Context Summarization & Garbage Collecting Context[1]

Faraz Rasheed, Yong-Koo Lee, Sungyoung Lee

Computer Engineering Dept. Kyung Hee University
449-701 Suwon, Republic of Korea
faraz@oslab.khu.ac.kr, {yklee, sylee}@khu.ac.kr

Abstract. Typical ubiquitous computing environments contain a large number of data sources, in the form of sensors and infrastructure elements, emitting a huge amount of contextual data (called context) continuously that need to be processed and stored in some context repository. Usually, this data is for software system's internal use to provide proactive services. Hence, it makes sense not to store this entire huge amount of data but to identify and remove some irrelevant data (garbage collecting context), summarize the left over and only store this summarized and more meaningful data. We believe that such a summarization will result in improved performance in query processing, data retrieval, knowledge reasoning and machine learning. Besides, it will also save the storage space required to store context repository. In this paper, we will present the idea and motivation behind context summarization and garbage collecting context and some possible techniques to achieve this.

1 Introduction

The idea of Ubiquitous Computing [1] is gaining popularity with every passing day. Several research groups are developing their own ubiquitous computing projects [2], [3]. Ubiquitous (or pervasive) computing provides computing environment where computing resources are spread through out, present everywhere in the environment and providing services to users seamlessly & invisibly without any explicit user intervention. A ubiquitous computing environment, thus, contains a number of devices, sensors, and software systems.

Context awareness is among the foremost features of any ubiquitous computing environment. In order to provide appropriate services, an application needs to be aware of the user and environmental context. Similarly at lower levels of abstraction, an application (or middleware) is required to be aware of the computational context including device and network state. So, what is 'context' itself? We take context as the 'implicit situational understanding' and consider all the information that defines a situation as context. So, location, temperature, network bandwidth, device profile, user identity can all be taken as the context information or simply context.

Since a Ubiquitous Computing system needs to deal with such huge and diversified information (context), there should be an appropriate context model to define,

[1] This work is supported by grant No. R01-000-00357-0 from Korea Science and Engineering Foundation (KOSEF)

represent, and store context efficiently in some context repository. The management of context information in ubiquitous computing imposes lots of issues and challenges. M. J. Franklin [4] has identified a number of issues in ubiquitous data management such as those posed by adaptivity, ubiquity, mobility and context awareness

We approach the context (or data) management in ubiquitous computing from a different perspective. We are working on to identify the relevance and significance of information that a Ubicomp system receives from sensors and its surrounding. We believe that identifying and removing the irrelevant context (we call it 'garbage collecting context') and summarizing the available or incoming context (which we call the 'context summarization') will result in the improved performance of knowledge reasoning, inference making, machine learning and efficient use of computing resource including the storage space required by the Context Repository.

The rest of the paper is organized as follows. Section 2 contains related work. Shortcomings of existing systems are in Section 3. Our proposed solution is in Section 4. Four techniques of Context Summarization are in Section 5 and our proposed model is in Section 6. Section 7 contains issues and challenges. Finally we mention about risk factors involved in Section 8 and conclude the paper in Section 9.

2 Related Work

Unfortunately, Garbage Collecting Context (GCC) and Context Summarization (CS) have not yet got the attention of researchers. One primary reason is majority of Ubicomp systems are academic projects and have not been deployed in real environment and used for elongated periods. The issues identified in this work come in one's attention when actual system is deployed and run for considerable time. The general focus of research community in ubiquitous computing is not towards context data management; but how to make ubiquitous computing operational in first place?

Several existing ubiquitous computing systems support features like noise filtering, privacy control, feature extraction [6], [3], [10] but we believe that using separate components for GCC and CS with clearly defining the responsibility of each component will produce better results; mainly because of the separation of concerns.

In Database Systems, data mining and data ware housing [7] use the concept of histogram [8] and multidimensional views of database and work on the aggregate, consolidated data instead of raw data to support higher level decision making and to identify the hidden patterns in data. Hence, when we extract underlying meaning from context data, it can be considered as something like 'Context Mining' where we extract higher level context from the lower level context. Online Analytical Processing (OLAP) and data mining are not done on actual data but on the historical, consolidated and aggregate data while we are performing the context summarization on actual context. Contrary to data mining and OLAP, we want to transform the raw context to summarized form taking less storage space and provide improved and efficient reasoning and machine learning. Anyhow, the concepts explored in the field of data mining and OLAP are highly useful for the Context Summarization.

Researchers in DBMS have also analyzed time series data streams for very large databases [9]. Here, they analyze the data coming in continuous streams with time.

They have proposed solutions on how to manage, represent and store the time series data streams. This is also highly related to the context summarization.

In traditional DBMS, data is seldom removed. But in our context summarizer, we do replace raw context with summarized information. Why? The answer lies in why, in first place, we are storing the context? We are storing the context and maintaining context history so as to reason on context, draw inferences from the context and make the machine learn. If the context is summarized properly, the application can reason, infer and learn about activities more efficiently; because, they need the history and consolidated data which we are providing as a result of context summarization.

3 Problem Definition

A ubiquitous computing environment comprises of a number of different sensors providing context information like Environmental context (temperature, pressure, light), Audio / Video, Location context, Computational context (network bandwidth, underlying operating system, hardware specification), the list goes on and on…

Context information comes in a continuous stream with each sensor emitting data regularly (at least during some interesting activity). We are heading towards flood of context data. Such a huge amount of data requires proper management. At this point, we need to answer what to do with such a huge amount of data? Do we need to store all of this? More importantly do we really need such a large amount of data?

Several data items sensed from the environment are required for some instant processing and reasoning, e.g., presence of a person can be used to trigger the activity of turning lights on or caching data related to a particular user. But, we also need to store context for later use; knowledge reasoning, inference making & machine learning. For instance, we may need to keep the context of user presence for some on going (near future) activity or to infer what she might be up to.

But storing all such context information imposes several issues. First, it requires considerable amount of storage space. Ubiquitous computing systems are essentially distributed, therefore, migrating larger amount of data puts significant burden over network traffic. Secondly, query processing and data retrieval on large context repository requires significant computing resources decreasing the overall throughput of the system. Thirdly, several contexts needs to be discarded and should not be stored permanently. For example, the data with low precision, because of noise, needs to be filtered out before sensitive operations (e.g., heartbeat rate of a patient). Privacy control also prevents us from storing all information, e.g., the information that user is in washroom. Lastly, efficiency of techniques such as knowledge reasoning, inference making and machine learning depends heavily on the size of supplied data.

4 Proposed Solution

First we need to identify low precision, irrelevant and redundant context; the one that is no longer useful and remove such context information. We call this process as Garbage Collecting Context (GCC).

Secondly, we need to summarize the actual (raw) context in such a way that it is more meaningful, can be used more efficiently for reasoning, etc and takes up less storage space. We name this process as the Context Summarization (CS).

A simple analogy is human behavior towards received news. Every day, we read a lot of news in newspaper, on internet and television. But do we (need to) remember all the words and information that make up a particular activity or event? What we actually (need to) remember is some compact information about a particular event that what has actually happened. For example, Bob watches a soccer match for 70 minutes but after the match is over, he does not remember exactly what had happened in the 14^{th} minute of the game. What he actually remembers is a summary of the match like who has won the match, few ups and down, and how many goals were scored and by whom. This is very close to what we mean by *Context Summarization* that instead of storing each and every raw information, only keep summarized and meaningful context information. Coming back to scenario, after the match is over, Bob tends to forget some information, e.g., how far did the ball go when Player X kicked it and who received it. Also, as time goes by, he also tends to forget more details like a spectator had broken in to the field. This act of discarding irrelevant information is analogous to the concept behind *Garbage Collecting Context*.

4.1 Garbage Collecting Context (GCC)

GCC is analogous to the garbage collection in programming languages [5] where we identify the memory areas no longer needed by a program and free it.

GCC can be used to filter out the noise in the data, i.e., the data with low precision so that it does not affect sensitive operations. Some systems [3] provide the precision value or the probability of the correctness of sensed value which can be employed.

GCC can be used to identify and remove the context no longer needed by an application. For example, if an application is storing temperature values after every 5 minutes then it may not require raw history forever. But generally, discarding information is not considered as a good idea; therefore, here we can employ the idea of context summarization and replace the raw history with this summarized history.

Privacy control can also be dealt using GCC. In this case, certain privacy policies determine which context should not be stored and included in the system processing. For example, the location of user in private places (like washrooms) and activities during the lunch break should not be processed and stored permanently in the system.

4.2 Context Summarization (CS)

Where Garbage Collecting Context (GCC) identifies and removes the irrelevant and less significant context, Context Summarizer (CS) operates on incoming and existing context to extract useful context from the original data so that it consumes less storage space, improves the efficiency of query processing, reasoning and machine learning. Consider a temperature sensor emitting temperature value after every 5 minutes.

Table 1. Temperature values stored after every 5 minute

Time	Temp.
12:05	23 °C
12:10	21 °C
…	
15:35	15 °C
…	

Using Context Summarization, e.g., we can group (average) this on the daily basis. Another possible way could be when a day is divided into periods like morning, afternoon, evening and context information is kept for each such period (See Table 2).

Table 2. Temperature values stored for different periods of day

Date	Period	Avg. Temp	Max. Temp	Min. Temp
12/01	Morning	5 °C	8 °C	3 °C
12/01	Afternoon	10 °C	14 °C	8 °C
12/01	Evening	9 °C	11 °C	7 °C
12/01	Night	7 °C	8 °C	5 °C
12/02	Morning	4 °C	8 °C	1 °C
…				

The above example demonstrates the summarization of historical data. CS can also be applied as data is received from sensors. For example, when receiving data from audio/video sensors we can extract useful information from it. With audio, we can extract Intensity and Audio type (music, talk, telephone ring). From video sensor, we can extract Pixel percent change, Motion pattern, etc. As a result, instead of storing actual audio & video context, we can summarize and only store relevant information.

One of the benefits of performing context summarization is reduced storage space, which will result in the faster query execution and data retrieval. It will also make the data migration in distributed environment more efficient with fewer burdens on network traffic. Another important motivation behind CS is to store only the relevant context information in such a way that it is more useful for context consumers.

Reasoning about context and drawing inferences is the primary reason for keeping context in context repository in first place, and is primary tool for providing context aware services. For example, if a ubiquitous computing system knows that when Bob comes to his office in morning, he likes to check his emails, then a system can start downloading his emails when Bob enters the room in the morning. The amount and quality of input data makes reasoning engine perform more efficiently. We believe that if context summarization is done properly then it will result in less data; optimized for reasoning and inference making and machine learning

Context Summarization can either be '*Active*' or '*Passive*'. In *Active Context Summarization*, the context is summarized as it is received from the context sources; sometimes, even before it being stored in the Context Repository, e.g., the summarization of audio and video context. Active Context Summarization is usually irregular and event-based and is performed more frequently.

Passive Context Summarization is usually performed on the context already stored in the repository. The summarization of temperature (as discussed earlier) and other numerical valued contexts comes in this category. It is regular and periodic, i.e., performed in background after a regular interval or at some pre-specified time. It is performed less frequently and may consume considerable computing resources

5 Context Summarization Techniques

We have identified several categories of context information based on the similarities and nature of context. Each technique is designed for a particular category of context.

5.1 Aggregation

In aggregation, the history of context information is aggregated to generate compact and consolidated context. Numerical context types like temperature, light intensity, humidity, available network bandwidth can be summarized using this technique. In section 4.2, we have demonstrated how this technique.

Aggregation is a passive, regular and periodic type of context summarization, which works in background periodically and is performed less frequently. It removes the original (raw) context after the context summarization has been performed.

5.2 Categorization

Here we categorize different context entities, e.g., user and device profile can be categorized into user and device groups. In this way, we can track the network bandwidth utilization by some particular user group (say doctors) or by some particular device group (say PDAs) during office hours.

Categorization is passive and static type of context summarization, i.e., it is not performed frequently. It can be performed at system startup by some human or the system can learn itself to define categories. In any case, categorization supports machine learning and higher level reasoning. Unlike other techniques, it does *not* remove original context information such as existing user or device profiles.

5.3 Context Extraction

In Context Extraction, useful interesting context is extracted from continuous streams like audio and video. For example, it can be applied to video stream received from Camera, Webcam to extract features like pixel percent change, picture motion pattern (like stable, regular, irregular), etc. Similarly, audio context can also be summarized.

It is an active, irregular and event based context summarization. It can start any time with an interesting activity. Unlike other techniques, it can be triggered even before the context is stored in the repository and even discard it before storage. It results in saving a lot of storage space but may take considerable time in doing so.

5.4 Pattern Identification

Context can be summarized by identifying existing patterns in the context repository or history of activities. Location context can be summarized using this technique. Consider the location context history in the context repository (See Table 3)

Table 3. Location Context History of Users and Rooms

Time	User	Room
09:05	1	1
09:02	2	1
10:08	1	2
11:26	3	3
11:44	3	3
…		

Using pattern identification, a system may deduce the pattern of user's location during week days and come out with something as presented in Table 4.

Table 4. Pattern Identification for User Location

Time Period		User	Room	Probability
From	To			
09:00	12:00	1	1	0.76
13:00	17:00	1	1	0.83
09:00	12:00	2	2	0.67
13:00	17:00	2	1	0.89
…				

Similarly, a system can identify pattern of room occupants at various time periods. Using categorization with pattern identification, a system may also infer which user group (doctors, programmers, etc) occupies which room at different periods of time.

Pattern Identification is again passive, regular and periodic class of summarization that works in background. It is resource intensive and thus performed less frequently. On the positive side, it results in reducing considerable amount of storage space and also supports higher level inference making, machine learning and in predicting future intentions of a user or device in the specific situation. Pattern identification works on the history of context and replaces larger history with patterns of activities.

6 Proposed Model for GCC and CS

The first question, while designing and developing the GCC and CS, is should these components be part of middleware or not? We believe that making these components part of a middleware will yield us the re-usability of design and code.

We prefer designing these components (GCC and CS) as frameworks [11] so that applications only need to provide the *hotspots* (areas of specification). Hence, *Garbage Collecting Context (GCC)* can be developed in such a way that application

specific techniques for Noise Filtering and Privacy Policies can be induced even while the application is operational. For example, an application can specify, through XML, that from 1 pm to 2 pm, there is a lunch time at room X, so the location and other activities of users over there should not be monitored. Figure 1 presents the proposed architecture of Garbage Collecting Context (GCC) module.

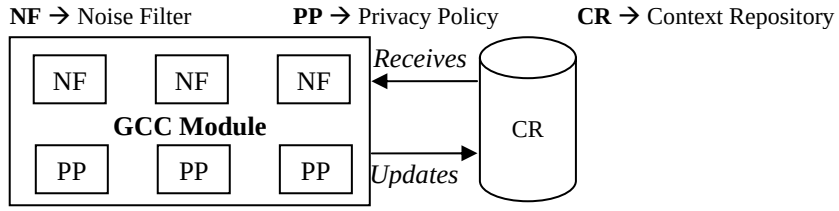

Figure 1. Garbage Collecting Context Module

GCC retrieves context data from Context Repository (CR), identifies noise (corrupted) context using Noise Filters (NF), applies Privacy Policies (PP) to remove privacy sensitive context and updates the context repository. In a particular implementation, GCC may not delete the context as it identifies it as garbage but only mark it or move such context to some other repository for some later analysis.

Context Summarizer (CS) can also be developed with framework technique. There are various context summarizer sub-modules for each different category of context, called Context Category Summarizer (CCS). Thus temperature, network bandwidth, etc can all be summarized using a single CCS. Context Summarizer (CS) is supplied context information along with Context Meta-Data (CMD). This context meta-data, usually represented through XML, specifies the category of supplied data, so that CS may decide which Context Category Summarize (CCS) should be used to summarize this context. All CCS sub-modules implement a particular interface so that CS can access each of them uniformly. Because of the framework based design of the CS, new CCS can be added and the existing CCS can be updated while the application is operational. Figure 2 shows the architecture of Context Summarizer (CS).

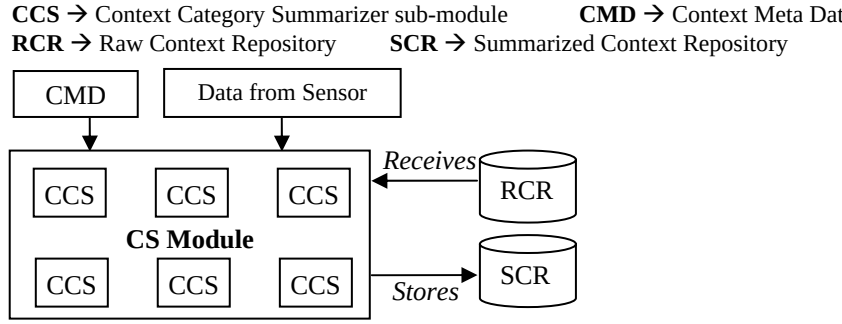

Figure 2. Context Summarization Module

7 Issues and Challenges in GCC and CS

Garbage Collecting Context (GCC) and Context Summarization (CS) have their own research issues and challenges both at conceptual and implementation level. Here, we identify several such issues and wherever possible identify few applicable solutions.

7.1 What Can Be Summarized & When The Context Should Be Summarized?

Can we summarize each and every type of context? We do not think so; we should only apply context summarization where our application specific analysis identifies some performance or storage optimization. Also, different kind of context information can not be efficiently summarized using a single method. Rather, we can form several categories according to similarities in the nature of context information and define mechanisms to summarize each different category of context data

Another important question asks what is the appropriate time to trigger context summarization? Should we start context summarization just as sensors provide contexts? (*may be useful for audio and video sensors*) or as the application processes it to some more useful form and stores in the context repository (*may be applicable for location context*) or once the context has become the history and is not directly useful for application (*applicable for temperature and similar type of contexts*) or periodically after some regular interval of time or some pre-specified time?

7.2 Performance Overhead & Other Challenges

Perhaps the foremost concern to apply these techniques is the performance cost. Do the benefits achieved by these methods justify the computing resource consumption? We believe that a proper application of GCC and CS (like those discussed in section 5) will yield the performance improvement and will not eat up many resources. In any case, the overall system should not be ceased or hung-up during the execution of GCC and CS modules, the resources (like context repository) should not be locked for noticeable period of time. But the problem is how to achieve this? We need GCC and CS only when there is considerable amount of context information, a considerable amount of context means a considerable amount of processing and resource consumption to produce useful output.

Other issues and challenges include how are we going to deal with distributed and ubiquitous nature of middleware, data repository and applications? What are security, trust and service level guarantees required for systems using GCC and CS techniques?

8 Risks Involved

Garbage Collecting Context (GCC) and Context Summarization (CS) are sensitive in nature as they directly access context information and modify it. Information is always one of the most important assets of any system and organization. In this

section, we will briefly mention about some risk factors that should be considered while developing and implementing GCC and CS techniques

Both GCC and CS result in some data and precision loss. Failing to compensate this precision lost may affect performance and overall throughput of the system badly.

Improper context summarization may make reasoning and machine learning more difficult, complicated, inefficient, incorrect and misleading instead of improving it

GCC and CS modify Context Repository (CR). Several modules of middleware and application access CR simultaneously. Such a sudden modification may be unexpected and make these modules produce unexpected results and must be avoided.

9 Conclusion

Garbage Collecting Context (GCC) and Context Summarization (CS) are new, interesting and useful research areas with a number of interesting research issues. We have presented both the benefits and risk factors involved in using these techniques and have also identified four techniques for implementing Context Summarization (CS). We have also proposed a model for implementing these concepts and identified certain research issues and challenges expected to be faced. We conclude with the fact that they are sensitive operations which must be handled carefully and applied after rigorous testing. Finally, 'to summarize and how to summarize?' that is the question!

References

1. M. Weiser, The computer for the 21st century. ACM SIGMOBILE 1999 Review
2. Dey, A.K., et al.: A Conceptual Framework and Toolkit for Supporting Rapid Prototyping of Context-Aware Applications. Human-Computer Interaction (HCI) Journal, Vol. 16. (2001)
3. Hung Q. Ngo et al: Developing Context-Aware Ubiquitous Computing Systems with a Unified Middleware Framework. EUC 2004: 672-681
4. Michael J. Franklin, Challenges in Ubiquitous Data Management. . Informatics: 10 Years Back, 10 Years Ahead, LNCS #2000, R. Wilhiem (ed)., Springer-Verlag 2001
5. Richard Jones, The Garbage Collection, http://www.cs.ukc.ac.uk/people/staff/rej/gc.html
6. Jason I. Hong, James A. Landay, Support for location: An architecture for privacy-sensitive ubiquitous computing, Proceedings of the 2nd international conference on Mobile systems, applications, and services, June 2004
7. Alex Berson , Stephen J. Smith, Data Warehousing, Data Mining, and OLAP, McGraw-Hill, Inc., New York, NY, 1997
8. D. Barbara et al., The New Jersey Data Reduction Report, Bulletin of the IEEE Technical Committee on Data Engineering December 1997 Vol. 20
9. Lin Qiao et al, Data streams and time-series: RHist: adaptive summarization over continuous data streams, Proceedings of the eleventh international conference on Information and knowledge management, Nov 2002
10. Moore, D., I. Essa, and M. Hayes, Exploiting Human Actions and Object Context for Recognition Tasks, In Proceedings of IEEE International Conference on Computer Vision 1999 (ICCV'99), Corfu, Greece, March 1999
11. Mohamed Fayad, Douglas C. Schmidt, Object-Oriented Application Frameworks, Communications of the ACM, Volume 40 Issue 10, Oct 1997