Some parts of this report have been submitted to the Third International Workshop on Wireless Information Systems (WIS), Porto, Portugal, April 13-14 2004
In conjunction with the 6th International Conference on Enterprise Information Systems (http://www.iceis.org)

The Mobile Hanging Services Framework for Context-Aware Applications: An Experience Report on Context-Aware VNC

Evi Syukur, Seng Wai Loke and Peter Stanski

School of Computer Science and Software Engineering
Monash University, Australia
900 Dandenong Road, Caulfield East, Victoria 3145
{evi.syukur, swloke, peter.stanski}@csse.monash.edu.au

Abstract.
Context sensitivity is particularly useful if we want to give mobile users an experience of responsive and attentive services depending on their current contexts. However, to support proactive context awareness in mobile environment requires the sufficient accuracy of location positioning system and spontaneously way of downloading and executing the context application on the mobile device. In this paper we discuss some of the issues and the necessity of adding context awareness to an existing application. In order to exploit the benefit of having context sensitivity in this field, we propose a comprehensive model, design and implementation of a Mobile Hanging Services framework. Our system aims to provide a generic mobile context aware framework that can be adapted to any existing traditional or non-traditional application. Besides that, it also provides an enhanced policy that governs the execution and termination of the service.

1 Introduction

Context sensing has the ability to usefully adapt the services or applications to the user's current situation, intention or environment. This would enable the user to receive a relevant set of services that fit his/her current context, instead of a barrage of irrelevant services. This sensing ability that can be found in context aware applications distinguishes them from existing traditional applications. Here, "traditional application" refers to a primitive stand-alone application that does not have a context sensing ability. One traditional application that we will discuss in this paper is the Virtual Network Computing (VNC) system.

VNC is a teleporting application that allows a user to connect and access his/her desktop information from any nearby machine in the current location. By adding context sensitivity to a VNC application, we can certainly maximize the user's experience in using VNC. In this case, a user can specify in his/her profiles where and when a VNC application should be started or terminated. Besides adding context-awareness to traditionally designed VNC applications, it is also useful to develop mobile code to access the VNC service. This code will be downloaded to the mobile device once the user is in the context where such code is relevant. The ability to, in an ad hoc fashion, download and execute mobile code on the device where the code can be used to control the VNC application (such as selecting the target device on which to display the VNC terminal or stopping VNC) gives the user convenient control over the application.

There are some challenges associated with developing context aware applications in a mobile environment. One challenge is to have a location positioning system that can determine the user's current location accurately. Another challenge is that the system needs to proactively discover services that fit the user's current context as well as spontaneously deliver and execute the relevant services on the user's mobile device. The third challenge is to create a generic mobile context framework that can support many different applications. Lastly, the mobile framework also needs to exploit policies or rules that can help the user to constrain or restrict the behaviour of the service (e.g., a user can specify when and where the service needs to be activated or terminated).

The research presented in this paper attempts to tackle the above issues by introducing the idea of Mobile Hanging Services (MHS). The MHS provide a generic mobile framework that can be incorporated into any existing traditional application such as a stand-alone VNC application, Media Player application, etc. Apart from being generic, they also explore context-sensitivity and mobile code in order to provide useful services for the user with minimal or no effort for service set up prior to use [16]. Besides that, MHS also allow a user to specify the policy that can govern the service execution.

In our definition, "context" refers to a set of parameters or conditions that can be used to trigger the relevant services with respect to the user's current situation. Contextual information can include location, time, a user's intention, current activities, history file, device resources [2]. Some other authors including those of [11, 12, 14] have explored much richer contexts that involve a variety of different physical sensors in the environment e.g., sound, temperature, light, touch, movement and so on.

Some work has also been done in [10] that presents a dynamic conceptual model of context that supports cognitive activities other than just a location, time and some other parts of physical contexts. Currently, contexts in our work comprise a user identity, location and time. A location context is represented by an indoor logical model such as room location. Sensing the user's context, the system computes a list of useful services and this list of services is then dropped into the user's mobile device. As the user selects the particular service name on the mobile device, the highly compact (with respect to limited device resources) mobile code that provides control for the service is then downloaded.

The rest of this paper is organized as follows. In section 2, we outline the MHS system and discuss how to make traditional applications context-aware (with VNC as the main example). In section 3, we present the user's perspective on context-aware VNC. In section 4, we briefly discuss the evaluation of the system. In section 5, we discuss related research and conclude in section 6.

2 MHS for Context-Awareness

MHS uses the Microsoft .NET Compact Framework technology that natively supports XML Web service calls. Our system consists of users with handheld devices and a Web service that determines the location of a user, which is published via the system. This section gives a high-level description of these parts of the system, and how the parts interact. The system architecture is illustrated in Figure 1 below.

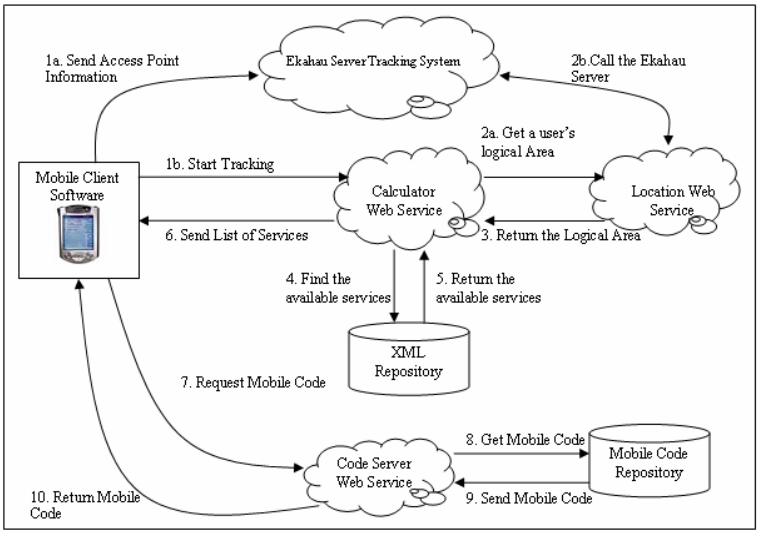


Figure 1: System Architecture of MHS system

Four of the main components of the system are discussed in the following sections:

a. Mobile Client Software. Users with mobile devices run software that continually polls a central Web service to discover the services that are available at the user's location context. When the user selects a particular service, the mobile device contacts the central Web service and downloads an application for interacting with the selected service. The mobile client software also caches downloaded applications. The cache contains code and metadata describing applications. Hence, if a downloaded application is running, its cache code also exists in the memory.

b. Location – Web service. To realise location-aware services, this system employs the current release of the Ekahau Positioning Engine (EPE). The EPE is an indoor positioning system that keeps track of a user's location based on signal strength measurements. It also supports devices such as wireless PDAs, laptops and any 802.11b-enabled devices [8].

c. Calculator Web service. In our work, the Calculator Web service computes a proactive service using a combination of user, location and time contexts. At the specific time and location, users get different types of services. Our current implementation has not incorporated other types of context parameters (e.g., history, user's intention, user's device resources) in deciding the combination of services that will drop into the user's device.

d. Code server Web service. Within our system, we employ the Web service as a method invocation to retrieve a mobile code application that matches the service name and this service method then returns the particular mobile application to the client device.

The following paragraphs describe each of the steps in Figure 1 above:

1a. Send Access Point Information. Once the Ekahau mobile user device is switched on, the EPE server then starts tracking the position of the mobile client.

1b. Start Tracking. Our system provides a login mechanism to the Mobile Hanging Services. The user needs to enter the credentials information such as a user name and password. The system then validates these credentials against the user's information, which is stored in XML database. The system will only redirect the user to the main service form, if all information that he/she enters is valid. If the user is valid, the system then invokes a Web method of the Calculator Web service called "Start Tracking" by passing the IP address of the device.

2a, 2b and 3. Get a User's Logical Area, Call the Ekahau Server and Return Logical Area. The Calculator Web service then continues to invoke the "get logical area" Web method of the Location Web Service and again passing the IP address of the device to this Web method. The Location Web service then fires the Location Listener Application on the Ekahau Server. The Listener application then continuously listening to the mobile client's movement. Finally, this Web method returns the most accurate user's logical area to the method caller (e.g., Calculator Web service).

4 and 5. Find and return the available services. Once the logical area is returned, the system then searches for the available services based on the current date and time that match with the user that is currently logged on and the current location. In this implementation, we store information regarding the mapping between contextual information (the user, time, location) and services available in the XML database. If there is a service associated with the current date and time, logical area and a user, the service information (service name and description) is then returned.

6. Send a list of Services to the Mobile Client. If the services are found, a list of services will then be sent to the mobile client. The mobile client application then displays these set of service names.

7, 8 and 9. Request a Mobile Code, Get and Send a Mobile Code. When the user chooses a service from this list, the code server Web service is contacted to provide code for invoking the selected service.

10. Return a Mobile Code to the Mobile Client. This returns the mobile code applications to the client device. Upon its arrival, the mobile client application then loads and

processes this service application, finally executing and displaying the service interface on the mobile device.

2.1 Adding Context-Aware Capability to Traditional Applications

In this section, we focus more on how to make an application context-aware in the sense that it appears to react intelligently and proactively to its surrounding contexts rather than being reactive. One sample traditional application that we will focus on is VNC. VNC can be considered as a simple remote display protocol that allows a user to access his/her desktop information on the nearest machine without requiring the user to carry any additional device or hardware. VNC also supports sharing information by allowing a single desktop to be accessed by several machines simultaneously. In addition, the VNC system is an open source application, which was initiated by a group of people from Olivetti & Oracle Research Laboratory (ORL). The researchers from ORL aim to develop a platform independent and a thin client system of remote display protocol [5].

The VNC system itself consists of two main parts: VNC server and VNC viewer. The VNC viewer also refers to a VNC client that acts as a display machine that the user interacts with i.e., passing the user's machine IP address. The VNC server refers to the end point where changes to the framebuffer of a user's desktop machine normally takes place. In order to be able to connect to a user's desktop machine (i.e., machine B), a user from a target machine (i.e., machine A), needs to pass in machine B's IP address to the VNC viewer on machine A. If the VNC system can find and connect to the VNC server on the specified machine (machine B), then the machine A will automatically bring up the machine B's desktop information. The teleporting system across the platform can be easily done using VNC. However, traditional VNC is pretty much just like a static stand-alone application that will not perform anything until there is a request from a user. To some extent, we can improve the user's experience in using VNC by adding context aware capability to it.

Using our MHS framework, we can add context sensitive behaviour to almost any existing traditional application. Our system permits a remote Web service call from a client device to a server or vice versa and from a server to the desktop machine. Then, using .NET Remoting permits remote execution of a process on a specified target machine from a Web service call. Hence, with the addition of .NET Remoting mechanism on top of our existing framework and having the extra policy file for the traditional application that specifies when and where to start the application, we can control the way the application is executed on the target machine: the system starts the VNC service as soon as the user enters B3.50 room at 3PM and stop the VNC service if the system detects that the user has walked out of the room.

We also developed mobile code downloaded onto the user's device to enable control of the VNC service. Once the code is downloaded and running, the user will be able to see a service interface where the user can select the target machine name that he/she prefers and can also terminate VNC execution on the target machine. We can also certainly make the VNC application react more "intelligently" by having a policy that specifies when and where to automatically start and stop the VNC service application in response to context changes.

In Figure 2a below, we illustrate in more detail how to start the VNC application on the target machine:

1. A user selects the target machine name. As soon as the VNC Remoting application is displayed on the mobile device, the user will be able to see a list of target machine names that they can select. Here, we assume all the target machines are in the idle state (i.e., none of the users have teleported to this machine). We store information regarding the mapping between the room and list of machines in an XML database. By knowing a user's current location, our system will be able to search and get a list of available machines for that particular room.

2. Get the user's machine IP. Once the user selects the target machine name that he wants his desktop information to be displayed on, our system will then get the user's desktop machine IP. For this implementation, we also store a mapping between a user and his desktop machine IP in the XML file.

3, 4 and 5. Search for the user's machine IP and return the user's machine IP. This then calls the User Profiles Web service to search for that particular machine IP from a User profiles XML database. If the mapping is found, the user's desktop machine IP is then returned to the VNC mobile client application.

6. Start the VNC service on the target machine. Once it is found, the system will initiate a VNC connection to the target machine i.e., start the VNC service on the target machine by calling Remote VNC Web service. Note that, the .NET Remoting client (in the form of .exe file) needs to be run on the target machine before all the communications between a Web service and target machine can be performed. In addition, the Remote VNC Web service contains code that can perform a task to start or stop the process of the VNC service on the target machine.

In .NET Remoting technology, a more robust approach to remoting objects can be achieved by using interfaces to define the contract between the client and the server, where normally an interface and its implementation reside on the client side. Here is a sample of the remoting client that implements the method called remoteVNCStart of the IMediaPlayerForm interface. Once, there is a request from a Remote VNC Web service to execute this method, the VNC process will then be started on the target machine. The code snippet below is written in the VB.Net language.

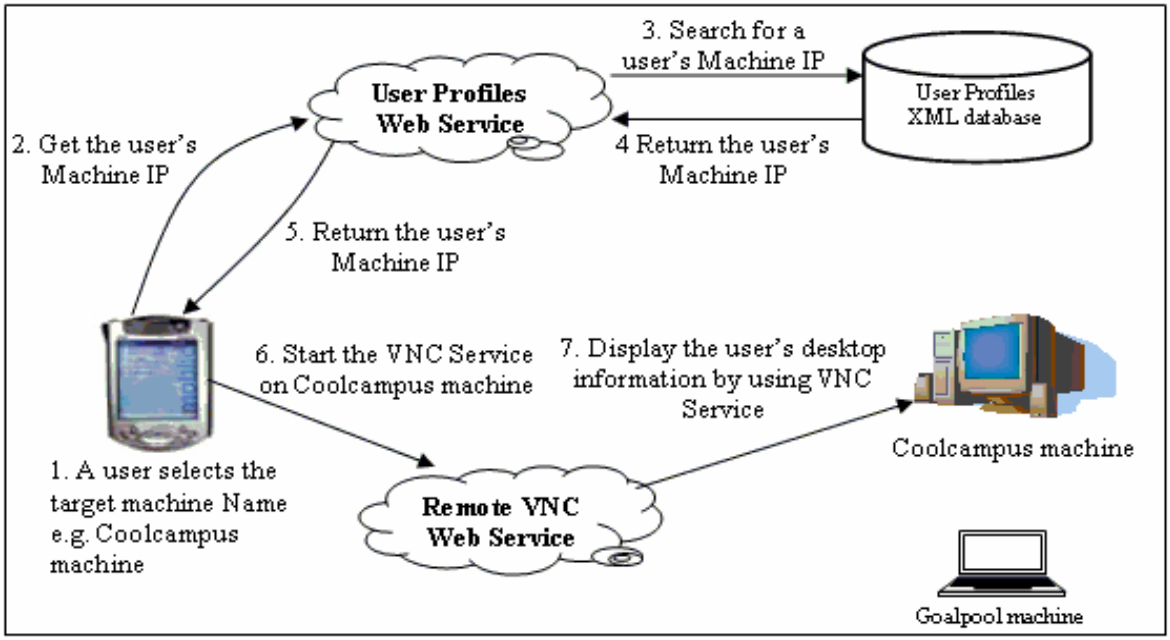
```
//a remoteStart method that takes an input of a target machine's IP
Public Sub remoteVNCStart(ByVal machineID As String) Implements
IMediaPlayerForm.remoteVNCStart
//initialize the System.Diagostics process in order to start the
//service i.e., VNC service on the target machine
   Dim aProcess As System.Diagnostics.Process
   aProcess = New System.Diagnostics.Process
//specify the location of the file
   aProcess.StartInfo.FileName = "C:\\Inetpub\\wwwroot\\vncviewer.exe"
//pass in the user's machine ID to a System.Diagnostics process
   aProcess.StartInfo.Arguments = machineID
//start the execution of the specified process i.e., a VNC service
   aProcess.Start()
End Sub
```

7. Display the user's desktop information on the target machine. Once there is a request to start a VNC application, our .NET remoting client that resides on the target machine then starts the VNC application process. This then displays a user's desktop information on the target machine.

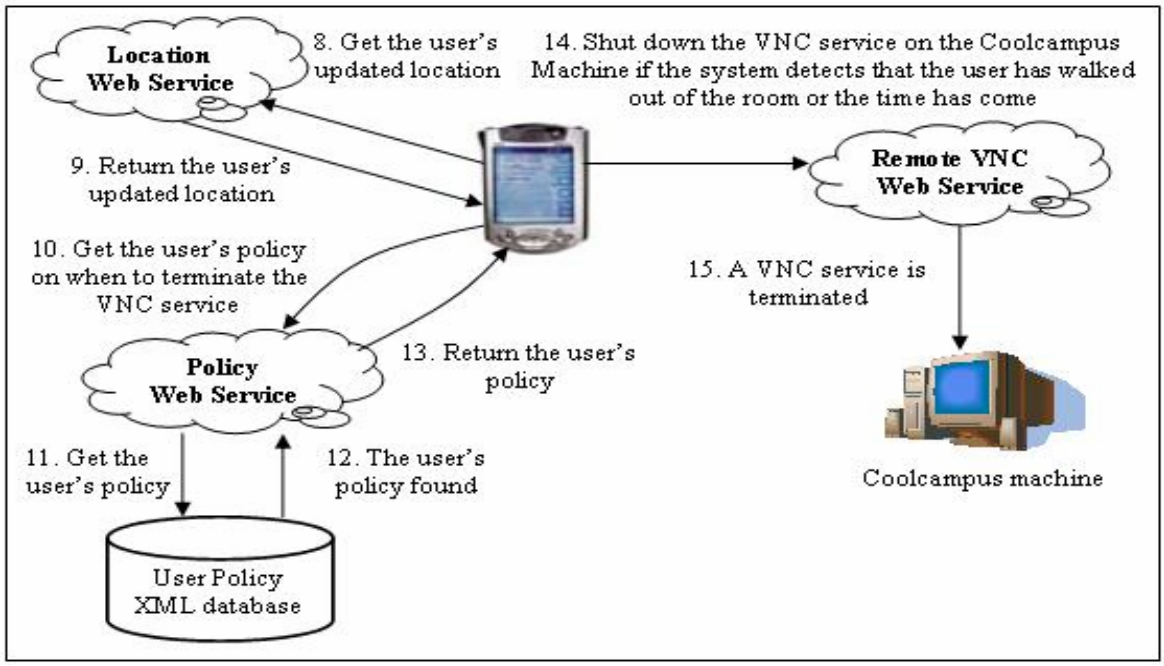
The steps mentioned above are for starting the VNC application process manually i.e., initiated by a user by selecting on the machine name from a mobile device. Our system is also able to start the VNC application automatically and without the user's intervention, by reading from a user policy file about when and where to start and stop the VNC process. For example, a policy could be: Activate Remote VNC application on every Wednesday between 2PM to 3PM, only if the system has detected that the user has entered B3.50 room. Here is an example of a user policy XML file:

```
<?xml version="1.0">
<VNCPolicy>
    <User name="dave">
      <PolicyDetails>
             <Activity day="Wednesday" startTime="2:00PM" endTime="3:00PM">
                    <Action>Activate Remote VNC</Action>
                    <Location>B3.50</Location>
             </Activity>
      </PolicyDetails>
   </User>
</VNCPolicy>
```

To process policies, the system uses multiple threads. The threading process will run once the mobile client application is started on the device. This running thread will keep monitoring the current time and location of the user. Once the time and location are detected, our system then automatically downloads and executes the specified service on the mobile device. If, for example, a system detects that there are two lists of machine names in the room, the system then searches through a user profiles XML database to find out on which target machine the user wants his/her desktop information to be displayed on. Once it is found, the system then calls up the Remote VNC Web service and passes on the user's desktop machine IP. After this, the steps are the same as steps 6 and 7 above.



(a) Start the VNC service on the target machine



(b) Terminate the VNC service on the target machine

Figure 2: Remote VNC Context Aware

After describing starting of VNC on the target machine, now, we discuss the steps to terminate VNC (see Figure 2b above). In general, there are two possible reasons of why the VNC service needs to be terminated on the target machine: (a) if the system detects that user is no longer in the room i.e., if the user has walked out , and (b) if the time to stop the VNC service in the user's policy file has been detected. In this case, a user specifies in his/her profile about when to start and terminate the VNC process. The second option would be viable if there is regularity in the user's activities. For example, at home, a user always checks his email at around 8-9PM and this is done by teleporting to his desktop machine at the office and the user usually finishes checking his email at around 10PM.

By specifying such behaviour in a policy file, the user is freed from performing regular tasks i.e., to activate and terminate the VNC service and can enable the system to do it automatically.

8. Get the user's updated location and return the updated location. Our system uses either of the above two ways to terminate the VNC service process on the target machine. Our system first checks whether the user is still in the same room. If system detects that the user has walked out of the room, the system then continues with step 14. Otherwise, steps 9,10,11,12 and 13 are performed.

9, 10, 11, 12 and 13. Get the user's policy and return the policy to the mobile client application. If the system detects that the user is still in the same room, the system then checks for a user's policy. If the time to shut down the VNC service has elapsed, the system then continues with step 14, otherwise, the system continues to monitor the user's current location (refer to step 8 above).

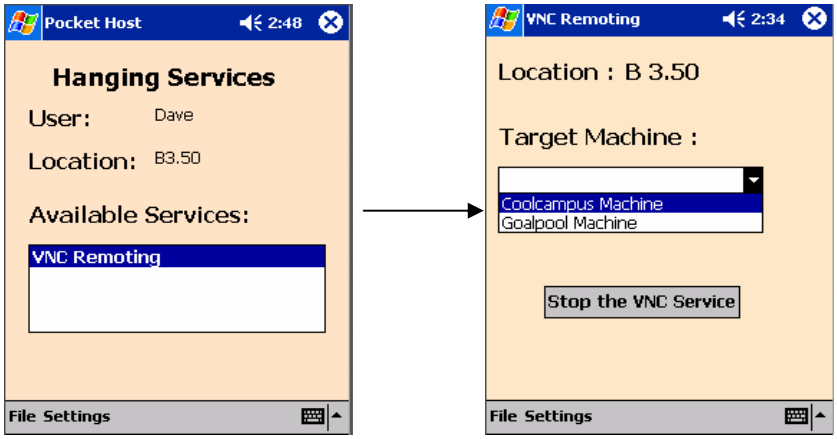
14. Shut down the VNC service on the target machine. If there is a request to shut down a VNC service, the remote client that resides on the target machine then needs to kill the running process of the VNC.

15. A VNC service is terminated. After the VNC process is killed, the VNC application that is displaying a user's desktop information is then terminated.

The steps 1-15 described above can be the same for adding context-awareness to other applications such as Media Player, Games, etc. The difference is that instead of the Remote VNC Web service starts the VNC process on the target machine, another type of application process will be started. The user may want to specify different execution and termination policies in the User policy XML database. Other points of control, apart from starting and stopping can be done if an application has an API to allow control from external processes.

3 Mobile User's Perspective of the System

The layout of the interface is critical for the effectiveness of the system. We have implemented a simple interface of our MHS system on handheld devices. The interface is divided into three main regions, the label that describes the user who is currently logged on, the current location context of the user and the type of services available for the particular context of user, location and time (see Figure 3a below – note only one service is listed but there can be many others). After selecting the service name i.e., VNC Remoting, a managed mobile code for this service is downloaded and executed on the device (see Figure 3b below). When the user selects on one of the target machine names i.e. Coolcampus machine, a user's desktop information is then displayed at Coolcampus machine (see Figure 4 below).



(a) Mobile Hanging Services (b) VNC Remoting Application

Figure 3: Mobile User Perspective Screen Shots

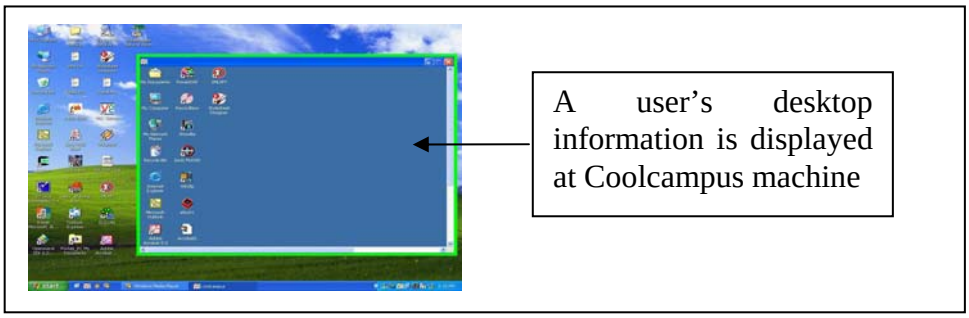


Figure 4: Displaying a user's information on Desktop machine - VNC started

4 Performance Evaluation

We have described a detailed evaluation of our MHS framework in measuring the time it takes to get a user's updated location by calling location Web service up to the time that it requires to execute the service on the mobile device [16]. We also have discussed in [18], different heuristic techniques that can be used to improve the service performance i.e., by reducing service execution time and context change delay when the user moves from one place to another. We now discuss in more detail the evaluation aspects in executing the VNC service on the target machine. The evaluation starts from the Web service call to the Remote VNC Web service, initiates a communication with the remote client, executes the VNC and up to the termination process of the VNC service. Figure 5 below shows each of the evaluation aspects of the VNC service.

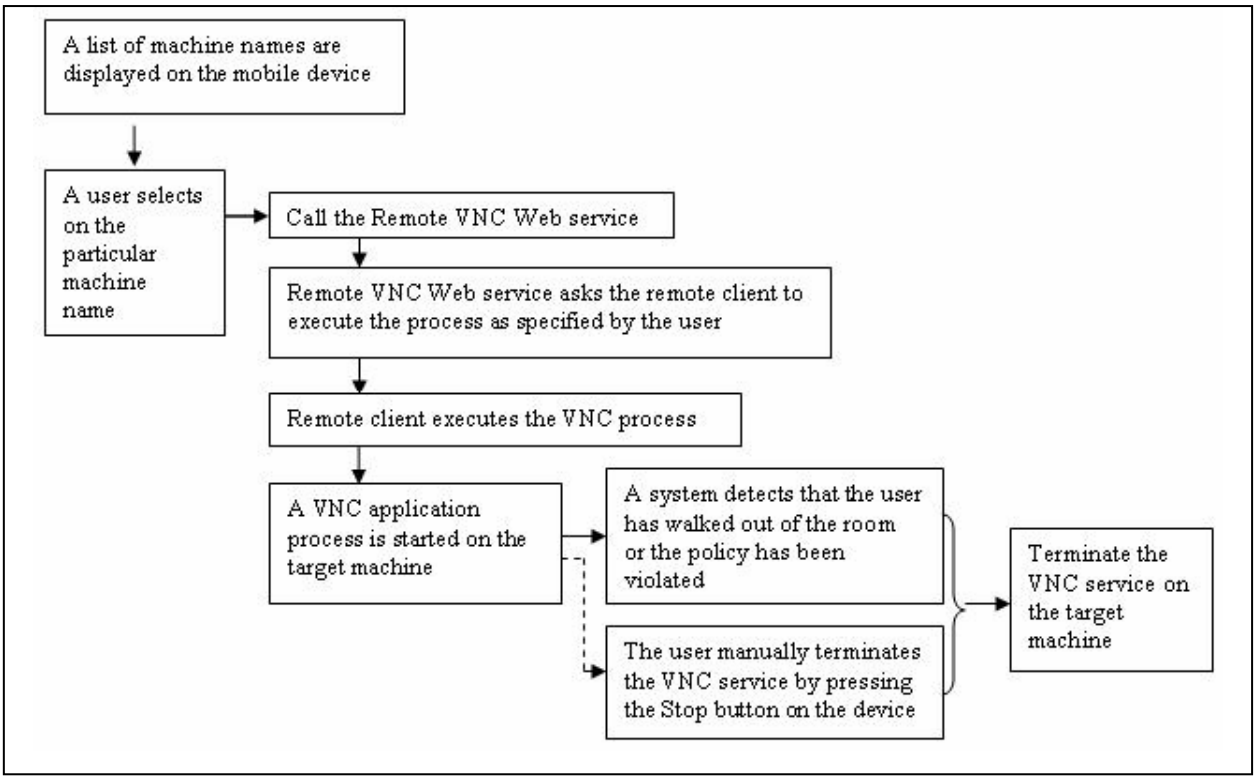


Figure 5: Stages of the VNC service

In this evaluation test, samples were collected over five times of VNC service activation. The Figure 6 below illustrates in more detail the timing variety that it takes to perform the five main aspects of the system on a 206 MHz iPaq on a wireless Wifi network.

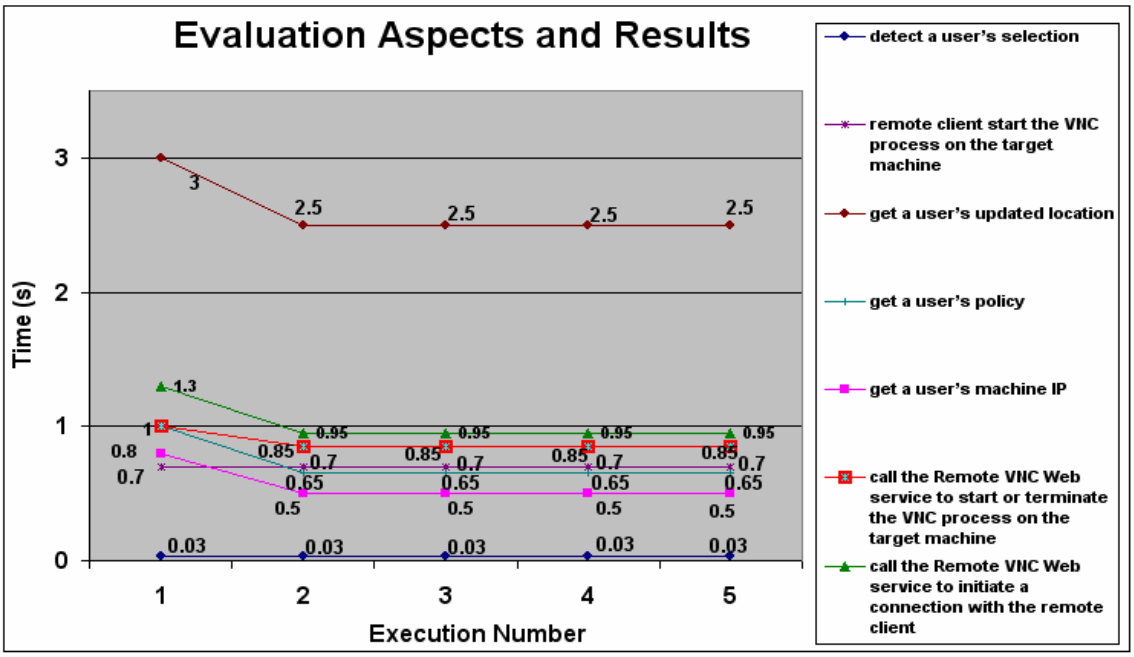


Figure 6: Experimental Results

Now, we give a formula to measure the time for VNC service activation on the target machine:

$$\mathbf{T_{VNC\ Service\ Activation(s)}} = T_{\text{detect a user's selection(s)}} + T_{\text{get a user's machine IP(s)}} + T_{\text{call the Remote VNC Web service to initiate a connection with the remote client(s)}} + T_{\text{call the Remote VNC Web service to start the VNC process on the target machine(s)}} + T_{\text{Remote client start the VNC process on the target machine(s)}}$$

Based on the formula above, the worst case scenario of VNC service activation is the first time of service execution which takes 3.83s (=0.03 + 0.8 + 1.3 + 1 + 0.7). The best case scenario for the VNC service activation is any execution number, which is not the first. It takes 3.03s (=0.03 + 0.5 + 0.95 + 0.85 + 0.7), as *the Web service proxy object has been downloaded to the client device and compiled in the first run* [16]. In addition, we also measure the total time that it is required to terminate the VNC service if the system detects that the user is still in the same location i.e., the user has not walked out of the room:

$$\mathbf{T_{VNC\ Service\ Termination(s)}} = T_{\text{get a user's updated location(s)}} + T_{\text{get a user's policy(s)}} + T_{\text{call the Remote VNC Web service to initiate a connection with the remote client(s)}} + T_{\text{call the Remote VNC Web service to terminate the VNC process on the target machine(s)}} + T_{\text{Remote client terminate the VNC process on the target machine(s)}}$$

From the testing and evaluation results above, we can conclude that the VNC service termination time is 7.5s (=3.5 + 1 + 1.3 + 1 + 0.7). The best case scenario i.e., to get the VNC service termination done in a minimum amount of time is 5.65s (=2.5 + 0.65 + 0.95 + 0.85 + 0.7). The testing and evaluation results are encouraging and suggest what sort of timing constraints our approach is feasible for. We believe that the time that it will require to activate and terminate other traditional applications on the target machine are similar with the VNC service.

5 Related Work

This section provides a brief overview about the research work that has been done to date that also concentrates on developing a framework for context aware application. While many authors have acknowledged the usefulness of context-awareness in the ubiquitous environment, only little work has been done to-date that supports a mobile framework in this field. Some earlier mobile context aware frameworks are the Hodes system [6] and Hive [7].

The Hodes system [6] introduces a concept of mobile context aware framework by employing the mobile code for downloading a service interface and application into a mobile device. This system aims at providing variable network services in different network environments, which involves changing connectivity. Hodes also introduced an open service architecture with minimal assumption about standard interfaces and protocols to support

heterogeneous client devices. However, the implemented prototype application has not incorporated the services for per-user location based interfaces. Our system has implemented different types of services for each user on the particular location. For example, the lecturer that is visiting the administration office may be interested in different services than the student in the same location.

Another mobile context aware framework in the field is Hive [7]. Hive is a distributed software agent platform that uses a combination of wearable and ubiquitous computing to address the concept of context-aware services. In this project, the computation and information are shared between the environment and the wearable. Rhodes claims that implementing the location-aware systems in both pure ubiquitous and wearable have presented fundamental difficulties [7]. This is due to the ubiquitous computing tend to have troubles with personalisation and privacy, whereas, the wearable system has some drawbacks with localised information, resource control and management.

Hive is a Java-based agent architecture, that relies on the Remote Method Invocation (RMI) distributed objects and mobile code. In contrast, our system is implemented on the highly compact mobile environment (.NET Compact Framework) that employs the concept of Web service for the purpose of retrieving the updated user's location and a list of available services in a particular location. Moreover, our framework also enhances some policies used for service execution.

There are also some various other context-aware applications surveyed in [2]. However, none of them are using mobile code and positioning technology like the way we do. Among previous work on exploiting a framework for context aware applications, much work has been done on location aware systems. Some existing projects, which have successfully developed an application that is aware of the user's location are Mobile Shadow. The Mobile Shadow [13] project focuses on prototyping applications for proactive cell-based location aware services with mobile code. Another project that also makes use of location sensing is Virtual Tour Guide [19]. A Virtual Tour Guide project uses a GPS system to detect a user's location in an outdoor environment. As soon as the user walks past or enters an area, which is delimited by the GPS coordinates, a relevant stick e-note to a physical location is delivered to the user's mobile device.

A Cool town project from Hewlett Packard also developed a project that makes use of the knowledge of the user's physical location [20]. The interesting part from this project is the creation of a mobile WWW infrastructure with respect to the physical location. As the mobile user moves toward the physical space, the relevant WWW pages are displayed on the user's mobile device. In addition, some other projects in location context-aware field are described in [15, 17].

Another interesting aspect that we will be looking at is some earlier location aware teleporting systems, which have been explored by researchers from Olivetti Research Laboratory [4, 9] and Bat Teleporting [3]. The teleporting system, which was developed by Olivetti Research Laboratory (ORL), introduced a notion of mobile application that is not fixed to a particular display, instead it provides a flexible X displays as the user moves from place to place. In here, X refers to an undefined target display as the target display can be initiated on demand by the user. This system was integrated with the Active Badge tool in order to determine the current location of the user and any other equipment within a room or building [1].

Apart that the badge can be used to detect the location, it can also act as an authentication process, as it basically has a unique identity and is normally worn only by its owner. In addition, the badge can also be used as a control mechanism for teleporting system. In this case, by pressing on one of the buttons on the Active Badge, this will then initiate a request to bring up a user's desktop information on the selected X server within the room. As the system detects that the user has leaved the room, the system then shuts down the teleporting session on the teleported machine.

The other teleporting project is the Bat Teleporting system. This system was built upon the existing infrastructure of the above ORL teleporting project with aims to address some of the issues found. Some of these are to maximize the user's efficiency in performing a teleporting on the target machine as with ORL teleporting project, a user needs to cycle through all the candidate machines by pressing the button on the Active Badge, in order to teleport to the destination machine. This is inefficient if there are a large number of machines in a room. Another issue is there is no resource information available on the target machine if it is currently not running or there is no X display software installed.

The Bat Teleporting system makes use of the Virtual Networking Computing (VNC) system in order to provide a teleporting mechanism. This project also introduced a concept of location sensing with Bats sensoring system that consists of a radio transceiver, logic controller and ultrasonic transducer. In addition, this project also uses a CORBA interface in order to initiate or terminate existing connection on the target machine. Hence, a user can directly initiate a connection on the target machine without the need to cycle through all the candidate machines in the room. The Bat Teleporting project has delivered an efficient framework for the teleporting system. However, it only focuses on the location context. In contrast to our conception of MHS system, we employ different variety of contextual information, which are used in conjunction with the VNC teleporting system i.e., a user, location and time contexts.

We also developed a prototype implementation that gives a mobile user the ability to control the execution of the VNC application from the mobile device i.e., users can manually initiate a VNC process on any target machine that they prefer or even terminate the VNC process by pressing a Stop button on the device. This is all possible as our system employs Web services that allow method invocation in disparate platforms and languages. In addition, as our VNC teleporting system was built on top of the MHS framework, we can easily enrich the contexts that we want to use. Besides that, we can also add a policy to enhance the automatic behaviour of the teleporting application. This is a benefit in our approach - developing context aware applications based on traditional applications.

6 Conclusion and Future Work

We have presented an architecture for "Mobile Hanging Services", allowing a mobile device to adapt its functionality to exploit a set of services that it discovers depending on the user, location and time contexts. We also proposed that adding context awareness to the traditionally designed application is really useful, especially if we want to give a mobile user an experience of proactive behaviour depending on his/her current situations. We also have developed a prototype implementation of adding context awareness to the VNC traditional application. This is all possible with the use of the existing MHS framework plus a .NET Remoting technology. Some evaluations on how long it takes to start and terminate VNC process on the target machine have also been given. From the testing and evaluation results, we can conclude that our system is fairly efficient. Our future work will consider more complex policies and add more types of contexts to our current prototype implementation, including a history log file and physical sensors.

7 Acknowledgement

We wish to thank Dominic Cooney for his contribution to the earlier stages of mobile framework that supports mobile code in ad hoc environment.

8 References

1. Want, R., Hopper, A., Falcão, V. & Gibbons, J. (1992), "The Active Badge Location System", *ACM Transactions on Information Systems*, vol.10, no.1, pp.91-102.

2. Chen, G. & Kotz, D. (2000), "A Survey of Context-Aware Mobile Computing Research", Dartmouth Computer Science, *Technical Report TR2000-381* [online], Available: ftp://ftp.cs.dartmouth.edu/TR/TR2000-381.pdf [Accessed 03 April 2003].

3. Harter, A., Hopper, A., Steggles, P., Ward, A. & Webster, P. (1999), "The Anatomy of a Context-Aware Application", *Proceedings of the 5th International Conference on Mobile Computing and Networking*, Seattle, USA, 15-19 August 1999, pp.59-68.

4. Richardson, T. (1995), "Teleporting Mobile X Sessions", Olivetti Research Laboratory, Cambridge, United Kingdom, *Technical Report TR95.7* [online], Available: http://www.uk.research.att.com/pub/docs/att/tr.95.7.pdf [Accessed 03 April 2003].

5. Richardson, T., Fraser, Q.S., Wood, K.R. & Hopper, A. (1998), "Virtual Network Computing", *IEEE Internet Computing*, vol.2, no.1, pp.33-38.

6. Hodes, T.D. & Katz, H.R. (1999), "Composable ad hoc location based services for heterogeneous mobile clients", *Wireless Networks*, vol.5, no.5, pp.411-427.

7. Rhodes, J.B., Minar, N. & Weaver, J. (1999), "Wearable Computing Meets Ubiquitous Computing: Reaping the best of both worlds", *Proceedings of the 3rd International Symposium on Wearable Computers*, San Francisco, USA, 18-19 October 1999, pp.141-149.

8. Ekahau Positioning Engine™ 2.0 Developer Guide [Available on commercial license].

9. Bennett, F., Richardson, T. & Harter, A. (1994), "Teleporting – Making Applications Mobile", *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, USA, 8-9 December1994, pp.82-84.

10. Prekop, P. & Burnett, M. (2003), "Activities, Context and Ubiquitous Computing", *Special Issue on Ubiquitous Computing Computer Communications*, vol.26, no.11, pp.1168-1176.

11. Schmidt, A., Beigl, M. & Gellersen, H-W. (1999), "There is More to Context than Location", *Computers and Graphics*, vol.23, no.6, pp.893-901.

12. Turner, R.M. (1998), "Context-Mediated Behaviour for Intelligent Agents", *International Journal of Human-Computer Studies*, vol.48, no.3, pp.307-330.

13. Fischmeister, S., Menkhaus, G. & Pree, W. (2002), "Context-awareness and Adaptivity Through Mobile Shadows", Software Research Lab, University of Salzburg, Austria. *Technical Report TR-C047* [online], Available: http://www.softwareresearch.net/reports/C47.pdf [Accessed 12 April 2003].

14. Castro, P. & Muntz, R. (2000), "Managing Context Data for Smart Spaces", *IEEE Personal Communications*, vol.7, no.5, pp.44-46.

15. Beadle, H.W.P., Harper, B., Maguire, G.Q. & Judge, J. (1997), "Location Aware Mobile Computing", *Proceedings of the IEEE International Conference on Telecommunications*, Melbourne, Australia, April 1997, pp.1319-1324.

16. Syukur, E., Cooney, D., Loke, S.W. & Stanski, P. (2004), "Hanging Services: An Investigation of Context-Sensitivity and Mobile Code for Localised Services", *Proceedings of the IEEE International Conference on Mobile Data Management*, Berkeley, USA, 19-22 Jan 2004, IEEE Computer Society Press, Los Alamitos, CA, USA, pp.62-73.

17. Davies, N., Cheverst, K., Mitchell, K. & Friday, A. (1999), "Caches in the Air: Disseminating Tourist Information in the Guide System", *Proceedings of the 2nd IEEE Workshop on Mobile Computer Systems and Applications (WMCSA)*, New Orleans, USA, 1999, pp.11-19.

18. Syukur, E., Loke, S.W. & Stanski, P. (2004), "Performance Issues in an Infrastructure for Mobile Hanging Services", *Proceedings of the First International Conference on Mobile Computing and Ubiquitous Networking (ICMU)*, NTT DoCoMo R&D Center, Yokosuka, Japan, 8-9 Jan 2004, pp.32-37.

19. Brown, P.J., Bovey, J.D. & Chen, X. (1997), "Context-Aware Applications: From the laboratory to the marketplace", *IEEE Personal Communications*, vol.4, no.5, pp.58-64.

20. Kindberg, T., Barton, J., Morgan, J., Becker, G., Caswell, D., Debaty, P., Gopall, G., Frid, M., Krishnan, V., Morris, H., Schettino, J., Serra, B. & Spasojevic, M. (2000), "People, places, things: Web presence for the real world", *Proceedings of the 3rd IEEE Workshop on Mobile Computing Systems and Applications*, Monterey, CA, USA, December 2000, pp.19-28.