A Service-based Approach to Supporting Portable, Persistent, Private Computing Environments

Dan MacCormac, Mark Deegan, Fred Mtenzi, Brendan O'Shea
School of Computing,
Dublin Institute of Technology, Dublin 8, Ireland
{dan.maccormac, mark.deegan, fred.mtenzi, brendan.oshea}@comp.dit.ie

ABSTRACT

As the trend of mobile devices continues to grow, combined with ubiquitous internet access, the need to explore new approaches to supporting device collaboration and consolidation becomes evident. We present a framework which demonstrates a service based approach to delivering persistent, mobility enabled, and private computing environments to a range of clients, regardless of the underlying hardware, platform or location. The framework is built on top of solid foundation of open source networking technologies, thereby supporting the widest range of clients in a flexible, efficient and robust manner. In this paper, we outline the motivation, key concepts, design and system architecture of the framework.

Key Words: ubiquitous, cloud computing, session, mobility, virtualization

1. Introduction

Ubiquitous computing is an emerging model of human computer interaction in which computing interaction is finely integrated into everyday objects and activities. This model succeeds the current desktop computing model, in which a user interacts with a single device for a specific purpose. Using the ubiquitous computing model, users can interact with multiple systems simultaneously, using a range of new and innovative interaction methods from motion detection to speech recognition. In a world of ubiquitous computing, this interaction is so finely integrated into everyday life that users may not necessarily realize the extent to which they are engaging with these systems. In Weiser's seminal paper on ubiquitous computing, he notes that before ubiquitous computing becomes a reality, there are three key requirements which must be satisfied [1]

- Cheap, low powered computers, both mobile and fixed.
- Infrastructures that allow these devices to communicate (networking capabilities)
- Software systems implementing ubiquitous computing applications

The first requirement has been well met to-date. People are now surrounded by an array of digital devices, which continue to increase in terms of capabilities, while decreasing in cost. Mobile computing has contributed strongly to this field, with a massive growth in the use of portable devices such as laptop and handheld computers. The second requirement, networking capability, has also been widely addressed to date. Wireless communications standards such as 802.11 Wireless networking [2] and Bluetooth [3] have strongly contributed. However, this growth in cellular and wireless technology has also brought tremendous incompatibility issues due to compartmentalization and isolation of standards [4]. This presents many challenges in relation to satisfying the third and final requirement; software systems implementing ubiquitous computing applications.

A session is a temporary information exchange between two or more points of communication, or between a computer and user. In the case of the latter, a login session is the period of activity between a

user logging in and logging out of a multi-user system. Currently, the average computing session is a short-term relationship between the user and the initial point of interaction (computer). The concept of a login session has become problematic and tedious in today's mobile driven world, and as result we propose the concept of a mobile-aware session to address this issue. A mobileaware session is a long-term relationship between a user and an end-point than may span across many devices and network connections. Mobile-aware sessions also eliminate expensive operations and overheads such as authentication, login process initialization, and synchronization across multiple devices. Most existing operating systems do not acknowledge such a concept; application instances are stationary, motionless and tied to particular system resources and processes. Applications are essentially un-grouped in this environment (albeit they are tied to a specific user ID), but not tied to an encapsulated mobile-aware session grouping; which we call a session container. We propose raising the entire end-users computing environment (including applications, network connections, user preferences) to a session container, which is platform independent, mobile aware, and highly adaptive.

2. Related Work

The issue of mobility has been well researched to date. As the range of devices and interaction complexity increases, the need to support mobility of a wider range of applications and devices becomes evident. Guyot et al. [6] investigate smart card performance as a token for session mobility between Windows and Linux. The system 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. However, this work does not address the use of mobile device. ROAM [7], an application framework that can assist developers to build applications that can run on heterogeneous devices and can allow a user to move a running application among heterogeneous devices is presented. Such solutions generally focus on providing mobility for a single application, or set of applications. Other approaches enable mobility of an entire desktop environment, supporting suspend and resume functionality and ubiquitous access. Nieh et al. [8] present MobiDesk, a virtual desktop computing hosting infrastructure. MobiDesk transparently virtualizes a users computing session, detaching the session from the underlying architecture and operating system, thereby allowing session to be suspended, migrated and resumed. The user can resume their session from any internet enabled device using simple thin client software programs. Similarly, Schmidt et al. [9] present SLIM (Stateless Low-level Interface Machine), which provides a network based desktop to light weight, fixed function terminals called SLIM consoles. The work promotes two concepts: Firstly, the authors propose the decoupling of displays from the server, sending data across the network as opposed to the traditional approach of the display and computer being coupled together. The second concept proposed by the authors is the management of computing resources at the back-end. To support this, the system must be scalable and capable of supporting dynamic load balancing. Additionally, the system must support user mobility, which allows a user to suspend their session, move to another terminal and instantly resume their session, in the same state as when they suspended it.

3. Design

The work focuses on development of software systems and applications to support the deployment of ubiquitous computing environments. Moreover, we aim to provide a framework to facilitate the seamless migration of computing

environments across a wide range of computing devices in a transparent manner. Such collaboration can help to enhance users' computing experiences' in addition to reducing total cost of ownership, administration, system-outages and power consumption.

3.1 Design Goals

Our framework aims to provide the following characteristics to support the concept of session abstraction.

- *Ubiquitous access*

Computing environments must be accessible from any network connected terminal, regardless of system architecture and underlying platform.

- *Persistent computing environment*

Computing environments reside on remote servers, and remain persistent even when users are not interacting with their computing session.

- *Mobile computing environment*

The design of traditional systems does not consider the issue of mobility. To meet the requirement of shifting the traditional personal computing model from a machine centric to a service oriented perspective, computing environments must be mobility enabled, allowing clients to interact with their session from any network connected client device.

- *Ease of Deployment*

Client deployment should require minimal experience and effort. In the same manner that a telephone handset simply needs to be connected to the phone jack, clients merely needed to be connected into the network and should attain all configuration information automatically. Similarly, deployment of software to clients should also require minimal effort, avoiding modification to existing applications and operating systems.

- *Private Computing Environments*

Users computing environments should be completely self contained. This prevents other users and processes from interfering with computing environments, as well as providing a finer granularity of customization and control for each user.

3.2 Approach

3.2.1 The Thin Client Model

To enable delivery of full desktop computing environments to remote users, we can employ the Thin Client model. The thin client computing model involves the use of a client computer or client software in client-server network architecture. The client device relies primarily on a central server for processing activities. The client is responsible for communicating input and output between the end user and the remote server. In contrast, a traditional desktop computer performs the majority of processing, communicating with remote servers intermittently. The thin-client computing model allows administrators to manage computing resources from a single centralized server, reducing hardware costs and maintenance time by an order of magnitude. A growing number of thin client protocols enable the delivery of full-fledged graphical user interfaces and entire desktop environments in order to meet the expectations of end users.

3.2.2 Virtualization

Virtualization is another existing concept that we incorporate into our framework.
The concept of virtualization is growing fast, and the benefits are becoming evident to both end users as well as system administrators. By leveraging this concept, we can provide self-contained, portable, sand-boxed environments to clients. This promotes the concept of privacy, customization as well as providing access to applications which were previously tied to a specific platform from a single point of communication.

The term virtualization is a broad term that refers to the abstraction of computer resources.

3.2.3 Web Services

As more and more devices become connected, wired and wirelessly, it is important for many of todays applications to communicate with other programs via the Internet. Previously, such communication was achieved using Remote Procedure Calls (RPC) between objects such as DCOM and CORBA. A better way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. However, HTTP was not designed to support this, and consequently the RPC approach poses a compatibility problem. In addition, many firewalls will normally block RPC traffic. SOAP was created to address this problem.

SOAP, originally an acronym for Simple Object Access Protocol (this acronym has since been dropped) is an XML based communication protocol which allows network based applications to send and receive XML based messages across the internet. SOAP is platform independent, simple and extensible. It is also capable of communicating in a wide range of environments regardless of firewall policy, since it communicates over HTTP on port 80. SOAP allows applications running on different operating systems, with different technologies and programming languages to communicate with each other. SOAP has been implemented in over 60 languages on over 20 platforms. [10]

4. System Architecture

We have implemented a prototype of the system in a Linux environment and created lightweight client applications for common client systems such as Windows, Linux, and Mac OS X. We present the system using a layered architecture. There are four layers associated with the framework; the application layer, the presentation layer, the session layer, and the transport layer. We will now discuss the role of each of these layers in further detail.

4.1 Application Layer

The Application Layer is the closest layer to the end user, and is the interface through which the end user communicates with the system. The application layer consists of two key components; the web server and the SOAP engine. For the web server, we have chosen to use the common Apache Web Server. This allows clients to communicate with the server from any internet enabled device via port 80. XML (eXtensible Markup Language) has emerged as an attractive protocol for information exchange between devices for a number of reasons. Some advantages include platform independence, language independence, unicode support, and support for common data structures.

In addition to the use of the Apache Web Server at the application layer, the framework also leverages Apache Axis to support XML based communication. Apache Axis is an open source, XML based Web service framework developed by the Apache Software Foundation. It allows developers to build interoperable, distributed computing applications. Axis consists of a Java and a C++ implementation of the SOAP server, as well as various utilities and APIs for generating and deploying Web service applications. We have implemented the application layer using Java, Apache Web Server and the Apache Axis framework. This allows clients to send XML based requests to the application via HTTP. The application layer then processes this information and passes the appropriate information to the presentation layer.

4.2 Presentation Layer

The presentation layer is responsible for communicating with other system layers, and formatting information appropriately for use by the lower application layer, or

the higher session and transport layers for further processing or display. In our framework, the presentation layer is comprised of two key components; the adaptation engine and the knowledge management component. The adaptation engine, which is implemented in Java, is responsible for communicating with adjacent system layers, and appropriately adapting the session environment to support a fine granularity of client devices. The Adaptation Engine defines variables associated with session delivery such as defining the most appropriate thin client protocol, and fine tuning of the chosen protocol(s). The second component of the presentation layer is the Data Knowledge Management (DKM) component of the system which is responsible for efficient management of network resources, user information, and domain information. The DKM engine is responsible for populating and querying resource databases, user preference databases, and domain information databases. This information is then passed to the Adaptation Engine, which in turn appropriately adapts the session information based on informed decisions of the DKM engine.

4.3 Session Layer

The Session Layer is responsible for establishing, managing and terminating connections between client devices, and the session hosting server. At this layer, we have implemented two components which manage connections between remote end users, and our framework. Firstly, the Connection Manger is responsible for creating, managing and terminating connections between clients and the appropriate back end servers. This process takes place in co-operation with the Adaptation Engine at the Presentation layer, and the protocols associated with the Transport layer. Secondly, the Session Manager is responsible for creating, maintaining, suspending, resuming, and terminating session. It is also responsible for defining session information.

4.4 Transport Layer

The Transport Layer is responsible for the transfer of data between session hosting servers and the end user client device. This is facilitated through the use of existing thin client protocols; namely X11, NX, and RFB, each of which is now discussed briefly.

The X11 protocol (also X Window System), is a windowing system that provides a graphical user interface for networked computers. X11, the current protocol version was first released in 1987 [11]. Unlike previous display protocols, X is designed to support network transparency, meaning that the machine on which the application executes and the machine on which the application is displayed may be completely separate, provided that they are both network connected. However, a regular X session, using common desktop applications generates hundreds of megabytes of data [12] which must be exchanged between client and server. NX is an application that handles remote X Window System connections, and attempts to improve performance of the X display protocol. NX works over port 22, compressing X traffic using various techniques such as message caching and differential encoding [12], allowing X to operate in a much more bandwidth efficient manner.

While X and NX are widely supported on various platforms such as Windows, Linux, and Mac OS X, there is little support for mobile devices. As result, we must consider alternative transport protocol as a means of delivering sessions to mobile devices. The RFB protocol provides the foundation for the popular VNC [13]thin client software, which enables access to remote computing sessions from a wide range of client devices. RFB is a stateless thin client protocol, which inherently supports mobility, since the client can terminate the connection at any point in time, and easily resume the connection at another point in time without losing session state.

5. Conclusion and Future Work

In this paper we outlined the problems associated with user mobility, and how this problem is compounded by the growing trend of mobile devices and platform heterogeneity. The ability to suspend and resume a session is seldom available to end users. Existing solutions tend to focus on specific applications, a single operating system, or a proposed toolkit or approach to developing mobile aware applications.

By leveraging existing thin client protocols, and adding presentation and data knowledge management components, we have demonstrated that we can provide a flexible and efficient solution to managing mobile aware session in the coming age of ubiquitous computing. Using this approach, it becomes possible to move sessions across a broad range of devices in a seamless and transparent manner. This alleviates the need for users to manually re-instate sessions, which can take several minutes; the aggregate cost of which is significant when interacting with numerous devices.

There are several areas of research yet to be explored in relation to the future development of the framework. The issue of security needs to be explored in further detail, as session privacy is a key concern. The efficiency of the DKM and Adaptation Engine components is crucial to the overall performance of the system, and as a result efforts have continued to improve performance in this area. Additionally, the performance of the system has yet to be evaluated in a real time environment.

Sensory tokens could also be used to store session state information, adding a further level of flexibility to system. As we enhance our work to date, in addition to evaluating other concepts associated with emerging model of ubiquitous computing, the unification of these individual approaches into a single diverse infrastructure will help to validate our vision of an intelligent, ever-connected, ubiquitous computing world.

References:

[1] M. Weiser, "The Computer for the 21st Century" Scientific American, vol. 265/ 3, 1991, pp. 94-104.

[2] IEEE, 802.11 Working Group, 2007.

[3] B. SIG, Bluetooth SIG Homepage, 2007.

[4] U.S. Jha, "Path Towards Network Convergence and Challenges," Managing Traffic Performance in Converged Networks, Springer Berlin / Heidelberg, 2007, p. 5.

[5] R. Bandelloni and F. Paterno, "Flexible interface migration," IUI '04: Proceedings of the 9th international conference on Intelligent user interfaces, New York, NY, USA: ACM Press, 2004, pp. 148–155.

[6] V. Guyot, 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, vol. 1, 2005, p. 5.

[7] H. Chu, H. Song, C. Wong, S. Kurakake, and M. Katagiri, "ROAM, A Seamless Application Framework," Systems and Software, vol. 69, 2004, pp. 209-226.

[8] R.A. Baratto, S. Potter, G. Su, and J. Nieh, "MobiDesk: mobile virtual desktop computing," MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking, New York, NY, USA: ACM, 2004, pp. 1–15.

[9] B.K. Schmidt, M.S. Lam, and J.D. Northcutt, "The interactive performance of SLIM: a stateless, thin-client architecture," SOSP '99: Proceedings of the seventeenth ACM symposium on Operating systems principles, 1999, pp. 32–47.

[10]S. Microsystems, Overview of SOAP.

[11]X.Org, X.org Licenses.

[12]N. Inc, NX X Protocol Compression, 2003.

[13]B. Shizuki, M. Nakasu, and J. Tanaka, VNC-Based access to remote computers from cellular phones, Institute of Information Sciences and Electronics, University of Tsukuba, 2002.