The Mobile Hanging Services Framework for Context Aware Applications: the Case of Context-Aware VNC

Evi Syukur, Seng Wai Loke and Peter Stanski

SCSSE, Monash University, Australia
900 Dandenong Road, Caulfield East, Victoria 3145

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. In this paper, we discuss some of the issues, usefulness, and a means of adding context awareness to an existing application, especially a VNC traditional application. In order to exploit the benefits of context sensitivity in the mobile environment, we propose the Mobile Hanging Services framework.

1 Introduction

Context sensing has the ability to usefully adapt the services or applications to the user's current situation, intention or environment. This ability to sense that can be found in the context aware applications has distinguished them from the existing traditional applications. Here, the traditional application refers to a primitive standalone 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. 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 profiles where and when a VNC application should be started or terminated – for example, while the user is not in his/her office but in a different room, sensing the user's location, automatically start the VNC application on any computer that he/she is near (in the room he or she is currently in), and terminate the VNC application as soon as the user walks out of the room.

Besides adding context-awareness to traditionally designed 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 the mobile environment. A 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 other 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 provides 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 of being generic, it also explores 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 [6]. Besides that, MHS also allows the user to specify the policy that can govern the service execution.

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 briefly discuss the evaluation of the system. In section 4, we discuss related research and conclude in section 5.

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 are 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.

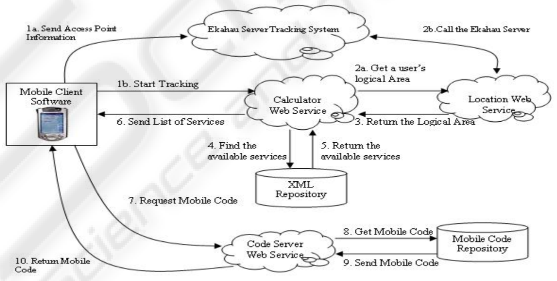

Fig. 1. System Architecture of MHS system

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 [6].

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 the signal strength measurements [4].

c. Calculator Web service. In our work, the Calculator Web service computes a proactive service using a combination of a user, location and time contexts. At the specific time and location, users get different types of services. **d. Code server Web service**. Within our system, we employ the Web service as a method invocation to retrieve and return a mobile code application that matches with the service name 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 application containing service interface and data associated with the service 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 context 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 [3].

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 without having to modify the existing code. 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 VNC application is executed on the target machine.

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.

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

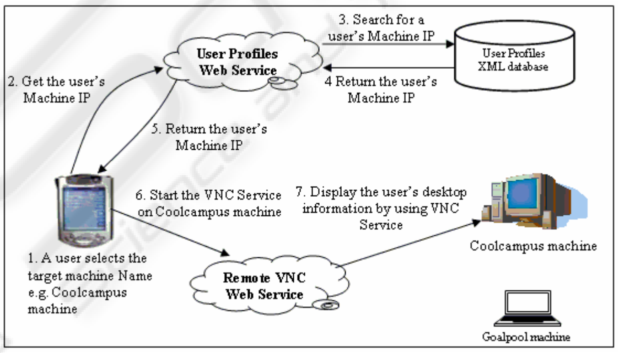
3. Search for a user's Machine IP
User Profiles Web Service
User Profiles XML database
2. Get the user's Machine IP
4 Return the user's Machine IP
5. Return the user's Machine IP
6. Start the VNC Service on Coolcampus machine
7. Display the user's desktop information by using VNC Service
Coolcampus machine
1. A user selects the target machine Name e.g. Coolcampus machine
Remote VNC Web Service
Goalpool machine

Fig. 2. Start the VNC service on the target machine

1. A user selects the target machine name. As soon as the VNC service 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 service 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 the communications between a Web service and target machine can be performed. At our implementation, the execution of .exe file is done automatically by the system. 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. **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 start 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 in response to context changes. 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="Monday" 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 VNC 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. We also have a detailed discussion on the diagram and steps required to terminate VNC application in our technical report [8]. Some of the screen shots of our VNC application are also illustrated in [8].

3 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 [6]. We also have discussed in [7], 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. This section discusses in more detail about 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, up to the execution process of the VNC service.

Figure 3 below describes each of the evaluation aspects to execute the VNC service. In this evaluation test, samples were collected over five times of VNC service activation. The detailed discussion on 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 is described in [8]. Now, we give a formula to measure the VNC service activation on the target machine:

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

Based on the formula above, the worst case scenario of VNC service activation is the first time of service execution which takes 7.53s (=0.03 + 2 + 2.5 + 2 + 1). The best case scenario for the VNC service activation is any execution number, which is not the first. It takes 6.93s (=0.03 + 1.8 + 2.3 + 1.8 + 1), as *the Web service proxy object has been downloaded to the client device and compiled in the first run [6, 8]*.

In addition, we also measure the total time that it is required to terminate the VNC service:

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

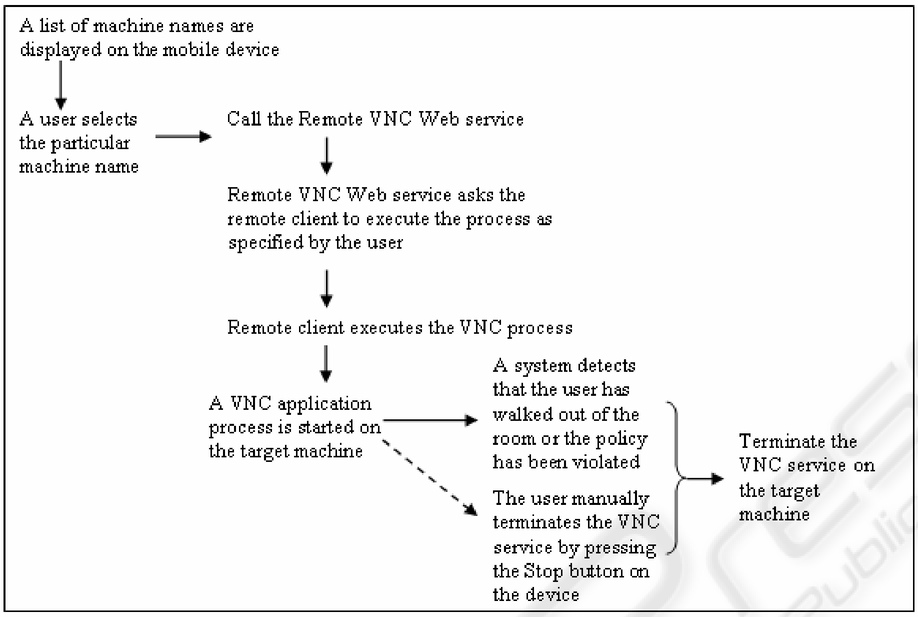
A list of machine names are displayed on the mobile device
A user selects the particular machine name
Call the Remote VNC Web service
Remote VNC Web service asks the remote client to execute the process as specified by the user
Remote client executes the VNC process
A VNC application process is started on the target machine
A system detects that the user has walked out of the room or the policy has been violated
The user manually terminates the VNC service by pressing the Stop button on the device
Terminate the VNC service on the target machine

Fig. 3. Stages of the VNC service

From the testing and evaluation results above, we can conclude that the VNC service termination time is 11.5s (=3.5 + 2.5 + 2.5 + 2 + 1). The best case scenario i.e., to get the VNC service termination done in a minimum amount of time is 9.6s (=2.5 + 2 + 2.3 + 1.8 + 1). 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.

4 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 location aware teleporting system. Some of the earlier projects which are closely related to our work are teleporting system from Olivetti Research Laboratory [2, 5] and Bat Teleporting [1]. Most of the existing teleporting projects only focus 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 service 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.

5 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 identity, 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. In addition, 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.

6 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

7 References

1. Harter, A., Hopper, A., Steggles, P., Ward, A. and Webster, P., "The Anatomy of a Context-Aware Application", Wireless Networks 1, pp. 1-16, 2001.
2. Richardson, T., "Teleporting Mobile X Sessions", Olivetti Research Laboratory, Cambridge, United Kingdom.
3. Richardson, T., Fraser, Q.S., Wood, K.R. and Hopper, A., "Virtual Network Computing", IEEE Internet Computing, Vol.2, number 1, January/February 1998.
4. Ekahau Positioning Engine ^{TM} 2.0 Developer Guide.
5. Bennett, F., Richardson, T. and Harter, A., "Teleporting – Making Applications Mobile", Olivetti Research Laboratory, Cambridge, United Kingdom.
6. Syukur, E., Cooney, D., Loke, S.W. and Stanski, P., "Hanging Services: An Investigation of Context-Sensitivity and Mobile Code for Localised Services", IEEE International Conference on Mobile Data Management, Berkeley, CA, USA, Jan 2004.
7. Syukur, E., Loke, S.W. and Stanski, P., "Performance Issues in an Infrastructure for Mobile Hanging Services", First ICMU Conference, NTT DoCoMo R&D Center, Japan, Jan 2004.
8. Syukur, E., Loke, S.W. and Stanski, P., "The Mobile Hanging Services Framework for Context Aware Applications: An Experience Report on Context Aware VNC". Technical Report, Monash University, Australia.