A Client-Server Architecture for Context-Aware Search Application

Feng Gui, Magno Guillen, Naphtali Rishe, Armando Barreto, Jean Andrian, Malek Adjouadi
Center for Advanced Technology and Education
College of Engineering and Computing
Florida International University
gui_feng@yahoo.com, Magno.Guillen@fiu.edu, rishen@cs.fiu.edu,
Armando.Barreto@fiu.edu, Jean.Andrian@fiu.edu, adjouadi@fiu.edu

Abstract

This paper develops a client-side context-aware search application which is built on the context-aware infrastructure. A context-aware architecture is designed to collect the mobile user's context information, derive mobile user's current context, distribute user context among context-aware applications, and support the context-aware applications. The context acquisition is centralized at the context server to ensure the reusability of context information among mobile devices, while context reasoning remains at the application level. Algorithms are proposed to consider the user context profiles. By promoting feedback on the dynamics of the system, prior user selection is now saved for further analysis expediting a subsequent search. A software-based proxy is set up at the client side which includes the context reasoning component. Implementation of such a proxy supports that the context applications are able to derive the user context profiles. To meet the practical demands required of a testing environment, a software simulation using Yahoo search API is provided as a means to evaluate the effectiveness of the design approach in a realistic way. The integration of user context into Yahoo search engines proves how context-aware searches can meet user demands for tailored services and products in and around the user's environment.

Keywords
Context Awareness, Context Server, Mobile Search, Personalized Search, User Profile.

I. INTRODUCTION

A decade ago, pioneer Mark Weiser envisioned that a human would live in an environment surrounded by hundreds of invisible computers connected with wireless networks [1]. Chen and Kots stated "Context is the set of environmental states and settings that either determines an applications' behavior or in which an application event occurs which is interesting to the user" [2]. In ubiquitous computing, "context" should reflect the mobile user's current state including physical and psychological behavior, mobile applications based on the explicit and implicit input, artifacts, social interaction, surround events, and environment.

To support such "context aware utility" anytime and everywhere, a context-aware architecture is required to actively acquire, analyze, and adapt to mobile user's given contexts, such as physical environment, social activities, and other dynamic characteristics at different levels without consuming much of the user's attention**.** Harry Chen proposed a context-aware architecture that separated the context acquisition completely from the resource-constrained mobile devices [3]**.**

Unlike most existing context architectures, this paper tries to strike a balance of responsibilities between the context server and mobile devices. The context server undertakes the context data collection from various sources including wireless sensor network and the mobile devices. In addition, the context server allows third-party vendors or service providers to register their services and products. Context server distributes the user context among networks, user devices, and applications.

The mobile applications become increasingly important source for user data. The ability of acquiring user data at client side should not be taken lightly. The mobile devices should play a role in collecting user context data given the increasingly capable applications.

Because of the variable and unstable nature of user context, the difficulty in context awareness at client side lies in the extraction of useful feature/context from changeable user situations. It is therefore important to design new algorithms at the client side to undertake the preliminary context analysis of user situation. Algorithms and a proxy at client side are proposed to relieve the networks (i.e., servers) of the computing burden and reduce the need for uplink bandwidth [4]. Such proxy manages the user context profiles to 1) filter out irrelevant user information and describe the user's current context; 2) update server with current user context; 3) assist in predicting user intention at both server and client sides.

For mobile users, one of the challenges in the information retrieval is the establishment of a context sensitive retrieval process. Furthermore, the mobile search for information on portable devices should match the web searches. Currently, most web search algorithms are limited solutions for context sensitive retrieval and mobile search. This is so because most existing algorithms do not take into account mobile user context inputs, such as surrounding environment. For this reason, a context-aware search scheme at server side (i.e.,

978-0-7695-3767-2/09 $25.00 © 2009 IEEE
DOI 10.1109/NBiS.2009.75

carrier's network) is devised to provide content based on the compiled context profiles.

In PC-based web search, researchers have explored personalized search to improve topical relevance of the result documents. Shen et al. [5] studied user's immediate and short-term search context to expand the current query. Qiu and Cho [6] learned user interest from the click history and developed a ranking mechanism based on the user interest. Chirita et al. [7] proposed personalized search and summarization algorithms which assist search keywords expansion based on extracted information from local desktop. Duo et al. [8] and Teevan et al. [9] investigated the personalized search strategies and stated that personalization improves the search accuracy on ambiguous queries.

So far, the personalization is studied only for PC-based web search. Most of such personalization strategies are limited to the user search history, returned search results, and documents stored in PC. We extend this line of work into mobile search and derive user context based on profiles which adapt to user location, activities, interaction history, and preferences. Unlike stationary PC, user can access the mobile web anytime and anywhere. The user context varies, i.e., people, activities, and settings. We further predict that the increased usage of location-based applications and services will lead mobile users to search for local services and information.

Mobile search has come to researchers' attention. Liu and Birnbaum [10] developed "LoalSavvy" which is a system that aggregates local views associated with news events or topics. Vadrevu et al. [11] pointed out the importance of identifying the regional sensitive queries. Hence, the mobile search must cope with local search query.

The aim of this paper is to introduce a client-server architecture for context aware search, assuming that the data collection hardware infrastructure required is available and properly deployed in site. The software components of the architecture are described and implemented. To validate the performance of the proposed architecture a simulation environment was utilized integrating a Yahoo search API as a means to evaluate the effectiveness of the design approach in a most realistic way.

II. Context-Aware Search Architecture

The proposed architecture consists of the modules depicted in Fig, 1. These modules are: the context interpreter, service registry, context manager, request interpreter, policy registry, and context-aware search manager.

These relevant components are now described; emphasis is placed on elucidating the different functions they serve in this study.

A. Context Interpreter

The context interpreter is responsible for interacting with low-level sensor networks, collecting context information, modeling user context, resolving context inconsistency, and updating context entities to help greatly to solve possible ambiguous query terms. This component includes code libraries of procedures for acquiring contextual information from heterogeneous sensor networks. In addition, context interpreter interacts with mobile devices to receive the user context profiles compiled from the client side. The purpose of the context server is to distribute user context to context-aware applications.

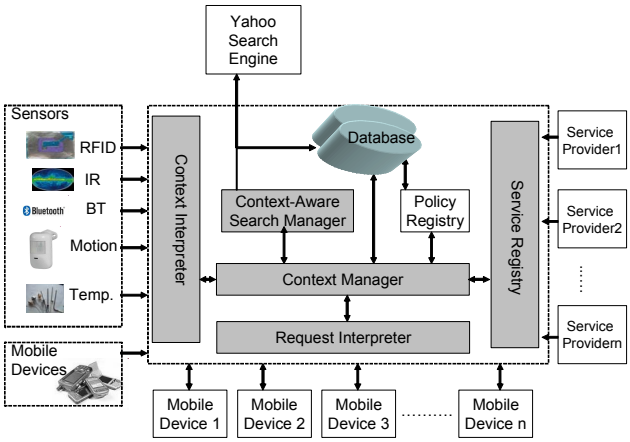

Fig. 1 Context-Aware Server Architecture

B. Service Registry

The service registry interacts with third-party service providers to register services which will be available to mobile users. Service providers usually provide services and products that might serve mobile user's interests given the current situation. For example, if mobile users are shopping in the local grocery store, then vendors and companies in food industries would like to promote similar products to users. Another example would be business in the local neighborhood try to get visitors' attention when they go by their areas. Context server distributes service providers' information to mobile user according to the user's current context. In a sense, the server acts like a broker who matches services to user's need which is derived from the user context. In this perspective, figure 2 illustrates the service mechanism among server, service providers, and mobile users. Service providers describe services to the service registry with the following format:

- Service type
- Service scope
- Network service and security policy
- Context inputs
- Context change reminder type
- Service description

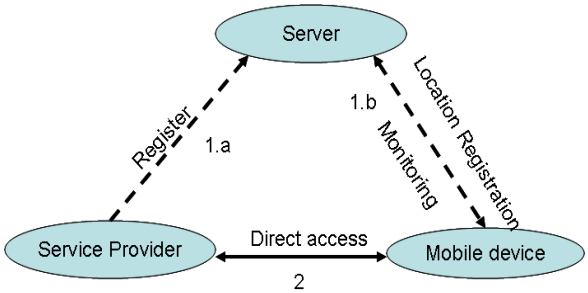

Fig. 2 Context Service Model

C. Service Interpreter

Service interpreter interacts with mobile devices to receive and interpret application requests. It formats the query or request. The formatted requests are passed to the service manager for service matching.

D. Context Manager

The context manager processes the formatted application requests. If the application requests can be solved with the registered service providers, then the context manager fetches the related information from the database and sends service provider's information to the mobile devices. However, if a user sends search requests that cannot be solved by the context manager, then the user's requests are forwarded to the context-aware search manager.

E. Policy Registry

The emerging trends of IP convergence and the convergence of heterogeneous networks make the management of Next Generation Networks (NGN) a priority. The context aware architecture must cope with the heterogeneous networks which might be WLAN, GSM, CDMA 2000, WCDMA, or TD-SCDMA. Policy registry stores the management policies the architecture enforces. These policies are important to make services adapt to the underlying network characteristics. In addition, the service provider has policies such as terms, conditions, fee schedules, and user management applied services. Moreover, mobile users have their preferences regarding the personal information sharing among the service providers. Therefore, the policy registry interacts with network management applications, service providers, and user privacy profiles.

F. Context-Aware Search Manager

Mobile search means user submits query to search engine on mobile devices. Mobile search stems from PC-based web search, but differs from PC-based web search due to constraint factors such as small LCD display, tiny keyboard, and network limitation. Mobile search is the second most used application only after social networking in wireless internet [12]. Search engines, such as Google or Yahoo, appear in top three of the most visited web sites in terms of wireless internet usage.

Most mobile search queries are kept short due to the hardware limitations afore mentioned. Early studies [13] attempted to provide solutions to mitigate the hardware limitations of the wireless devices. The top 100 mobile queries at AT&T [14] reveal that a great number of search queries are navigational in nature [15]. The navigational searches, for example "Google", usually steer mobile users to specific web sites conveniently. Unlike navigational queries, words like "images" and "free" which are informational and transactional are ambiguous to search engine. A housewife and an iPhone user interpret "apple" differently in search context. A housewife is likely to know the apple variety and prices at the local grocery stores, while an iPhone user is interested in service or products related to iPhone. Researchers studied methods and models to determine the query ambiguity. Clarity score [16] was proposed to evaluate the relative entropy between the query language model and the collection language model. A large click entropy indicates that user clicks more web pages to solve the query, thus the query is ambiguous. A small click entropy means mobile users have common understanding for a search query. Song [17] developed classifier to automatically identify three types of queries, ambiguous, broad, or clear query. We believe these methods and algorithms work equally well to identify the query ambiguity in the mobile search.

G. Client-Server Solution for Context-Aware Search

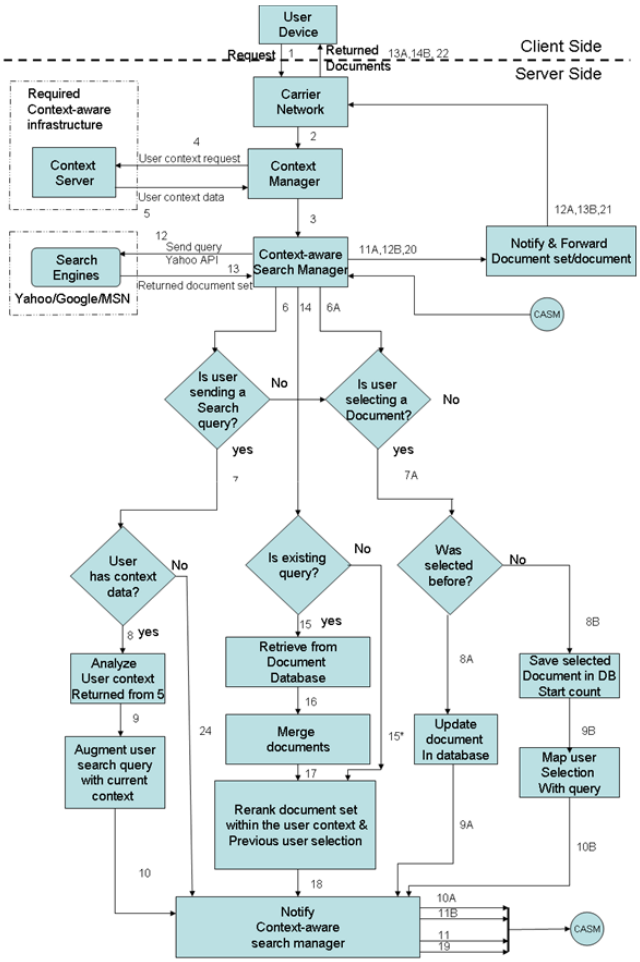

Fig. 3 Context-Aware Search Design

The flow chart provided in Fig. 3 describes logical steps of how the context-aware search application solves the mobile user's query.

III. REQUIRED CLIENT SIDE ARCHITECTURE

The traditional client/server paradigm fits well into the context-awareness applications. The carrier's network functions as the server providing data and voice services to subscribers. Mobile devices request services from the network. What makes this approach different to other models is that the mobile devices play an extensive role in collecting, analyzing, and extracting context entities. Context profiles compiled at client side greatly reduce computing burden at network/server side.

Fig. 4 demonstrates the structure of the paradigm for the client-side solution. The user inputs (voice or digital data) and surrounding environment inputs (i.e. temperature, position, altitude, etc.,) are collected by the hardware logic or the applications such as the operating system. The context-aware proxy further inspects inputs and extracts context entities from the inputs. Finally, the proxy compiles the context profile and sends it to the network/server. The network/server learns the user situation using the context profile. Server application provides services and data to client based on the learning of the user situation.

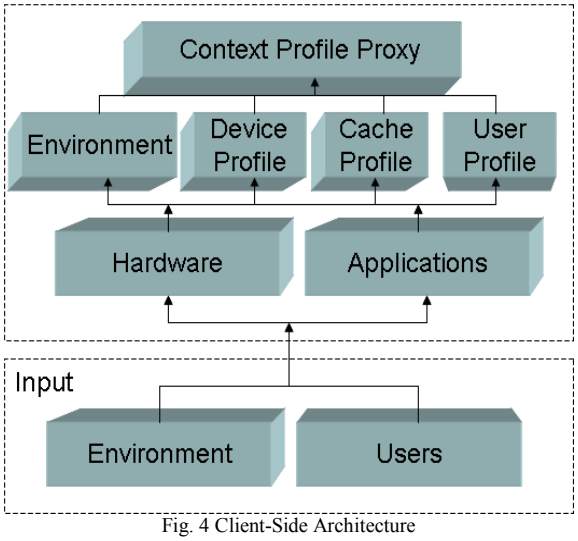

Fig. 4 Client-Side Architecture

A. *Client Side Requirements for Context-Aware Solutions*

The client proxy, running in the user devices, monitors and collects mobile user information through sensors and applications. In addition, the proxy further compiles the context profiles which adapt to and reflect mobile user's changing situation. Virtual frame and reference frame are thus proposed to improve the efficiency of the proxy's operation at the client side.

The context profile is a collection of context entities extracted from the on-board sensors, client applications, user activities, and so on. Proxy on mobile devices frequently updates context profiles and uploads them to the network for reference if necessary. Due to the limited uplink bandwidth, the context profiles should be concise. As user changes activities/context, context entities are added to or dropped from the profiles. Whenever an addition or deletion of context entities occurs, the proxy notifies the network of the changes. There are four context profiles managed by the client proxy: 1) user profile; 2) device profile; 3) environment profile; and 4) data profile.

In this paper the concept of virtual frame is introduced. The idea of virtual frame comes from the movie industry. The virtual frame expresses the meanings of the mobile user's context by featuring weights that are most significant at a particular moment. In other words, the virtual frame does not store any graphical components but weights of the context entities that capture the context of the mobile user at specific time frames. Fig. 5 illustrates this concept. For example, virtual frame f_1 consists of weights that are recorded to capture the context of the mobile user at time t(1). Likewise, virtual frame f_n contains all meaningful weights that describe the mobile user characteristics and environment at time t_n. Time plays a great role in predicting the mobile user's context or intention. Intuitively, the user's context is likely to remain unchanged or consistent over a short time interval in most scenarios.

With the virtual frame defined, the notion of reference frame can be introduced. The reference frame includes data weights as virtual frames. However, a reference frame is not like a virtual frame where the reference frame is a moving average of the virtual frames. The reference frame averages the most-recent N virtual frames, as shown in Fig. 6, depending on the configuration of individual mobile devices.

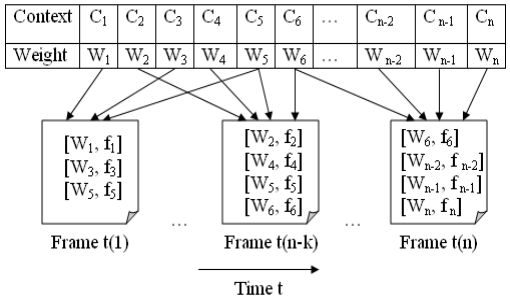

Fig. 5 Structure of the Virtual frame

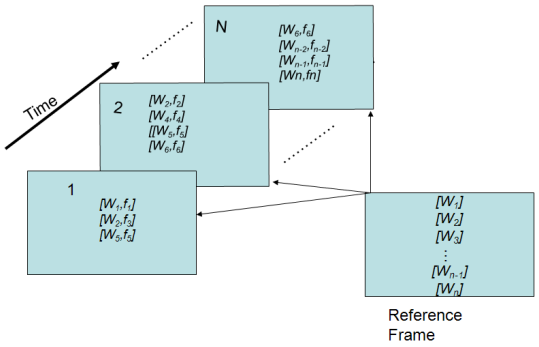

Fig. 6 Structure of the reference frame

Weights included in the reference frame are calculated as shown in Eq. (1):

$$W_k = \frac{1}{N}\sum_{f=1}^{N} W_{fk} \tag{1}$$

Where f indicates the virtual frame number as $f = 1, 2, 3, 4, \ldots, N$; while k refers to all individual weights contained in N virtual frames as $k = 1, 2, 3, 4, \ldots$

It is likely that a specific weight, W_k, might not be captured in every virtual frame, f_i, as mobile user's context changes over the time. Clearly, weights that appear most frequently in virtual frames would be the dominating weights, so they appear in the reference frame as well. Thus, the reference frame is a much more balanced frame that describes the current mobile user's context with respect to both present and past. Definitely, there would be a tradeoff between the history context and the current context. As N increases, more virtual frames in the past would be included for consideration. So the latest virtual frame has less bearing on the reference frame. Therefore, the reference

frame derived from a large set of virtual frames would account for more consistent context for mobile users. On the other hand, if N decreases, then the most recent virtual frame weighs more on the reference frame, which better fits mobile users changing their context dramatically in a short period of time.

Obviously, it is desirable to make N adaptive to the context change. If the client proxy figures out that the mobile user's context remains relatively consistent meaning no new context entities are brought into the most recent virtual frame, then N is incremented up to the total virtual frames in memory as in Eq. (2):

$$N = N_{previous} + 1 \tag{2}$$

where $N_{previous}$ refers to the total number of virtual frames used to derive the reference frame in the last iteration. By increasing N, the reference frame would take more historical virtual frames into consideration than those considered if the context changes were less.

If the client proxy detects that the most recent virtual frame will bring in new weights, considering they do not exist in the current reference frame, then N changes according to Eq. (3):

$$N = \left\lceil N_{previous}\left[1 - \frac{k(w_{new})}{k(w_{reference}) + k(w_{new})}\right] \right\rceil_{ceiling} \tag{3}$$

where $k(w_{new})$ counts new weights that do not exist in the last reference frame; $k(w_{reference})$ sums the number of total weights recorded in the last reference frame. Next, the proxy rounds N by taking the ceiling value. As N decreases, the last virtual frame could weigh more in the new reference frame. Thus, the cache system adapts much faster to the mobile user's context which changes dramatically over a short time.

B. Implementation of an Artificial Neural Network (ANN) for Deriving the Weights of the Context Entities

An ANN was implemented to derive the weights for the context entities. For the implemented ANN, the learning rule is based on the following general equation:

$$\Delta W_{ij}(t) = x_i(t).y_i(t) \tag{4}$$

In this case, $x_i(t)$ is defined as a function of the context changes (ΔC) and frequency count (ΔF) of the context entities in the context cache, which is managed by the client proxy between the current epoch and the previous epoch. In addition, $y_i(t)$ is defined as the weight values, $W_{reference}$, in the current reference frame.

Thus, the derived equation of weight change for our approach at epoch t is given by Eq. (5):

$$\Delta W_k(t) = \left|\Delta C(t) + \mu \Delta F(t)\right|_k W_{k,reference}(t) \tag{5}$$

Where $\Delta C(t)$ is calculated as the ratio given by:

$$\Delta C(t) = \frac{|C(t) - C(t-1)|}{C(t-1)} \tag{6}$$

$C(t)$ in this case is the current context value, and $C(t\text{-}1)$ is the previous context value. If $\Delta C(t)$ is zero, then this same entity is recomputed as the following absolute difference:

$$\Delta C(t) = -\left|C(t) - C_{average}\right| \tag{7}$$

The $C_{average}$ value in this case represents the mean of the elements C(1) through C(t-1). The term $\Delta F(t)$ is the difference between the virtual frame at current epoch and the virtual frame from previous epoch based on frequency counter for context entity, $\Delta F(t) = F(t) - F(t\text{-}1)$. μ is a coefficient that is machine specific.

Finally, the weights for context entity k at epoch t are updated as

$$Wk(t) = W_k(t-1) + \Delta W_k(t) \tag{8}$$

If the virtual frame brings in new context weights that do not exist in the current reference frame, then the initial value of Wk(t) is set to 1.

IV. EVALUATION OF CONTEXT AWARE DESIGN

A. Software Platform

Java platform is selected to implement components on both client and server sides. Java 2 Micro Edition (J2ME) is a programming language for writing the applications on the mobile devices. Sun Java Wireless Toolkit 2.5.2 is used to run the J2ME applications.

The database management software deployed is MySQL 5.1.30 which includes features like stored procedure, trigger, and query caching. Like Java, MySQL can run on multiple operating systems such as Windows and Linux. MySQL server is available to individual developers as free software under GNU General Public Licenses.

Yahoo Search API is utilized to integrate the search function into the simulation. Yahoo search site takes the query as input and returns the documents in XML. An XML parser is required to interpret the search results. The search manager ranks these documents and reorders them based on how relevant they are within the current user context.

B. Evaluation Strategy

The ideal prototype requires significant resources in order to build the wireless sensor networks. It is also needed the infrastructure for Piconet, WLAN to provide services, and carrier's network in order to independently evaluate the merits of the proposed design strategy. Obviously, the cost to set up all the required testing environment is not permissible. Therefore, the software simulation using a Yahoo API service is provided as a means to evaluate the effectiveness of the design approach in a most realistic way. At the same time, this simulation alleviates the prohibitive cost and time demands of the eventual deployment of the hardware requirements.

Client proxy is programmed to analyze the user context data collected on the mobile devices. An ANN-based algorithm is implemented to calculate the context weights. Context profiles are compiled to reflect user's current situation. Context search engine is simulated to take advantage of user's context

profiles. The context search engine combines searching algorithms with context profiles.

In general, a mobile user who is roaming sends a query from a mobile device to the server. Server works on the user query and forwards it to the Yahoo search engine. Yahoo search engine returns the document sets back to the server. The server processes the document sets and sends them back to the mobile device.

C. Evaluation Methods

One user scenario is provided to evaluate the context-aware designs at both server and client sides. This scenario is close to the daily activities of mobile users. The value of μ, the machine specific coefficient in Eq. (5), is set between 0.0008 and 0.005 in the simulation process. To simplify the context computing, it is necessary to quantify some context entities into digital format. For example, weather: sunny = 1, cloudy = 2, raining = 3; Location is expressed by (x, y) such as (1, 1), (2, 2), …, (m, n); numbers are assigned arbitrarily to store type: department store 10, Woman Cloth = 20, foot wear = 30; Wendy's has a type 61, Subway is set to 63, Gold Gym = 42, Bank = 75, …

Scenario: Roaming off the main campus of Florida International University

In this scenario, a mobile user goes off the main campus of Florida International University. The Google Earth screen shot of this travel route is as shown in Fig. 7.

Fig. 7 Google Earth Screen Shot of Travel Route for the Scenario

The user in this scenario walks east bound. We describe user activities through the following steps.

- User is walking off the main campus and crossing the 107th Avenue. The location coordinate (6, 6) at the speed of 2 mile/hr while the noise level is at 60 db.
- User is still walking at the same speed passing Wendy's restaurant and location coordinate (4, 7). The noise level rises to 63 db.
- The noise level remains the same. User walks by Subway, location (4, 8), and at the speed of 1 mile/hr.
- User walks at 0.5 mile/hr and passes Gold's Gym. The noise level continues to rise to 77 db. Gold's Gym has location coordinate (2, 6).
- User walks by Terry Bank at 0.2 mile/hr. The noise level is at 67 db at this regional bank with coordinate (2, 5).
- User arrives at Little Caesars. User walks by at this restaurant whose location coordinator is (2, 4) with a noise level of 67db.
- User reaches Publix supermarket (2, 3) and starts using his mobile device to search information. Noise level rises to 67 db.

The above 7 steps are summarized in Table 1.

Table 1 Context entities of the scenario

Step	Speed	Place	Noise	x	y
S(1)	2mile/h	59	60	6	6
S(2)	2mile/h	61	63	4	7
S(3)	1mile/h	62	63	4	8
S(4)	0.5mile/h	82	77	2	6
S(5)	0.2mile/h	75	67	2	5
S(6)	0.2mile/h	67	67	2	4
S(7)	0mile/h	68	67	2	3

The simulation results are listed in Tables 2 and 3 for the virtual and reference frames.

Table 2 Virtual frames of the scenario

Frame	Speed	Place	Noise	X	y
f(1)	1.000	1.000	1.000	1.000	1.000
f(2)	1.005	1.038	1.055	1.338	1.172
f(3)	1.512	1.078	0.005	0.452	1.345
f(4)	2.276	1.408	0.006	0.681	1.687
f(5)	3.653	1.536	0.007	-0.405	1.977
f(6)	0.809	1.707	0.001	0.133	2.383
f(7)	1.624	1.742	0.000	-0.018	2.991

Table 3 Reference frames of the scenario

Frame	Speed	Place	Noise	X	y
f(1)	1.003	1.019	1.027	1.169	1.086
f(2)	1.172	1.039	0.686	0.930	1.172
f(3)	1.448	1.131	0.516	0.868	1.301
f(4)	1.889	1.212	0.414	0.613	1.436
f(5)	1.710	1.295	0.346	0.533	1.594
f(6)	1.697	1.359	0.297	0.454	1.794

In this scenario, the context weights in the virtual frames capture the context changes detected. As the context entities change is less obvious, the context weights decrease. The weight of location changes significantly as user goes by different places. The location is an important context entity for this user. When it comes down to the ambiguous query, the context entities help the search engine to identify the user intention.

The user reference frame, generated at step 7, is listed as follows:

Frame	Speed	Place	Noise	X	y
f(6)	1.697	1.359	0.297	0.454	1.794

The user reference frame contains location information and store information that will help the search engine to optimize the search results. Suppose that the user enters the key word "apple". The interesting fact about "apple" is that this word is frequently associated with the high-tech company Apple Inc. which sells hot gadgets such as iPods and iPhones. In general, most search engines including Google will consider that the user was searching for products from Apple Inc. On the other hand, apple is a fruit which is a type of food. Fig. 8 shows the returned documents based on the user profile only.

Because the context file records that the mobile user is near the restaurant and grocery store, so the search word "apple" is more likely related to food. The search engine returns information about grocery stores and restaurants. On the other hand, the context search manager also considers the possibility that the search key word is related to products from Apple Inc. However, user needs to scroll down to see documents related to apple products.

In comparison, the Yahoo search engine, in general, is not aware of the user context. If the user submits the query "apple" to the search engine, the search engine returns the result documents that are mostly related to the Apple INC. This is due to the popularity of the Apple products such as IPhone and IPod. Clearly the mobile user benefits from the context-aware search approach.

Fig. 8 Returned Search Result for the Scenario

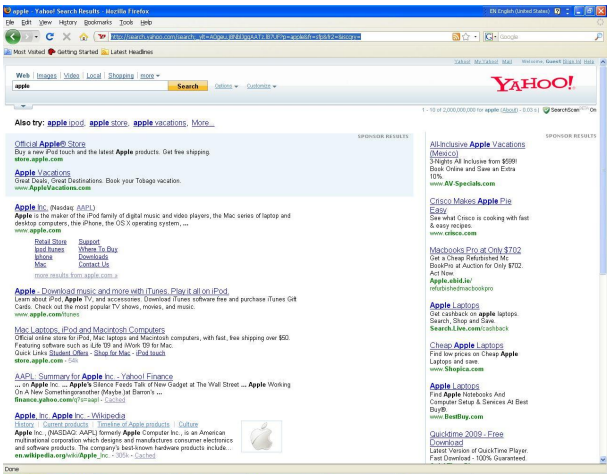

Fig. 9 Yahoo Search Results without User Context

Context search manager implicitly processes the user selection as the feedback. The essence of the user's feedback is that it beats any search algorithm so far. The feedback mechanism provides the semantic meaning of the searched keyword. Should the user decide to review with more detail the listed documents, the user selection contributes more votes for the selected documents. In turn, more votes improve the chance to move up in the listing order for future searches. Fig. 10 shows the user selection of one document.

The left screen shot in Fig. 10 shows that mobile user selects the document titled "Ipod near Miami, FL at TheFindLocal.com – Find products and stores nearby" from the list of documents. The right screen shot in Fig. 10 shows the description of the document. The feed back mechanism records the user selection and relates the selected document as the high ranking document to solve the future user query. Fig. 11 shows the new order of the search documents. The display order of the selected document is precedent.

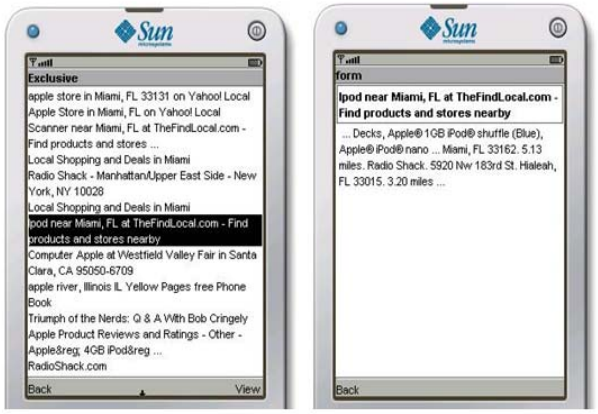

Fig. 10 Returned Search Result Augmented with User Feed Back

Fig. 11 Effect of a User Vote on the Listing Order

V. CONCLUSIONS

The main motivation on this paper is to propose algorithms and methods which enable mobile applications to learn user's current situation, considering that context reusability will result in making low level context acquisition transparent to application developers.

The major contribution of this paper is proposing a context aware design methodology that allows a context aware application developer to focus more on the context-aware application itself rather than dealing with the low-level hardware used to acquire the context data. Context awareness within this design concept defines the user context in a changing environment under different situations, and provides solutions for useful applications.

Context data acquisition using current technology is highly feasible if performed in a cost-effective way. The focus in this paper is placed on the search application with all its support algorithms. The context aware architecture at the server side provides a platform to share the user context among context-aware applications. Since context reusability release application developers from details of context acquisition, the application design is simplified. The required architecture is envisioned to accommodate third-party service providers. Service provider could register their services and products with the designed architecture. The architecture provides user with services and products based on the user's current situation. A context search manager is developed to provide search function based on the current user situation. A user feedback mechanism augments this new search engine.

Virtual frame and reference frame are introduced to reflect the mobile user's activities and surrounding environment. A proxy is added as a consequence to process user profiles and context frames.

Furthermore, this paper points out the importance of designing new algorithms at the client side to undertake the preliminary context analysis of the user situation. The client proxy should then relieve the networks (i.e., servers) of the computing burden.

Since Client proxy interacts with context server via API, Yahoo search API is integrated to evaluate the effectiveness of the proposed design structure in a most realistic way. The simulation results show that the overall design is highly effective, providing new features and enriching the mobile user's experience through a broad scope of potential applications.

REFERENCES

[1] M. Weiser, "The Computer for the Twenty-First Century," *Scientific American,* vol. 265, no. 3, pp. 94-104, 1991.

[2] G. Chen, and D. Kotz, "A Survey of Context-Aware Mobile Computing Research." *Technical Report TR2000-381*, Dartmouth College, 2000.

[3] H. Chen, "An Intelligent Broker Architecture for Pervasive Context-Aware Systems." vol. PhD thesis Baltimore County: University of Maryland, 2004.

[4] I. Chlamtaca and J. Redi, "Mobile Computing: Challenges and Potential " in *Encyclopedia of Computer Science*, 4th ed: International Thomson Publishing, 1998.

[5] X. H. Shen, B. Tan, and C. X. Zhai, "Implicit User Modeling for Personalized Search", Conference on Information and Knowledge Management, Bremen, Germany, 2005.

[6] F. Qiu and J. H. Cho, "Automatic Identification of User Interest for Personalized Search", The International World Wide Web Conference, Edinburgh, UK, May 22-26, 2006.

[7] P. A. Chirita, C. S. Firan, and W. Nejdl, "Summarizing Local Context to Personalize Global Web Serach", Information and Knowledge Management, Arlington, Virginia, November 5-11, 2006.

[8] Z. C. Dou, R. H. Song, and J. R. Wen, "A Large-scale Evaluation and Analysis of Personalized Search Strategies", International World Wide Web Conference, Banff, Canada, May 8-12, 2007.

[9] J. Teevan, S. T. Dumais, and D. J. Liebling, "To Personalize or Not to Personalize: Modeling Queries with Variation in User Intent", ACM SIGIR Conference, Singapore, July 20-24, 2008.

[10] J. Liu, and L. Birnbaum, "What do they think? Aggregating Local Views about News Events and Topics", International World Wide Web Conference, Beijing, China, April 21-25, 2008.

[11] S. Vadrevu, Y. Zhang, B. Tseng, G. Sun, and X. Li, "Identifying Regional Sensitive Queries in Web Search", The International World Wide Web Conference, Beijing, China, April 21-25, 2008.

[12] Opera Software, "State of the Mobile Web Report: First Quarter", May 20, 2008. http://www.opera.com/smw/

[13] A. Soffer, Y. Maarek, and B.W. Chang, "WWW2002 Workshop on Mobile Search", Mobile Search, Honolulu, Hawaii, May, 2002.

[14] B Meunier, "Characteristics of the Top 100 Mobile Search Queries at AT&T", Jan, 2008.

[15] B. J. Jansen, D. L. Booth, and A. Spink, "Determining the User Intent of Web Search Engine Queries", The International World Wide Web Conference, Banff, Canada, May 8-12, 2007.

[16] S. C. Townsend and W. B. Croft, "Quantifying Query Ambiguity", Proceedings of the Conference on Human Language Technology, San Diego , pp. 94–98,2002.

[17] R Song, Z. X. Luo, J. R. Wen, and H. W. Hon, "Identifying Ambiguous Queries in Web Search", The International World World Wide Web Conference, Banff, Canada, May 8-12, 2007.