A Framework for a Session Mobility System

Dan MacCormac, Mark Deegan, Fred Mtenzi, Brendan O'Shea
School of Computing, Dublin Institute of Technology, Ireland.
{dan.maccormac, mark.deegan, fred.mtenzi, brendan.oshea}@comp.dit.ie

ABSTRACT

As the average user interacts with more and more computing devices on a daily basis, the need for interaction continuity when moving from device to another becomes stronger. Users are seldom provided with the ability to move a session from one device to another. Instead, they are often left to manually restore applications to their previous state on each device encountered. This can be difficult, time consuming and sometimes impossible. This approach can also lead to varying copies of data and unsynchronised information. We present an adaptive approach to session mobility which alleviates the need to manually restore sessions when moving between devices. Moreover, our approach does not tie a user to single operating system, but rather provides a rich heterogeneous environment consisting of a range of applications specific to various platforms. Dynamic movement of user sessions across a broad range of devices from the desktop computer to the mobile phone is supported in a flexible and adaptive manner.

Key Words: Session mobility, pervasive computing, systems integration, thin client computing

1. Introduction

In both corporate and academic environments, many users tend to move between multiple computers running a range of operating systems and software in the course their day. Even at the desktop computer level, the ability to suspend, resume or move a session is seldom provided. Terminating the session by closing all applications, moving to a new computer, and reopening and restoring applications to their previous state is the most common solution in practise. This can be time consuming and becomes increasingly frustrating over the time. Without the ability to move a session from one device to another, the problem of unsynchronised information and communication barriers become apparent. To address these problems, people sometimes carry a notebook computer or PDA, carrying their session and associated data with them at all times. This solution is partial at best and presents further barriers such as limitations imposed by a particular device or operating system as well as the physical inconvenience of carrying the device at all times.

A system which allows sessions to be suspended, resumed or moved to new terminals allows users working in public domains to become more productive. Our proposed approach does not tie a user to a particular operating system, as many existing solutions do, but instead we aim to provide a heterogeneous environment comprised of applications specific to a wide range of platforms and architectures. This heterogeneous mobile session is capable of being dynamically moved across a broad range of client devices running various systems, from the traditional desktop computer to the smart-phone.

The concept of pervasive computing in addition to the emergence of new wireless and sensor technologies presents new and exciting possibilities and allows the exploration of new concepts in

terms of user tracking, management and location based facilities. Provision of a heterogeneous sensor network would provide an infrastructure in which to deploy the proposed system. The session mobility application could be notified of the movement of users from one room to another, and this information could then be used a basis for movement of sessions from one device to another. Ideally, this should be completely transparent to the end user.

From the problems outlined above we have identified several key objectives which we aim to address.

- Enable the mobility of legacy applications
- Avoid modification to existing Operating Systems
- Support heterogeneous client platforms
- Support seamless integration of mobility enabled applications where possible
- Enable sharing of workspaces with multiple users for presentations and collaborative work
- Support efficient management of network resources

The remainder of the paper is structured as follows. In section 2 we discuss related work in the area. In section 3 we outline our approach to achieving our objectives, and in section 4 we present the design of our system. Having discussed the chosen technologies for the system, in section 5 we present the system architecture, discussing the role of each component of the system. In section 6 we analyse the effectiveness of our approach and re-evaluate our chosen approach to building the system. Finally, in section 7, we present are concluding remarks and discuss potential future work.

2. Related Work

The issue of web interface mobility has been discussed in [1]. The work outlined supports applications built using a multi-modal approach, and is capable of choosing the most appropriate mode for the current device. However, to take full advantage of the capabilities of the system, applications must be built using a specialised toolkit. Similarly, in [2], the authors present a ``multibrowsing'' system which allows the movement of web interfaces across multiple displays. This work supports the movement of existing web applications, broadening its usage scope. Kim, Baratto and Nieh [3] present pTHINC, a thin client approach to supporting wireless web browsing on various handheld devices. The work takes into account the limitations of today's mobile web browsers in relation to multimedia content which is of growing importance to user interaction. As the interaction with mobile devices becomes more complex in response to their growing capabilities, the need to support mobility of a wider range of applications becomes apparent. ROAM, a system to support the movement of Java based applications between heterogeneous devices is presented in [4]. This work also requires developers to build applications using a specific approach thereby limiting the applicability of this work. Guyot et al. [5] investigate smart card performance as a token for session mobility between Windows and Linux. This work supports mobility of a wide range of applications, and is also capable of remotely fetching and installing necessary applications which were available on the previous terminal but not on the present terminal. The approach taken in this work involves the restoration of a session based on a session state file. Our approach involves the dynamic movement of an application from one device to another. Furthermore, this work does not address the use of mobile devices, which is central to our work.

The work outlined in this paper focuses on enabling mobility of legacy applications across heterogeneous devices. We aim to provide *seamless* integration of the interface of the mobility enabled application into the users current environment, allowing both mobile and stationary applications to work side by side. Moreover, our approach takes into

consideration the diverse range and capabilities of various target platforms, providing adaptive methods of session mobility. This reduces footprint when moving sessions to mobile devices with constrained capabilities. We use a thin client approach to providing session mobility. Existing thin client solutions which display an entire desktop environment confine users to particular Operating Systems. This prevents users from merging the power of applications which are specific to a variety of platforms.

3. Approach

The use of thin client computing is a suitable approach to providing session mobility. Thin client computing involves running user applications on a server machine which direct their output to light-weight client devices which merely act as display terminals. All application logic is executed on the server side and consequently the duties and requirements of client devices are minimal. There are two varieties of thin client computing; hardware based and software based. The software based approach involves using a small client application on a traditional desktop computer to connect to a remote session on a thin client server. The hardware based approach involves much simpler client hardware which is specifically designed for the purpose of connecting to and displaying sessions running on remote servers. Such devices are not capable of acting as independent workstations since they lack the hardware of their counterparts, running firmware as opposed to an entire operating system. Aside from client requirements, the thin client approach is advantageous in a number of respects. It simplifies the duties of system administrators, centralising management and simplifying deployment of applications. Thin client software solutions allow movement of sessions to existing common devices and are therefore the focus of this work.

To identify suitable technologies for this system, we assessed and considered a number of thin client software products. Firstly we considered the underlying protocol of the technology, and secondly we assessed the performance of each product in a laboratory environment. The testing laboratory consisted of a range of client devices running operating systems such as Microsoft Windows, Mac OS X, Linux, Sun Solaris, Windows Mobile and Symbian OS. Products tested included Microsoft's Terminal Services [6], Virtual Network Computing (VNC) [7], the X Window System [8], GraphOn GoGlobal [9], Citrix [10], and Tarantella products [11].

Using a single thin client protocol can limit the capabilities of the system. Merging two suitable protocols together provides a richer more flexible solution to the problem of session mobility. From our technology assessment, we concluded that the X Window System and VNC were most suitable for the requirements of this system.

4. Design

Having identified key objectives and a suitable approach to achieving these objectives, we now discuss the role of each technology in relation to the design of the system.

4.1 The X Window System

The X Window System is a windowing environment that forms the basis of the majority of UNIX-based GUI systems today. Inherently networked, all X applications are already capable of remote display. X is commonly associated with Linux, but implementations of X are also present on many systems; for example, BSD, Sun Solaris and Apple Mac OS X. Microsoft Windows does not come with support for X, but many third-party implementations exist, both free and proprietary.

X is suitable for development of this system for a number of reasons. Primarily, X is freely available under GPL. This could be crucial for development of add-on packages to X for the purpose of session mobility. In addition to remotely displaying the interface of applications in a seamless manner, X is also sparing on bandwidth as well as being widely supported. There are however some significant obstacles to consider when using X. Current implementations of the X Window system provide no means of moving the output of an application from one X server to another [12]. Furthermore, in the case of bandwidth under 1 Mbps, X does perform as well as some competitors, which can lead to slow sessions over low bandwidth connections. With the help of additional compression modules [13], X can be made to work more efficiently on low bandwidth connections. The fact that X does not provide a truly thin client environment is another drawback. The remote host is responsible for the current instance of the applications interface, therefore if the remote host crashes or the network connection breaks, the state of the application will be lost. X is supported by many thin-client hardware manufacturers, for example as SunRay Terminals [14], which is useful for environments in which such devices are present.

4.2 X11 Client Mobility

While the X window system provides the capability to remotely display applications, there is no support provided by the X protocol or the X11R6 implementation for the dynamic movement of clients between servers. The low-level C API for programmers to write X11 applications - *xlib* - provides no functions for moving the client, thus X client mobility must be provided in the form of an extension to existing X11 software. There are several possible approaches to providing X11 client mobility. Solomita [15] outlines and discusses these approaches to achieving X11 client movement and the use of a *pseudo-server* has been suggested as a suitable approach. A pseudo-server is an intermediary positioned between client and server. This allows for the interception, interpretation, change, and redirection of the X protocol messages exchanged between client and server and hence this information can be used as a basis for window movement.

The pseudoserver acts in a similar manner to a standard X server by listening for requests from new clients. When a pseudo-server receives a connection from a client, it opens a new connection to the real X server. It then serves as a pipeline, forwarding and possibly translating messages between client and server. Any application started on this pseudo X server will then be capable of having its output redirected to any other X server - real or pseudo. The main advantage of this approach is that it enables X client mobility without any modification to the client applications; therefore it can be used with the majority of software in existence. The primary disadvantage of this approach is that many aspects of client operation must be deduced at the protocol level. Another disadvantage of this approach is that it adds overhead since all messages must pass through the pseudo-server.

4.3 Supporting Mobile Devices

In order to support non X enabled devices such as smart phones, VNC can be used. VNC (Virtual Network Computing) enables server machines to supply not only applications and data but also an entire desktop environment that can be accessed from any client machine using simple client software. Regardless of the location or capabilities of the client machine, the state and configuration of the VNC desktop are exactly the same as when it was last accessed. VNC clients are truly thin client; they are stateless clients. When the client disconnects, the session continues to run on the server, and can be resumed from any other suitable client. VNC client software is available for a wide variety of client platforms, and the underlying protocol RFB (Remote

Framebuffer Protocol) is much lighter on client resources than X. VNC client software is available for PDAs and smartphones, in addition to all major operating systems. X and VNC provide varying levels of remote access. This allows the adaptation of session delivery, based upon the capabilities of the current client device.

4.4 Session Multiplexing

In addition to providing session mobility, the ability to broadcast a session to multiple terminals simultaneously could also be beneficial, especially in academic environments. Such a facility would allow lecturers to share their session in a *view-only* mode with the students within a laboratory, as oppose to using projectors. Groups of students or staff could also work collaboratively on tasks, increasing productivity. Several software packages which are capable of enabling this are available. Tools which can provide this feature are sometimes referred to as a *mux* utilities since they can multiplex output onto several remote terminals. Mux utilities include XMX [16], HP SharedX [17], RealVNC [18] and xtv [19]. In the case of multiplexing in view-only mode, *xtv* is a suitable choice. xtv allows remote users to view the contents of an X session within an xtv client window. The multiplex client cannot provide any input to the X session, instead a view only session is provided. To enable collaborative shared sessions, multiple users can connect to the a single VNC desktop from a several clients simultaneously. This approach allows us to leverage the already existing VNC component of the system.

4.5 Security Considerations

X terminals are capable of authenticating a connection from an X11 client using two methods - host based or token based authentication. Host based authentication relies on the IP address of the X server, while token based authentication relies on the provision of an authentication token, commonly referred to as MIT-MAGIC-COOKIE-1 scheme. Unwanted content being displayed within a session can be prevented by employing either of the above methods. Moreover, user sessions often contain potentially private data, which creates a strong need for protocol encryption. Neither X nor VNC offer encryption capabilities by default. Instead X11 and VNC protocol messages must be encapsulated in an SSH tunnel, enabling secure communication.

4.6 System I/O

Communication between this system and adjacent pervasive computing applications, for example a Location Based Services application, is facilitated by use of a custom built XML schema. This allows client devices to send requests in the form of XML strings to the client, which are then parsed and processed accordingly by the server. XML parsing is achieved using custom built XML parsing applications driven by *expat*, an open source XML parsing library. Responses issued by the system are in XML format. Furthermore, system databases are structured using XML, and the query functions are driven by expat based tools.

5. System Architecture

Having discussed the fundamental technologies which can be used as building blocks for the system, we now present a high level conceptual overview of each component of the system.

5.1 Client Side

In the case of the initial prototype of the system, clients can interface with the server using light weight TCP client applications. We have created client applications for Linux, Windows, Mac OS, BSD and Solaris. Using these simple client applications, users can *push* their sessions to any remote workstation or *pull* their session to their current workstation. A

higher level of transparency could be provided by delegating the task of session hand-off to a Session Management Server. This server could handle session movement from one device to another based on the physical movement patterns of a user, supported by appropriate user tracking applications.

5.2 Server Side

On the Session Mobility Server, a dedicated TCP server will listen for incoming requests on a specified port. These requested are then analysed and processed accordingly. This is achieved by invoking necessary modules within the server framework. Using our custom XML schema, an appropriate response is then returned to the client detailing the outcome of the request. The role of each component of the server framework is now discussed.

- *Request Listener*
  The Request Listener waits for incoming requests from clients. The Request Listener will first perform fundamental error checking to ensure that the incoming data meets minimal requirements. Having passed fundamental error checking, the request is then passed to the Request Manager module, where further error checking and analysis occur.

- *Request Manager*
  The Request Manager is responsible for processing requests and carrying out appropriate actions in addition to notifying the Response Manager of the outcome of the request. Conceptually, the Request Manager has five functions. These five conceptual functions are :

  1. StartSession()
  2. StopSession()
  3. MoveSession()
  4. MultiplexSession()
  5. MultiplexStop()

  Starting or stopping sessions can be achieved via the startSession() and stopSession() functions. Starting a session for a user involves starting a pseudo X server capable of window movement, as well as a VNC desktop. Movement of a session from one device to another is accomplished by invoking the moveSession() function. Activating or deactivating the Multiplex capability of the system is achieved by invoking the Multiplex() and MultiplexStop() functions respectively. Stopping a session by calling the stopSession() function will terminate the VNC desktop, pseudoserver, any associated applications and any multiplex clients.

- *Database Manager*
  The growth of wireless networking has lead to a change from the traditional static network model to a dynamically changing model. Many environments now provide wireless networking facilities and as a result new devices will continually be added to and removed from the network. The database manager must be able to manage this continuosly changing network model by dynamically updating necessary databases to reflect the current state of the network.

  Entries are created for newly discovered destination devices and trimming the database of obsolete devices can be accomplished by provision of a TTL (Time to Live) field for each device. While static devices have an infinite TTL value, mobile devices such as laptops and PDAs are assigned a specified TTL. Expiration of a TTL field triggers the removal of the specified device from the database.

- *Device Database*
  Information about devices, for example physical location, vacancy status and operating system details, are stored in a low level device database. The database is structured using XML format. The database can be queried by the Database Manager in a case where the request contains insufficient information for the Request Manager to fulfil the request. As an example, if a request to move a session from one workstation to another contains only a zone name as opposed to an IP address, the database must be queried in order to obtain the information necessary to complete the request.

- *Mobility Module*
  By invoking this function, an application currently directing its output to one terminal can dynamically redirect its output to an alternative terminal without interrupting the state of the session. As specified by the parameters of the request, a single application, multiple applications, or the entire session can be moved. Applications can be grouped together to form *sub-sessions* which can be displayed on multiple terminals, effectively creating multiple sessions or workspaces.
  Either the X or RFB protocol will be used to remotely deliver the application to the client, depending on the capabilities of the client. Either an IP address of specific device or a *zone name* can be provided as a destination. If a zone name is specified as a destination a suitable device in that room will be chosen.

- *Multiplex Module*
  The Multiplex module provides the ability to share or multiplex a session to multiple client devices in either view-only or collaborative mode. The multiplexing function is capable of taking one or more IP addresses of client devices to share a session with. Alternatively a *zone name* can be specified as a destination. In the case that a zone name is specified as a multiplex destination, all devices in the specified zone will be used. A list of active devices in specific zones are retained by the system making this feature possible. Multiplexing can be terminated by use of the multiplexStop() function.

- *Response Manager*
  The Response Manager will always receive a response from the Request Manager indicating either success or failure which is in turn sent to the client. The response is in XML format conforming to the communication standard specified for this system and supporting frameworks. The response consists of a response code as well as a textual description of the outcome of the request.

6. Evaluating Our Design

To evaluate our approach, there are several considerations to be made.

While some thin client protocols aim to minimise the processing requirements of the client, X is designed with the goal of minimising bandwidth by expending computing resources locally. Both approaches offer advantages, and the approach of minimizing local processing comes at the cost of higher bandwidth overhead. X is more suited to productivity applications rather than multimedia based applications. To address this problem, we can employ the RFB protocol, which performs well in relation to multimedia based applications, as discussed in [20].

We considered the overhead added by interposing a pseudo-server between X client and server. In [15], tcpdump (a tool which captures network packets and assigns a time-stamp) is used to establish the latency added by xmove as opposed to using a standard X server. Test results showed that xmove is virtually unnoticeable when communication between client and server is asynchronous; for example in the case of a colour page redraw, a delay time of 4% is added. In a scenario where communication is synchronous, meaning the client must wait for acknowledgement from the server between each message, the overhead becomes noticeable. The tests showed that for communication which involved a series of synchronous messages sent between client and server, xmove added an overhead of approx. 2 ms, bringing the roundtrip time from 3 ms to over 5 ms. This is significant, since it accumulates overtime. However, clients do not regularly communicate in this fashion; when they do it is often during start-up procedures or at other times when the user is expecting a delay, rather than time critical periods.

Preliminary tests showed that connecting to a VNC session via 100 Mbps LAN takes under 3 seconds, and once connected, session interaction is fluid. As discussed in [20], VNC performs well, maintaining frame rate over low bandwidth connections such as 1.5Mbps broadband. Tests showed that in comparison to other popular thin client protocols, VNC had the smallest memory footprint (less than 300kb) and an executable file size of just 172kb. The latency of VNC was also measured, showing that it was capable of completing tasks such as typing and mouse motion in under 150ms, making delay time unnoticeable to the end user.

From our evaluation, we feel that both VNC and X perform well over moderate bandwidth and are suitable protocols for working with thin client applications. Aside from the performance of the underlying protocols, the approach outlined offers the advantage of a rich heterogeneous environment in comparison to the alternative methods. When working with sessions containing a considerable amount of state, many users have resorted to saving state of a Virtual machine (VMWare for example) on portable storage devices. While users can run individual systems in parallel using VMWare's tabbed environment, these parallel environments lack consolidation and the task of switching between tabs quickly becomes cumbersome. The approach of running several entire operating systems uses considerable system resources, and furthermore resuming a virtual machine on a processor architecture which differs from the previous architecture is known to be problematic.

7. Conclusion and Future Work

There is seldom the ability for users to move their session from one device to another. Existing implementations of such systems tend to focus on homogeneous devices or lack support for legacy applications. X can be used to deliver applications to desktop class terminals. An advantage of X is that it delivers only a single application which integrates neatly in the windowing system of the current terminal. Many other thin client solutions (Terminal Services for example) deliver an entire desktop to the remote terminal, concealing the underlying operating system. The seamless integration offered by X allows remote and native applications to work side by side in a transparent manner. Mobile devices with constrained resources are generally not suitable for use with X based applications. Microsoft Windows Mobile does not support X applications, and there appears to be little work in the area of adding X support in the form of an extension to Windows Mobile. To support mobile devices such as those running Windows Mobile and Symbian OS, VNC can be used. VNC uses minimal client requirements is available for a wide number of client platforms. Merging these

protocols, we can provide a dynamically adaptive approach to session mobility.

By merging and extending existing thin client technologies and adding adding additional components to the system such as a knowledge management component, the ability to move sessions across a broad range of devices becomes evident. The ability to share this mobile session with multiple users provides a useful tool for presentations, teaching and collaborative work. Manually managing all of these technologies to provide such services is difficult and often impossible. In the past, such barriers have been a deterrent to the use of these technologies. By deploying the framework for session mobility within the pervasive computing model it is possible to significantly enhance the experience of power users while simplifying the experience of novice users in an unobtrusive and transparent manner. Deployment of this framework has proven that it is possible to provide session mobility to users across a broad range of devices in a seamless manner. Before this system could be deployed in a live environment, there are several further aspects which need to be addressed.

The need for load balancing between multiple servers is a fundamental issue which must be addressed before deploying the system in a large environment. Session transfer over wide area networks and low bandwidth connections has yet to be tested. Compression of X protocol messages could also significantly improve performance in such circumstances. Security is another crucial area of research which is yet to be explored in further depth. Adding UNIX style user and group permissions to the system is one approach. Other challenges include preventing dropped sessions due to broken network connections and the mandatory use of SSH tunneling on the client side.

References:

1. Bandelloni, R. and F. Paterno. *Flexible interface migration*. in *IUI '04: Proceedings of the 9th international conference on Intelligent user interfaces*. 2004. Funchal, Madeira, Portugal: ACM Press.
2. Johanson, B., et al., *Multibrowsing: Moving Web Content across multiple displays*, in *3rd international conference on Ubiquitous Computing*. 2001, Springer Verlag. p. 346-353.
3. Baratto, R., J. Nieh, and L. Kim. *THINC: A Remote Display Architecture for Thin-Client Computing*. 2004 .
4. Chu, H., et al., *ROAM, A Seamless Application Framework.* Systems and Software, 2004. **69**(3): p. 209-226.
5. Guyot, V., N. Boukhatem, and G. Pujolle, *Smart Card performances to handle Session Mobility.* Internet, 2005.The First IEEE and IFIP International Conference in Central Asia on, 2005. **1**: p. 5.
6. Microsoft Corporation. *Terminal Services Overview*. 2006
7. Richardson, T., et al., *Virtual network computing.* Internet Computing, IEEE, 1998. **2**(1): p. 33 - 38.
8. Scheifler, R.W. and J. Gettys, *The X Window System.* ACM Transactions on Graphics, 1986. **5**(2): p. 79-109.
9. GoGlobal Inc. *GraphOn GoGlobal for UNIX: Data Sheet*. 2006
10. Citrix Systems. *Citrix Presentation Server*. 2006.
11. Sun Microsystems. *Sun Secure Global Desktop Data Sheet*. 2005.
12. Scheifler, R.W. and J. Gettys. *X Window System: The Complete Reference to Xlib*. 1992: Digital Press.
13. Danskin, J.M., Q. Zhang, and D.M. Abrahams-Gessel. *Fast Higher Bandwidth X.* in *Multimedia and Networking*. 1995.
14. Sun Microsystems. *SunRay Overview*. 2004

15. Solomita, E., J. Kempf, and D. Duchamp, *XMOVE: A Pseudoserver for X Window Movement.* The X Resource, 1994. **11**(1): p. 143-170.
16. Bazik, J. *Sharing X Applications With XMX.* 1999.
17. Garfinkel, D., *HP SharedX: A tool for real time collaboration.* Hewlett-Packard Journal, 1994. **45**(2): p. 23-36.
18. RealVNC Ltd., *RealVNC Documentation*. 2006.
19. Adbel-Wahab, H.M. and M.A. Feit. *XTV: A framework for sharing X window clients in remote synchronous collaboration.* in *In Proceedings, IEEE Tricomm '91: Communications for Distributed Applications and Systems.* 1991.
20. Nieh, J., S.J. Yang, and N. Novik, *A Comparison of Thin-Client Computing Architectures*. 2000, Network Computing Laboratory, Columbia University.