total delay average of 15590.5.

4.3 Total Hit

The total hit is defined as the number of data items hit or accessed in the mobile user cache. The total hit simulation results are presented in figure 3, where the statistical results of each cache replacement algorithm total hit versus specific periods of simulation time are shown.

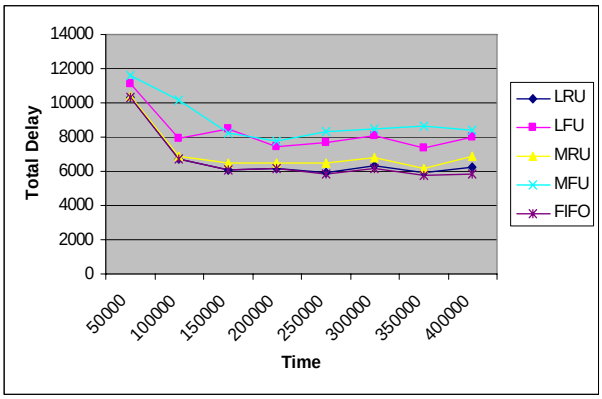

Figure 2 Total delay versus simulation time

Among all the cache replacement algorithms, it is shown that the FIFO has the highest total hit with the total average of 5184, while the MFU has the lowest total hit with the total average of 4166. The total hit affects the system performance because it affects the delay time of a user request for data and the network traffic. In fact, as the number of data hit gets higher, users' queries are answered more quickly because fetching the data from the user cache is much faster than getting it from the server database. And the network traffic is reduced because the number of users' requests for data from the server database is lower as the data hit in the users' caches is higher. Consequently, using the FIFO with SACCS improves the performance of the system because it has a higher total hit result than the based LRU with a total hit average of 5091.

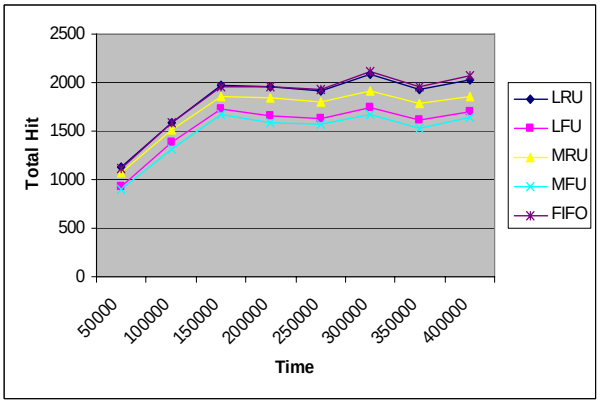

Figure 3 Total hit versus simulation time

4.4 Total Miss

The total miss is defined as the number of data items invalid or missed in the mobile user cache. The total miss simulation results are presented in figure 4, where the statistical results of each cache replacement algorithm total miss versus specific periods of simulation time are shown.

Among all the cache replacement algorithms, it is shown that the FIFO has the lowest total miss with the total average of 11565, while the MFU has the highest total miss with the total average of 12522. Because the total miss factor is inversely proportional to the total hit factor, the performance of the system is reduced as the total miss gets higher. In fact, when a requested data is missed in the user cache, a delay exists for the data should be brought from the server database through the network to the user. A data miss costs the system a delay time and additional network use. Consequently, using the FIFO with SACCS improves the performance of the system because it has a lower total miss result than the based LRU with a total miss average of 11656.

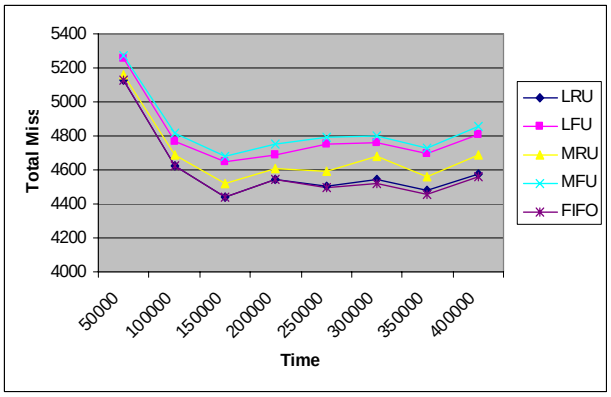

Figure 4 Total miss versus simulation time

4.5 Performance Evaluation

As already shown in the previous charts of cache miss ratio, total delay, total hit, and total miss, the FIFO has the best results among the cache replacement algorithms: LRU, LFU, MRU, and MFU. The FIFO has the lowest miss ratio, the lowest total delay, the highest total hit, and the lowest total miss among all proposed replacement algorithms. All these factors affect the system performance of SACCS by affecting the speed of handling the users' requests for data and the network traffic. Thus, the system performance of SACCS with the based cache replacement algorithm LRU can be improved by using the FIFO for MUC management instead of the LRU.

5. CONCLUSION

Mobile computing is one of the most important technologies in the computer field. Data caching is imposed when data is exchanged between the mobile users and the server in order to improve the performance of the system. Maintaining cache consistency is one of the major problems arising in mobile computing. A cache consistency maintenance algorithm called Scalable Asynchronous Cache Consistency Scheme (SACCS), gathering both stateless and stateful approaches, was proposed with the Least-Recently-Used (LRU) replacement algorithm to manage mobile user caches. A detailed description for the SACCS is done.

In this work, different cache replacement algorithms are proposed to be used with SACCS and compared with the LRU already used. The impact of the proposed cache replacement algorithms LFU, MRU, MFU, and

FIFO on the system performance is studied. A simulation of SACCS with each of these algorithms is done, producing results evaluating the system performance. The simulation results, such as the miss ratio, total delay, total hit, and total miss are compared showing the variation of the performance of the system. Statistical results point out the advantages and disadvantages of each proposed cache replacement algorithm compared to the based LRU.

As a result, the FIFO used with SACCS is shown to be the cache replacement algorithm with the best simulation results compared to all the cache replacement algorithms used. The FIFO has the lowest miss ratio, the lowest total delay time, the highest total hit, and the lowest total miss. In this case, the system performance of SACCS is improved when the FIFO is used instead of the based LRU for MUC management.

REFERENCES

[1] G. Cao. "Adaptive Power-Aware Cache Management for Mobile Computing Systems." IEEE Transactions on Computers, Vol. 51, No. 6, pp. 608 – 621, 2002.

[2] P. Chuang and C. Hsu. "An Efficient Cache Invalidation Strategy in Mobile Environments." Proceedings of the 18th International Conference on Advanced Information Networking and Applications, Vol.2, pp. 260- 263, 2004.

[3] W. Hou, C. Wang, and M. Su. "Composing Optimal Invalidation Reports for Mobile Databases." Journal of Digital Information Management, Vol. 3, No.2, pp. 126-132, 2005.

[4] A. Kahol, S. Khunara, S. Gupta, and P. Srimani. "A Strategy to Manage Cache Consistency in a Disconnected Distributed Environment." IEEE Transactions on Parallel and Distributed Systems, Vol. 12, No. 7, pp. 686 – 700, July 2001.

[5] K. Y. Lam, E. Chan, and M. Au. "Broadcast of Consistent Data to Read-Only Transactions from Mobile Clients." Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, Louisiana, USA, 1999.

[6] S. Mazumdar and P. Chrysanthis. "Achieving Consistency in Mobile Databases through Localization in PRO-MOTION**."** Proceedings of the 10th International Workshop on Database & Expert Systems Applications, Washington, DC, USA, 1999.

[7] E. Pitoura. "Supporting Read-Only Transactions in Wireless Broadcasting." Proceedings of the DEXA'98 Workshop on Mobility in Databases and Distributed Systems, 1998.

[8] A. Seetha and A. Kannan. "Maintaining Data Consistency in Mobile Database Broadcasts." Proceedings of MAP India, 2002.

[9] Z. Wang, S. Das, H. Che, M. and Kumar. "Scalable Asynchronous Cache Consistency Scheme (SACCS) for Mobile Environments." IEEE Transactions on Parallel and Distributed Systems, Vol. 15, No. 11, pp. 983 – 995, 2004.

[10] K. L. Wu, P. Yu, and M. Chen. "Energy-Efficient Caching for Wireless Mobile Computing." Proceedings of the Twelfth International Conference on Data Engineering, Washington, DC, USA, 1996.

[11] L. Yin, G. Cao, and Y. Cai. "A Generalized Target-Driven Cache Replacement Policy for Mobile Environments." Journal of Parallel and Distributed Computing, Vol. 65, No. 5, pp. 583 – 594, 2006.

Ramzi A. Haraty is an associate professor and the assistant dean of the School of Arts and Sciences at the Lebanese American University in Beirut, Lebanon. He is also the Chief Financial Officer of the Arab Computer Society. 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 Association of Computing Machinery, Arab Computer Society and International Society for Computers and Their Applications.

Lana Turk received her B.S. degree in Computer Science from Haigazian University – Beirut, Lebanon, and her M.S. degree in Computer Science from the Lebanese American University – Beirut, Lebanon. Her research interests include mobile networking and cache management.

Acknowledgement: This work was funded by the Lebanese American University.