HETERGOGENOUS DEVICE CONTEXT AWARE INFORMATION DISSEMINATION

G.Kousalya[*], P.Narayanasamy[2]

**Research Scholar, Computer Science and Engineering Department*

Anna University, Chennai-600025, India.

[2]Professor and Head, Computer Science and Engineering Department

Anna University, Chennai-600025, India.

Abstract: This paper focus on device aware context delivery with middleware based support for heterogeneous client devices. Data networks are increased tremendously to support heterogeneous devices. There are myriad of new portable computing devices that each have different capabilities with regard to communication platform, communication protocol, computing power, display capabilities etc. Middleware based provisioning of various services based on the device context is proposed in this paper. Our work focuses on identifying different type of handheld devices based on the context of the device. Based on the device context (DC), the necessary services are served from the dynamic web server to the clients. The device identifier object provides the necessary information related to the context of the heterogeneous device (clients). This enables the server to provide the required information as per the context of the device. Our approach is generic in nature to support any kind of device context in an efficient way for different kind of application in a heterogeneous platform. The experimental results provide best solution for the forth-coming devices based on its context. This paper overcomes the deficiencies and limitation of existing device-aware computing by providing a suitable system and mechanism for device aware content delivery.

1. Introduction

Computing devices and applications are now used beyond the desktop, in diverse environments, and this trend toward ubiquitous computing is accelerating Dey et al [7]. Any information that can be used to characterize the situation of an entity, where an entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves is termed as Context. It is typically the location, identity, and state of people, groups, and computational and physical objects. Letting computer systems sense automatically, remember history, and adapt to changing situations is Context Aware Computing [5].

The typical user is not facing a desktop machine in the relatively predictable office environment anymore. Rather, users have to deal with diverse devices, mobile or fixed, sporting diverse interfaces and used in diverse environments. In appearance, this phenomenon is a step towards the realization of Mark Weiser's ubiquitous computing paradigm, or "third wave of computing," where specialized devices outnumber users [8]. There are various categories of Context such as Computing Context – computing information (network connectivity, communication cost, communication bandwidth, nearby resource), Device Context (differing hardware configurations) User Context -user's information (profile, location, nearby people), Physical Context – environmental information (lighting, noise, traffic, temperature), Time Context – such as time of day, week, month. Likewise Device Context refers to the differing hardware configuration and profiles of various devices- such as Laptop, Mobile, PDA, Desktop PC, etc. In this sense it often means of a Pervasive Computing Environment.

Location is a very important component in implementing a Context aware infrastructure. In this sense there are many beacon methods available such as: Using Satellites (the famous GPS positioning system), Using mobile phone network (using the identity of the current BTS), Short range positioning beacons (such as Bluetooth, WiFi, RFID, IR,etc,) In current situation there are simple cell phones with messaging capabilities; smart phone with increased capabilities and Personal Digital Assistants (PDAs) with wireless communication capabilities. Within a particular device category, such as cellular phones, there are typically more than two hundred different models available in the market from a wide variety of manufacturers. Each of these devices typically has a different set of capabilities, features and particular implementation of communications protocols application interfaces and operating systems.

Personal electronic devices such as personal computers, PDAs, wireless telephones and pagers have become prevalent in recent years. These devices allow mobile computing by communication over wireless and/or wire line networks. The networks provide interconnection of these mobile devices with information sources as well as other similar devices. Commonly, the networks include communication over the Internet.

Typically, mobile computing devices include some form of user interface (UI). The nature of the user interface on a device is a function of the computing power and the hardware represented on the device. For example, a wireless telephone has limited computing power and a relatively small graphical user interface (GUI) with limited graphics and user interface capability. Conversely, a notebook personal computer has relatively extensive graphics capability, user interface capability and computing power. As the popularity of mobile devices increases, the main device platform for applications may shift from desktop personal computers to mobile devices.

There are several key limitations with the GPS positioning methodology. For instance, GPS is unsuitable for any indoor tracking activities; it is only effective for outdoors since the signal strength and accuracy is weak during indoor. The processing delay and the compulsion of using a GPS receiver at each desired client device is not only a technological overhead but also an issue cost wise.

WLAN is one of the promising technologies that come in handy for the stated scenario (indoor context aware computing). It is much suitable for implementing an application that is accessible by various hand held devices within a building. The other technologies like RFID and BlueTooth are also much prominent that rely on radio transmission. Of the above-mentioned technologies, RFID is not suitable as it is workable only in a close proximity. WLAN seems to be effective, but still it has some limitations. For instance, not all hand held devices lend support for WLAN support. Another important parameter is the coverage area. The coverage area is good when a whole building is considered.

But what if, the requirement is that there has to be provision of various services within the Organization building.

One significant difficulty with developing applications for mobile devices is device heterogeneity. The server application has to identify the exact device context and hardware profile for the currently communicating device client. It is necessary to vary the response service according to the differing device context. Due to variations in display size, display resolution, command input methods and GUI libraries, an application developer may have to re-design and re-implement applications for each device platform. With the large number of different mobile devices currently available, or coming into the market, re-design and re-implementation may be an ongoing, labor intensive and cost prohibitive endeavor. Not all types of client devices can access the intended service. Even when supported, the intended service that the client requires comes at an unaffordable cost.

The proposed work focuses on the development of device aware applications that support for heterogeneous client devices. The context aware environment has been narrowed down to indoor. In this sense the proposed architecture is flexible enough to determine the device context of the client device and provide service accordingly. As the work is focused on indoor context aware computing, the usage of positioning technologies becomes a key issue. Focusing on indoor location, the number of existing typologies to perform location dramatically increases: ultrasounds, radiofrequency, artificial vision, magnetic field sensors, inertial sensors, etc.

Context-aware computing offers the promise of significant user gains - the ability for systems to adapt more readily to user needs, models, and goals. Computational systems have clearly lacked the vast amount of state data required to handle even relatively simple customizations, leaving systems without the ability to handle an individual's level of desired social nuance and control[6].

Collaborative systems, for example, with their need for fine-grained control over personal data could profit enormously from additional context.

Mark et al described a context-aware computing offers the promise of significant user gains - the ability for systems to adapt more readily to user needs, models, and goals. Dey et al presents a masterful step towards understanding context-aware applications. Mark et al examine Dey et al in the light of privacy issues -- that is, individuals' control over their personal data – to highlight some of the thorny issues in context-aware computing that will be upon us soon. We argue that privacy in context-aware computing, especially those with perceptually-aware environments, will be quite complex. Indeed privacy forms a *codesign* space between the social, the technical, and the regulatory. We recognize that
Dey et al is a necessary first step in examining important software engineering concerns, but future research will need to consider how regulatory and technical solutions might be co-designed to form a public good.

We recognize the utility from the user's perspective of integrating the behavior of a number of software applications. With the emergence of Web-based applications and personal digital assistants (PDAs), there are even more opportunities to provide integration of software applications. There are some limitations, however, to the current approaches for providing this integration. These limitations impact both the programmer and the user.

2. Related Works

Andrew et al (2007) [9] developed an application for pervasive computing presents a number of challenges to the software engineer. In particular, application adaptation based on context such as environmental factors, device limitations and connectivity, requires the programmer to handle a complex combination of factors that manifest themselves throughout the application. Their paper presents an approach to managing such complexity based on a combination of aspect-oriented development techniques, and model driven development. In this position paper, they discuss the potential to address both observations (cross-cutting behavior and inability to reason out application semantics from the technical specifics) by combining aspect oriented software development (AOSD) with model driven development (MDD). They focus on AOSD, but the device profile collection mechanism model is yet to be enhanced.

The article by Dey et al [11] observed that there have been numerous context-aware efforts, which are mostly demonstrations of various technologies for sensing, capturing, and presenting, and interacting with information in the physical context of people's work activities and demonstrations of various applications on user mobility and location-dependent information. The thrust of their work is to systemize the complex task of designing and developing such applications. They first define context in a very general way as any information that describes the setting of the users' activities, with emphasis on the physical attributes: time, place, people, physical artifacts, and computational objects. They identify the difficulties of handling context due to the myriad of sensors and devices involved, its distributed nature, the need to interpret the data into useful abstractions, and the need for persistent storage and access of context information. They address these issues by a conceptual framework consisting of context "widgets" (for capture and interaction), interpreters and aggregators, services, and resource discoverers. This framework is embodied in their Context Toolkit. They then illustrate several applications and how the toolkit supports their implementation, and, in one case, how the framework can be the basis of a systematic software design process.

Hong and Landay **[14]** explored the idea of context-aware computing from the perspective of a service infrastructure, a pervasive middleware approach in which much of the work of collecting and processing context information can be decoupled from the application itself. They argue that this approach provides a number of benefits that will be needed if context-aware computing is to be adopted on a broad scale.

Smailagic and Kong [11] proposed a smart spaces which interact naturally with the user. For these interactions to occur, the system must be aware where the user is. Since location is a very important component in context aware computing, their efforts have focused on developing a Location Service. Their implementation is based on signal strength and access point information from the IEEE 802.11b Wavelan wireless network that covers the entire Carnegie Mellon campus. This is in contrast to a GPS-based approach, which tends to have poor indoor accuracy and also requires each client to be equipped with a GPS unit that adds weight and consumes power. The goal was to develop a power efficient, scalable and private Location Service that can be used by a variety of clients including wearable and laptop computers. They have explored two distinct approaches: client-centric and server-centric. In the client-centric approach, the signal strengths of multiple Wavelan access points are obtained by a client. The server-centric approach is less accurate, but much simpler. In this approach, the identity of the access point currently servicing a client is recorded on a central server and made available to other clients.

Damião et al [12] introduced a service-oriented architecture for context-aware applications named Omnipresent. Omnipresent is based on well-established standards such as Web services, and those from the Open Geo Spatial Consortium. Omnipresent offers several services including map presentation, routing, advertisement, and also works as a reminder tool. Users may use Omnipresent either by mobile devices or Personal Computers.

Christine et al [13] proposed the highly dynamic multi-agent systems; the multitude of devices provides a varied and rapidly changing context in which agents must learn to operate. This paper presents a protocol for mediating the context-based interactions among mobile agents. They present an implementation and show how it facilitates information exchange among mobile application agents. They also provide an analysis of the tradeoffs between consistency and range of context definitions in highly dynamic ad hoc networks.

Ubiquitous computing environments integrate a large number of heterogeneous devices, which convey an increasing level of complexity when developing ubiquitous applications [14]. A solution to this problem resorts to the use of a software abstraction layer, known as middleware, which encapsulates the underlying elements of the environment and offers unified and standardized access to applications which need to make use of the resources of the environment. Moreover, a middleware layer can also provide high-level built-in services, such as context management services.

We will consider three fundamental aspects of this type of generic smart environments: acquisition of contextual information of the device, control of the environment and user interaction with the environment.

3. Device Aware Information Dissimiation Framework

The Server Component plays a vital role in establishing a device aware environment. The architecture of the Server is described in the figure. 1.1. The two important components within the server are BlueTooth sensor and the device aware component. BlueTooth sensor is required to listen for any incoming connection request and to establish connection with the requested client device. The service range, data rate and the hardware specification of the BlueTooth sensor varies according to the intended service that is to be provided. Device aware component is the central controlling component within the server.

The proposed work suggests the usage of effective but cheaper positioning system in the form of BlueTooth. The advantages of BlueTooth implementation over the existing/ other means is many fold. First of all it is much cheaper when considering all other means. The general availability of the BlueTooth support in almost all mobile and handheld systems is an added advantage. The wide range of coverage (from small coverage such as few to 10 mts. to as high as 100-500 mts.) enables for the flexible configuration of the desired infrastructure. This means the setup shall be made for a entire building as a whole or service shall be differed with respect to differing locations within the same building (using smaller range BlueTooth sensor).

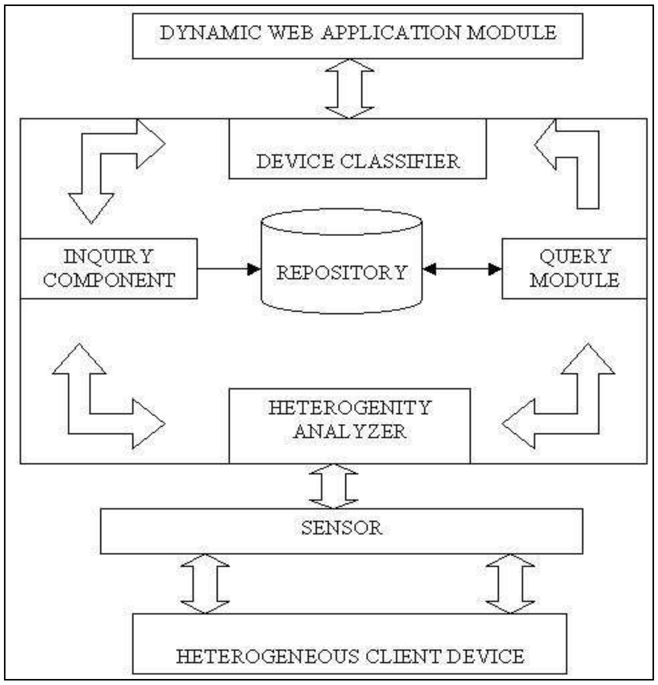

Figure 1.1 Device Aware Information Dissimiation Framework

To overcome the difficulty in device heterogeneity, special focus is made in the implementation of the server application. The device context is identified first of all before any service response is delivered. Using the identified device context the service response is effectively varied in order to suit the end user client device. Also in the development of the client applications, the usage of the Java-based component enables a much simpler shift in providing a common application that supports for varying hardware configurations among the client devices.

The responsibility of the Device aware component is to determine the hardware profile of the device client and thereby the device type of the requesting client device. In order to maintain the history of connections made and to fasten the connection setup during later visits the Device aware component maintains a database. Necessary updates are made to the database periodically. Device Inquiry is invoked by the Device aware component when a particular client device establishes connection for the first time. Device Inquiry involves discovering of all BlueTooth enabled devices that is within the range of the BlueTooth sensor. The Device aware component also is responsible for creating and starting an appropriate client thread with respect to the device type identified formerly.

The heterogeneous client devices available within the service range of a particular Server which has been incorporated with BlueTooth sensor. All the clients are BlueTooth enabled devices. Once the Server has been started properly, the server is pushed in to the listen mode waiting for any incoming requests for connection establishment. In order to detail the work flow, the entire process has been classified into three broad steps.

The first step involves the connection setup between the client device and the server component. When a particular client (say Mobile) wants to communicate to the server and thereby get the intended service first it makes a request for communication via BlueTooth that is residing within its device. Now since the server is in the listen mode it detects the request made by the client. In reply the server acknowledges the request with a corresponding connection response. Once the conncetion setup is over between the client and the server, the client device returns a connection object (in this case a Remote Device object) to the server. The server uses this remote device object for any further communication with the client device. The server then utilises the remote device object to access the BlueTooth IP address of the client device. *The above mentioned identifier is unique in the sense each and every bluetooth enabled device can be spotted distinctly*. Thus obtained bluetooth IP address is then passed on to the Device aware component for further usage. This completes the first step of the Work flow which is shown in figure 2.

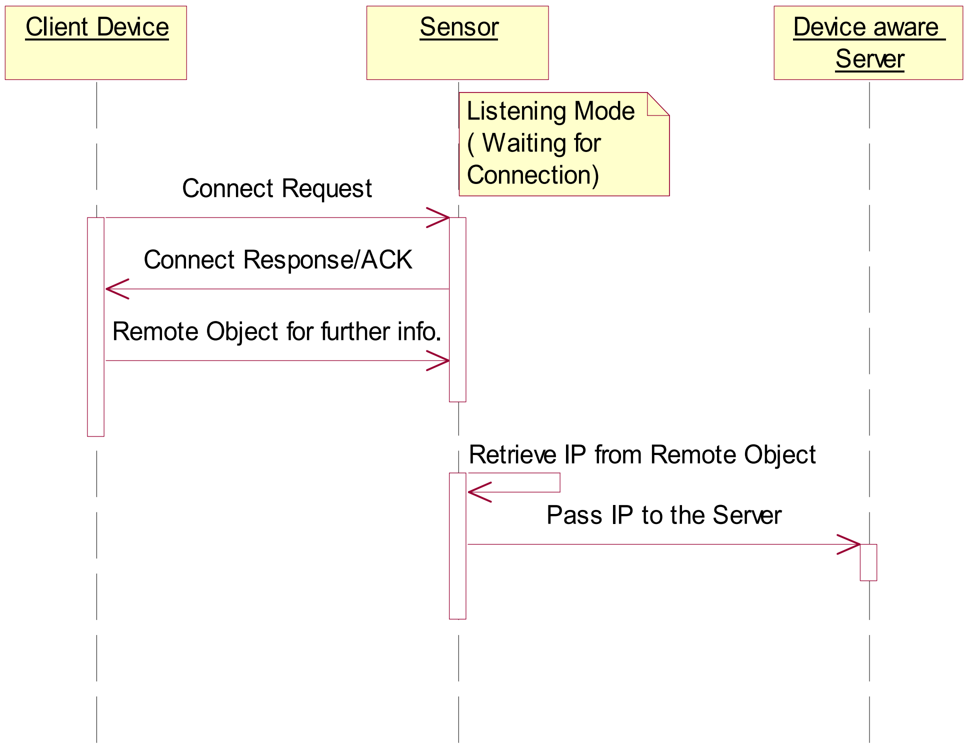
Client Device
Sensor
Device aware Server
Listening Mode (Waiting for Connection)
Connect Request
Connect Response/ACK
Remote Object for further info.
Retrieve IP from Remote Object
Pass IP to the Server

Figure 2. Connection establishment

Second and third steps are based on the two available situations that may prevail for any communicating client device. The first situation results when the communcating client device is a previous visitor and that the device context of that particular client device has been captured and stored by the Distribution Server (DS) is shown in figure 3. In this case the Device aware component first queries the DS using the bluetooth IP address of the client device as the key. The DS responds it with a result. If there comes a non-null value in the result then that means the client device is a previous visitor.

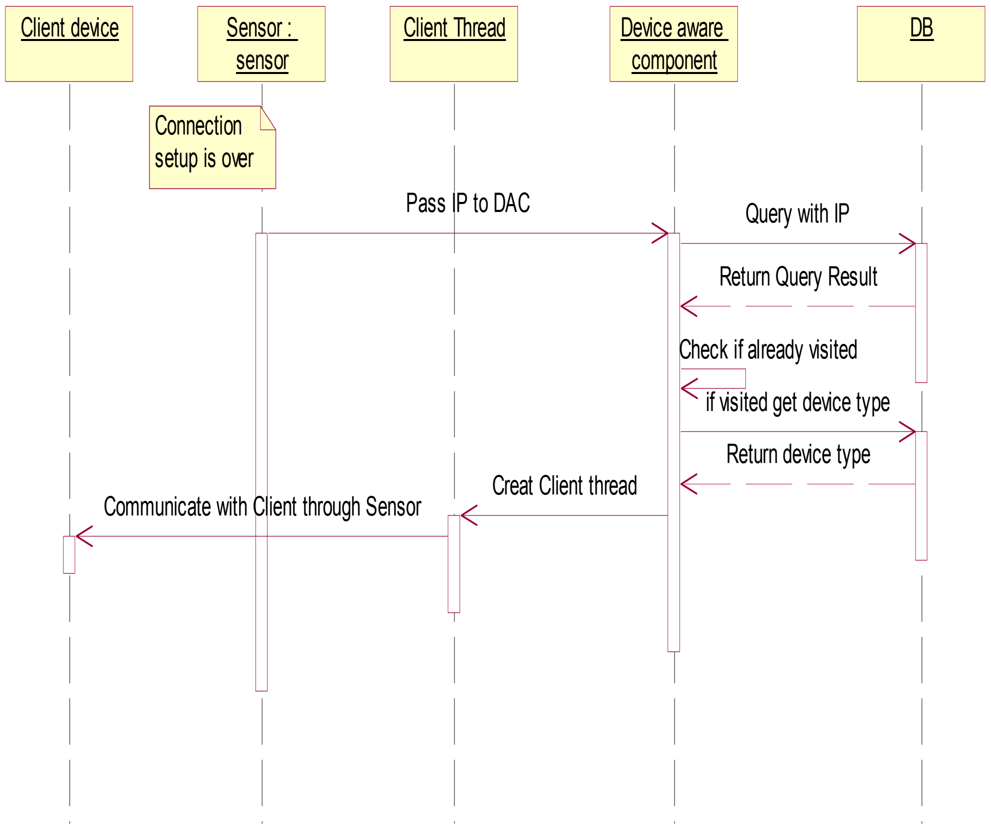

Figure 3 Situation 1- Already visited client

The result returned by the DS constitutes of the entire device context of the bluetooth hardware. One of the key parameter in this device context is that the device class type. Using this type value the device aware component determines the class of device that the client belongs to (Mobile, Laptop, PDA, Desktop PC,etc.,.). Once the client device has been calssified the device aware component creates and starts a corresponding thread with repect to the hardware profile of the communicating device. The reason for creating independent threads for each kind of hardware profile is that the intended service cannot be provided as such to all type of devices. For instance the display resolution and configuation differs between a Laptop client and a Mobile client. Likewise the data rate that is supported by a Mobile device farby differs from its Laptop counterpart. This means that much *crispier communcation* has to be made for a Mobile while a much richer content shall be provided for a Laptop or a Desktop client.

Another situation results in when the communicating client device is a new comer is shown in figure 4. In this case a query to the DS with the bluetooth IP of the client would return a null value. Then the Device aware component invokes the Device inquiry component.

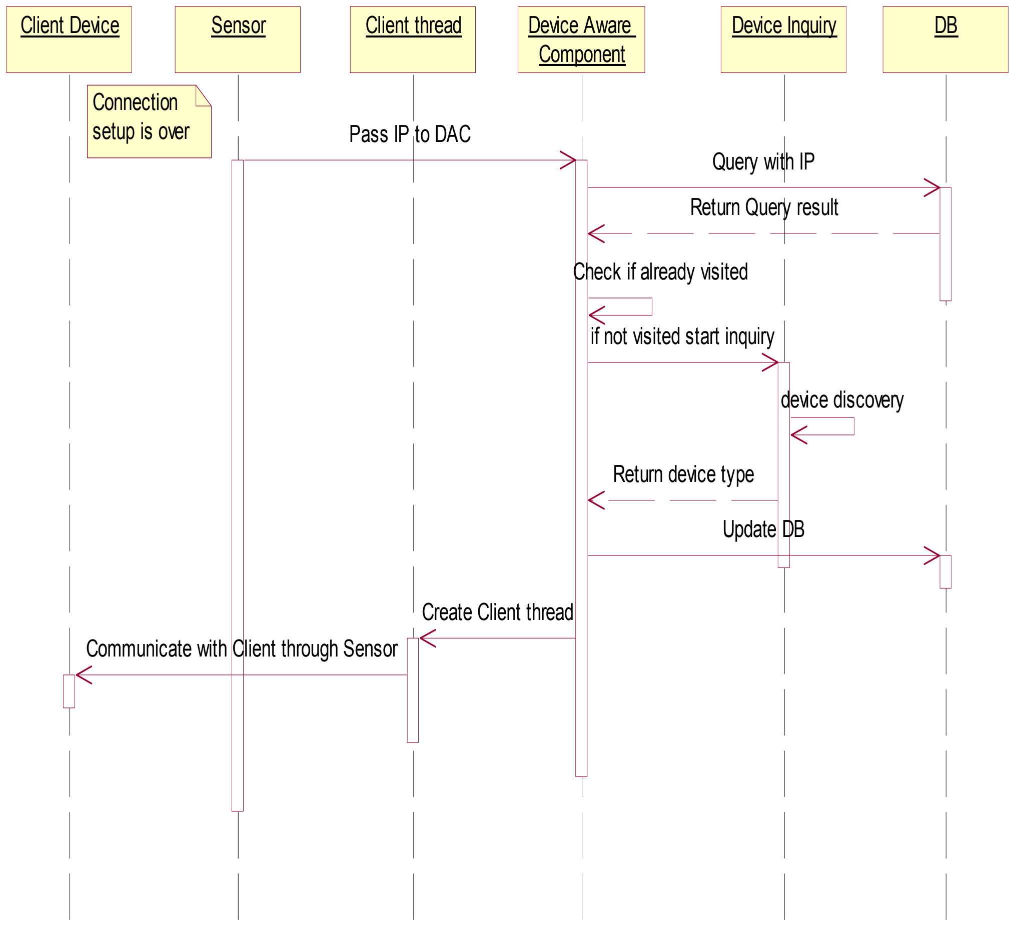

Figure 4 Situation 2- Newly visiting client

During Device Inquiry all the bluetooth enabled devices that are within the service area of the bluetooth sensor are detected. On each device that has been discovered the IP address of it is matched with the IP got from Client's remote object. If a match occurs then the Device class object is used to get the device type of that particular client device. This new record is appended to the existing database maintained by the DS. After determining the device type of the client device the corresponding client thread is created and started to handle further communication between the client and the server.

5. Implementation Details

The device aware component is developed using J2EE and with blue Cove. BlueCove is a Java library for Bluetooth (JSR-82 implementation) that currently interfaces with the Mac OS X, WIDCOMM, BlueSoleil and Microsoft Bluetooth stack found in Windows XP SP2 or Windows Vista and WIDCOMM and Microsoft Bluetooth stack on Windows Mobile. BlueCove can be used in Java Standard Edition (J2SE) 1.1 or newer [3][4]. We carried out a set of experimentations in a platform. Different kinds of devices are used to access the content: a Pocket PC (iPAQ 3600) running under Windows CE and connected through the wireless network, a laptop computer using both the wired and wireless connection and a personal computer that uses only the wired network and a mobile phone Nokia N series, Sony. Most of the adaptation methods are implemented in JAVA. It provides an automatic data selection feature, allowing for the transfer of data between (Java and non-Java) applications via a clipboard-style interface, and supports the use of reusable software components .

RemoteDevice Object

RemoteDevice object plays a key role in acquiring context information of the particular device client that is making the connection request. This object shall be created on demand just after the connection is established between the server and the device client using the *getRemoteDevice* method available under the *RemoteDevice* class. The main purpose of this *RemoteDevice* object is that to determine the Bluetooth address of the communicating device client. The Bluetooth address is unique for each Bluetooth hardware. This address is determined using the *getBluetoothAddress*() method of the *RemoteDevice* class. Other than the essential Bluetooth address various other information such as Friendly name of the device (remote device name) are provided by the *RemoteDevice* object.

Scope: Heterogenity analyzer, Query module, Inquiry component.

DeviceClass Obejct

DeviceClass object comes into play when the device client communicates with the server application for the first time. There would be no information stored in prior about the client device by the Distribution Server(DS). *getMajorDeviceClass()* method under the *DeviceClass* class is used to determine the device type of the communicating device. This object is automatically passed as an argument to the *devicediscovered* function during device inquiry process.

Scope: Inquiry component, Device classifier.

StreamConnection Object

StreamConnection object is the first object that gets created once the connection requested by the device client is accepted by the server module. All socket connections that are further required for the communication between the server and the device client are created from the *StreamConnection* object. This object is even used in the creation of the *RemoteDevice* object.

Scope: Entire stretch of communication; until end of connection.

HETEROGENITY ANALYZER

Heterogeneity Analyzer is responsible for initial connection setup operations between the server and the client device. However the main function of heterogeneity analyzer concerns about the analysis of the device context of the end device and thereby varying the service response accordingly. It serves as the control unit within the entire Device Aware Component (DAC). *RemoteDevice* object is extensively used in this part of the DAC for the determination of the Bluetooth address of the client device. Heterogeneity analyzer delegates further processing to Query module and the Inquiry component depending on the situation whether the client device is a new comer or not.

REPOSITORY

Repository is the database that is maintained by the Distribution Server. Both the Inquiry component and the Query module communicate with the Distribution server either to query or update the underlying database. Entire history of visited client devices is maintained in this repository. All the information regarding the client device such as Device name, Bluetooth address, Device type, last visited date and time, frequency of visit, etc., are maintained.

QUERY MODULE

This module is invoked by the Heterogenity analyzer in order to query the database to determine the communicating device has already visited or not (i.e old user or new user). Query module first extracts the Bluetooth address from the *RemoteDevice* object of the end device. Then it uses this address as the query key and queries the database. The returned result from the repository is further analyzed. If the returned result is a non-null value then the device had already visited and communicated with the server before and that its information had been stored in the repository. If this is the case the record set is passed on to the Device classifier component for further processing.

INQUIRY COMPONENT

Inquiry component is invoked by the Heterogenity analyzer when the result returned by the repository to the Query module contains null value. This means that the device client is communicating with the server for the first time. In this case, Inquiry component first starts up the Inquiry process automatically. For each discovered device *deviceDiscovered*() method is called. Within this method *RemoteDevice* and *DeviceClass* objects are used to match the current device client. Once a match is formed, corresponding device type is determined using the *DeviceClass* object. Then this new information is updated in the repository maintained by the Distribution server through a simple QueryUpdate. Once this update is commited, the control is transferred to the Device Calssifier component.

DEVICE CLASSIFIER

Device Classifier gets control either from the Query module or the Inquiry component according to the situation. The main function of the Device classifier is to identify the hardware context of the device client. The *DeviceClass* object is used here to determine the device type of the device. Once the type is determined, the service response is achieved by creating a suitable client thread (Application Thread) and starting the same thread. After the suitable Client thread is started up all the further service(s) are taken care by that thread.

The entire connection is based upon the Stream connection and Socket level communication between the server and the client device through the Bluetooth sensor.

6. Performance Analysis

The performance of the system is measured by measuring the individual components of the system. As the project focuses on high throughput to the end-user while maintaining the accuracy of the provided information, it is necessary to measure the performance of the system. Following table lists the various performance tests that were performed and the outcomes of them.

The Device Inquiry is done for discovering the type of the device. To improve the performance the Device Inquiry is not done for each time, instead the inquired device details are maintained in a database. When the same device requests for service next time its details are fetched from the database instead of inquiring.

Table 6.1 Performance factors

Task Performed	**Response Time**
Server startup	15 sec
Device Inquiry	10 sec(avg.)
Mobile client-first time	14 sec(avg.)
Laptop client-first time	15 sec(avg.)
Mobile client-already visited	4 sec(avg.)
Laptop client-already visited	5 sec(avg.)
Multiple clients simultaneously	5 sec each(avg.)
Turn around time for search	500 msec(avg.)

7. Conclusion

In this paper, we presented a context-based adaptation infrastructure for mobile devices. As the adaptation is applied on the content, we have proposed a device independent content model that facilitates the adaptation processes and provides genuine universal content delivery in different contexts. The model considers the adaptation at the presentation level based on an important aspect related to the content semantics as a hierarchy of elements. In addition, we have taken advantage of the client device profile scheme description and Web services in the context management and for querying and inquiring process. As we have shown, going toward a device independent content model facilitates the generation of adapted presentations for small devices. Furthermore, querying and exchanging only context fragments with a direct relation with content adaptation can really increase the performance of the client deivces.

ACKNOWLEDGEMENT

This work is supported by a grant from Security Engineering Research Center of Ministry of Knowledge Economy, Korea and Hannam University, Korea.

References

[1] Boudreau T, Greene S and Woehr J, 'Net Beans', O'Reilly and Associates, 2002.
[2] Edwardes A, Steiniger S, Neun M and, 'Foundations of Location Based Services', University of Zurich., http://www.geo.unizh.ch/publications /cartouche/lbs_lecturenotes_steinigeretal2006.pdf
[3] Jewell T and Chappel A. D, 'Java Web Services', O'Reilly and Associates, 2005.
[4] Mahmoud H Q, 'Wireless Application Programming with J2ME and Bluetooth', O'Reilly and Associates, 2002.
[5] Moran T P, Dourish P, 'Context-aware Computing' A Special Triple Issue of Human-Computer Interaction', Lawrence Erlbaum, 2002.

[6] Ackerman, M. S, 'The Intellectual Challenge of CSCW: The Gap Between Social Requirements and Technical Feasibility'. Human Computer Interaction, 15, pp.179-203.

[7] Dey, A. K., Salber, D., Abowd, G. D. ' A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications' Journal of Human-Computer Interaction, 2001.

[8] Weiser M,'The computer for the 21st Century', Scientific American, 265(3), 66-75.

[9] Andrew Carton, Siobhán Clarke, Aline Senart, Vinny Cahil, 'Aspect-Oriented Model-Driven Development for Mobile Context-Aware Computing', in the Workshop on Software Engineering of Pervasive Computing, Applications, Systems and Environments (SEPCASE) at ICSE 2007.

[10] Hong, J.I. and J.A. Landay, 'An Infrastructure Approach to Context-Aware Computing'. Journal of Human-Computer Interaction (HCI), 2001. Vol No. 16 (2-3).

[11] Smailagic, A. Kogan, D,' Location sensing and privacy in a context-aware Computing environment', IEEE journal of wireless communications Vol. No. 9 2002.

[12] Damião R. Almeida, Cláudio S. Baptista, Elvis R. da Silva, Cláudio E. C. Campelo, Camilo P. Nunes, Bruno A. D. da Costa, Wilkerson de L. ndrade, Jairson M. Cabral, 'Using Service-Oriented Architecture in Context-Aware Applications', VII Simpósio Brasileiro de Geoinformática, Campos do Jordão, Brasil, 20-23, 2005, INPE, p. 15-29.

[13] Christine Julien and Gruia-Catalin Roman ,' Supporting Context-Aware Interaction in Dynamic Multi-agent Systems', Springer , Lecture Notes in Computer Science , Vol.No. 3374/2005.

[14] Zigor Salvador, Mikel Larrea, Daniel Cascado, José Luis Sevillano, Roberto Casas and Álvaro Marco, 'A Middleware-based Approach for Context-aware Computing', UPGRADE, Vol. VIII, No. 4, 2007.