3C05 Advanced Software Engineering
CW2: Seminar Series

Title: "*Mobile Computing*"

John Tang

Introduction:

In the current day we take many parts of technology for granted including computers, people just don't care how these things work- just that it does. The problem comes when a new technological advancement or vision is made, in this case Mobile Computing, and the end users expect the same level transparency as they would from a fixed computer. Mobile computing is basically the ability to use desktop like applications but not fixed to one point. This could be your mobile GSM Smartphone running Microsoft Mobile Edition 2003 [1]; or your laptop with a wireless LAN (IEEE 802.11x) connection [2]; a PDA with both 802.11 and GSM capabilities [3]; or even more simply radio frequency tagging (RFID) as shown in BT's "ker'ching" concept [4], where items in a store are all RFID tagged for easy checkout etc. In the end, it is related to three main issues that are communication, mobility and portability. But first let us first set the scene with where mobile computing is heading with an example scenario.

Scenario (adapted from [5, 6, 7])**:**

Imagine in the near distant future you have spent the whole night finishing up your final project presentation which you need to give to the whole year group including a demonstration of the system. It's now 3am, and you decide to take a nap and finish the rest in the morning.

Mistake! You are woken by the vibrate on your Palm XM wireless handheld computer- It's the alarm and you are very late! You quickly get ready and rush out of the flat to the Tube station, totally forgetting to download any of your work to print off. Your Palm is running the latest "Angel" intelligent software, and the technology now kicks in. Angel infers from your calendar that today's importance and see's that the required files are missing from your Palm and so communicates with you home computer and starts downloading your dissertation and system code. It also senses you are late running later then usual to get into university, and

checks the status of the tube. There is a tube strike, but the intelligent software swiftly warns you of this problem and suggest a better route by bus. As you get on the bus and head towards university, you finish the rest of the presentation using voice commands. When you get into university, you sneak into the lecture theatre and Angel recognises that there is a projector available. You are next up and Angel starts to load the slides onto the projector for you.

You decide to quickly read through your presentation slides. You open the document on your PDA blissfully unaware that Angel has been working in the background. After rehearsing through them, it is your turn and Angel uploads the slides to the projector. You end up given a wonderful presentation and no one even knows that you were under prepared.

Pervasive Computing

A far fetched example perhaps, but this is the aim of Pervasive (what was known as Ubiquitous) computing first introduced in Mark Weiser's [5, 6, 7] seminal lecture from his paper on "The Computer for the 21st Century". As we can see from the example, the bottom line is transparency from the underlying hardware to the mobile device and then towards a higher level awareness that will be a culmination of several areas of research i.e. intelligent systems, networks, software engineering etc. There is this entity known as "Angel" that has this "context awareness" and so provides a service to the user based on the current situation. Currently, research is looking at individual contexts with no high level entity being able to choose and smoothly switch to a different context. The IEEE Pervasive Computing group [8] writes "Pervasive computing aims to integrate computation into our daily work practice to enhance our activities without being noticed. In other words, computing becomes truly invisible" [11]. There are many projects that try and meet this vision including "Planet Blue" [25] at IBM, "Project Aura" at Carnegie Mellon [26], "Oxygen" [27] at MIT, and the commercial "eHome" project at Microsoft. So this is the vision, the rest of the paper will look at how we are getting towards this goal.

Current Issues

There are three main types of problems with Mobile Computing that we will discuss as a path to getting towards pervasive computing. Firstly, compared to the current desktop PC

specifications, there are device hardware specific problems like a short battery life, low CPU speeds, low memory etc. Now in general, with miniaturisation, the PC spec of today will be the spec for mobile devices in the near future and so as a computer science point of view we will assume that electrical engineers can deal with this, and we will not look into it in any depth. However we will see how we can be aware of changes in the environment and accommodate them. The second point is network related, which includes factors such as temporary and unannounced loss of connectivity (i.e. if we are moving around), ad hoc host discovery, synchronisations etc. Lastly, we are also interested in software engineering related problems like data integrity, HCI, and how we can work with distributed objects in transactions.

Distributed System Requirements

Before we move onto the solutions to our two main problems, first let's discuss the requirements for both a fixed and mobile distributed system. Firstly some loose definitions. A fixed distributed system comprises a set of terminals, servers etc. connected in some network (be it a combination of LAN and WAN over the internet etc.). A mobile distributed system comprises devices such as mobile GSM phones, laptops, tablet PC's etc. that connect wirelessly to some network that may or may not be connected to a fixed network. Distributed networks have five main requirements [13, 14]:

1. Fault Tolerance – Transparent Recovery
2. Heterogeneity – Cross Platform Compatibility
3. Openness – System Extensibility
4. Scalability – Higher Loads
5. Resource Sharing – Share Printers, database etc.

Basically, all these issues mean that designing software (and the devices that they run on) is tightly coupled with the issues surrounding the underlying network. So for both fixed and mobile distributed systems, the software engineers would have to design the software to deal with the requirements above. However with a vast array of architectures, protocols, operating systems, services etc. there is no way that we can manage all. This brings us to the solution-

Middleware [12, 13] which deals with the underlying communication, mobility and portability issues.

Middleware:

Middleware basically sits on top of the underlying network operating system and provides a way for software engineers to concentrate on designing the software without getting bogged down with the issues that were outlined before. If we follow the ISO/OSI reference model, middleware basically implements the presentation and session layers of the stack (figure 1) and in doing so, it provides application developers with a higher level of abstraction built using the primitives of the network operating system [13]. Its main goal is to enable communication between distributed components and was used in fixed distributed systems before being taken up in a mobile distributed system, although we shall see that many of the assumptions we use in a fixed network cannot be ported over to a mobile environment and so new middleware was developed to deal with these inherent issues.

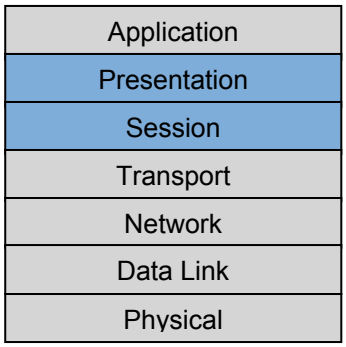

Figure 1

Firstly in a fixed distributed system we see that middleware assumes that disconnections are infrequent and are expectations when they occur. To deal with heterogeneity issues in a fixed distributed system, middleware provides a set of primitives that hide any underlying hardware or network implementation. In terms of openness, middleware provides interfaces that allow dynamic discovery of new functionality. To meet scalability issues, middleware uses replications of services through hierarchical structuring of lookup tables, which means we can transparently relocate services. Lastly, transaction servers check the integrity of data and middleware helps to ensure authentication and security.

As mentioned, with Mobile Distributed systems however, we find that many of the assumptions that we built on with a fixed network cannot be used again. For example for fault tolerance in a mobile network, we cannot assume that disconnections are infrequent, as a mobile device will most definitely be on the move. One approach from the middleware is to provide caching and buffering so that work can be continued if connectivity is lost. Technologies such as XMiddle [17] and Lime [18] deal with fault tolerance issues. Also as connectivity may unstable, we favour an asynchronous approach to data communication or even more advanced approaches like JMS (java messaging service [19]) for it's point-to-point and publish/subscribe communication model. This is similar to a email forum, where you can sign up to some forum and emails are sent to you about this topic.

Heterogeneity is essential in a mobile environment, as we may encounter many different types of hardware and protocols on the move. Middleware provides reconfiguration technique for dynamic adaptation, either handles by the middleware itself i.e. using a mobile port of the IIOP CORBA implementation; or by passing it to a higher level for the mobile host. GAIA [20] uses a universal interoperable core (UIC), that is composed of a pluggable set of components that allow developers to specialise the middleware targeting different devices and environments, thus solving heterogeneity issues [13].

Openness issues are apparent in mobile environments, as we are highly likely to meet new services and change contexts. Middleware such as Odyssey [21], adapts to changes in current resources such as low power, bandwidth etc., and to select an appropriate protocol for the current environment.

In terms of scalability, middleware is important for the ability to accommodate new hosts and services without degrading the network quality etc. XMiddle solves issues with host and service discovery by continuing monitoring their environment to discover what is available. To maintain a good quality of service (QoS), Mobiware [22] shows us that using a "service provision" scenario (similar to a mobile phone subscription), where mobile hosts are roaming but permanently connected, we have a simpler context to deal with QoS.

Lastly, middleware needs to deal with resource sharing issues such as security and transactions. Security is still an area of research one reason being that effectively there is no

central host for authentication etc. However to transparently deal with data integrity in transactions in a mobile context with unreliable connectivity, the Kangaroo [23] approach was introduced where the state of a transactions if recorded in a base station if connection is lost, and when the mobile device reconnected again (perhaps on a different base station) then the saved transaction "hops" to that base to continue the transaction.

Conclusion:

So we have seen the goal of mobile computing to provide seamless integration with mobile devices into our daily lives, but with this brings many problems. The current attempt is to use middleware as with a fixed distributed system, but remould our assumption so that it will work with a mobile distributed system. This works well but there are still many areas of interest open to research including security; dealing with a "lossy" network, where we cannot guarantee that data will arrive at it's destination etc. This stretches to a software engineer's perspective where we have to take into account these factors, and means that designing mobile applications is very different from a normal piece of software. For example we have a trade off between battery life and CPU intensity of applications. Whereas in a desktop, the CPU can handle quite an intensive load (perhaps from lazy software engineering), we have to be very efficient with algorithms and functions used to optimise the battery lifetime. We also have to take into account that due to the presumed unreliable connections, we have to somehow deal the data being modified on the mobile device, but now inconsistent with the original on the hosts. This is similar to the problems of "synching" your PDA to your PC, as you will only synchronise once a day while you work with data on the machine during the day, even if it is not up to date. Lastly, although ubiquitous (pervasive) computing was outlined over a decade ago development in this field is still slow and mobile computing as a whole is still in its infancy.

References and Further Reading:

[1] Windows Mobile Edition 2003 (http://www.microsoft.com/windowsmobile/default.mspx)

[2] IEEE 802.11x standard (http://standards.ieee.org/getieee802/802.11.html)

[3] Handspring Treo (http://www.handspring.com/products/communicators/treo600_overview.jhtml)

[4] "BT welcomes retailers to the virtual 'store of the future" Computing 2005 (http://www.computing.co.uk/news/1161658)

[5] "Pervasive Computing: Vision and Challenges", M. Satyanarayanan 2001 (http://www-2.cs.cmu.edu/~aura/docdir/pcs01.pdf)

[6] "Mark Weiser's Website" (http://www.ubiq.com/hypertext/weiser/weiser.html)

[7] "The Computer for the 21st Century", Mark Weiser 1991 (http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html)

[8] IEEE Computer Society: Pervasive & Ubiquitous computing (http://www.computer.org/pervasive/)

[9] "International Conference on Pervasive Computing 2004", (http://www.pervasive2004.org/)

[10] "Pervasive Computing", University of Utah, (http://www.cs.utah.edu/~sgoyal/pervasive/)

[11] "IEEE Pervasive Computing Group Guest Editors Introduction: Energy Harvesting and Conservation", Roy Want et al. 2005, (http://www.computer.org/pervasive/promo/promo2.pdf)

[12] "Principles of Mobile Computing Middleware", **Cecilia Mascolo**, Dept. Computer Science UCL 2004 (http://www.cs.ucl.ac.uk/staff/c.mascolo/www/principles.pdf)

[13] "Mobile Computing Middleware", **Cecilia Mascolo, Licia Capra and Wolfgang Emmerich**, Dept. Computer Science UCL 2002 (http://www.cs.ucl.ac.uk/staff/w.emmerich/publications/Networking2002/)

[14] "Software Engineering and Middleware: A Roadmap", **Wolfgang Emmerich**, Dept. Computer Science UCL 2002 (http://www.cs.ucl.ac.uk/staff/w.emmerich/publications/ICSE2000/SOTAR/)

[15] "Software Engineering for Mobility: A Roadmap", Gruia-Catalin Roman, Gian Pietro Picco, Amy L. Murphy, 1999 (http://www.cs.wustl.edu/mobilab/Publications/Papers/1999/99-31%20(Mobility%20Roadmap)%20ICSE%202000.pdf)

[16] "Engineering Distributed Objects (EDO99)", **Wolfgang Emmerich**, Dept. Computer Science UCL 1999 (http://www.cs.ucl.ac.uk/staff/w.emmerich/publications/ICSE99/EDO99/index.html)

[17] "XMIDDLE: A Data- Sharing Middleware" for Mobile Computing. Int. Journal on Personal and Wireless Communications, C. Mascolo, L. Capra, S. Zachariadis, and W. Emmerich 2002.

[18]. "Lime: A Middleware for Physical and Logical Mobility.", In Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS-21), A. Murphy, G. Picco, and G. Roman, 2001

[19] Java messaging service website, (http://java.sun.com/products/jms/)

[20] A Development Infrastructure for Active Spaces. In Workshop on Application Models and Programming Tools for Ubiquitous Computing (held in conjunction with the UBICOMP 2001) R. Cerqueira, C. Hess, M. Romn, and R. Campbell, 2001

[21] Mobile Information Access: Coda and Odyssey, M. Satyanarayanan, CMU (http://www-2.cs.cmu.edu/afs/cs/project/coda/Web/coda.html)

[22] The Mobiware Toolkit website, (http://comet.ctr.columbia.edu/mobiware/)

[23] "A Mobile Transaction Model That Captures Both the Data and Movement Behavior." ACM/Baltzer Journal on Special Topics in Mobile Networks and Applications (MONET) 2, 149–162. M. Dunham, A. Helal, and S. Balakrishnan 1997.

[24] "Challenges of Mobile Computing", George H. Forman, John Zahorjan 1994 (http://csdl.computer.org/comp/mags/co/1994/04/r4038abs.htm)

[25] IBM Planet Blue Project, (http://www.research.ibm.com/compsci/planetblue.html)

[26] Project Aura, Carnegie Mellon HCI department, (http://www-2.cs.cmu.edu/~aura/)

[27] "Oxygen" project, MIT, (http://www.oxygen.lcs.mit.edu/)