In the IEE International Workshop, Intelligent Environments 2005 (IE05), Colchester, UK, 28-29th June 05

SUPPORTING MOBILE SESSIONS ACROSS PERVASIVE SMART SPACE ENVIRONMENTS

A Shahi — University of Essex, UK
M Gardner — University of Essex, UK
V Callaghan — University of Essex, UK

ABSTRACT

Pervasive computing is at an exciting stage of its evolution, with an increasing number of devices of various shapes and sizes appearing in our everyday surroundings. In such an environment, the need to utilise application sessions within and across device rich spaces becomes apparent. This paper introduces the concept of mobile sessions for smart space environments, by outlining a candidate framework, OTIS (object transfer in smart-spaces), that specifically addresses session transfer in smart space environments, such as intelligent buildings. Having done this we compare OTIS to the AURA and GAIA infrastructures, which most closely relate to our work. Finally we summarise the main findings of our research and outline our plans for taking this work forward.

1. INTRODUCTION

Pervasive computing causes us to examine ways in which existing computing infrastructures combine with everyday physical and environmental spaces, by understanding the dynamics of device rich environments, how devices are networked to correspond to the boundaries of physical space and how users generally interact within and across these spaces: commonly referred to as 'smart space environments'. Generally, users and mobile devices interact with a smart space by autonomously joining the space, performing some form of interaction with any space specific services, and leaving a space in a seamless manner. Interaction and use of a smart space may differ depending on its type. For example, smart spaces may be private, social or public; such as a private room in an intelligent building or a public meeting place.

Smart spaces of the future will allow users to seamlessly access and use services across the myriad of devices provided by each space. Achieving this level of seamlessness requires true interoperability across heterogeneous devices, networks and applications. Much of this work is being lead by standards bodies, which recommend their own standards for addressing interoperable systems needed in smart space environments; including various types of networking technology, device and service middle-ware, and methods for assigning semantic meaning to network resources. All these technologies are well known for forming an integral part of any ubiquitous computing environment, with the challenge being to combine these to offer new types of behaviour; characterised by being considerably more powerful and seamless than services today.

We envision one such service being to take existing applications, associated with a user or space, mobilising these applications into transferable sessions, and then allowing these to be transferred between devices in a space. Our findings come from earlier industrial work in making web browsing sessions mobile across multi-device environments, and the resulting approaches acquired, together with lessons learned, in making an application move its active state to another device. This has then been combined with the concept of smart spaces for pervasive computing environments, to provide an overall framework for discussion, OTIS, which we hope will generate interest in both the intelligent building and wider pervasive computing research community.

Firstly, we acquaint the reader with the underlying concept by studying three distinct scenarios. We then identify techniques in moving application state and present OTIS, our sample framework for session transfer in smart space environments. Finally, we compare OTIS with two similar systems: AURA and GAIA, before outlining future work.

2. SCENARIOS

A few scenarios should help to narrow down the concept of session transfer in smart space environments:

Scenario 2.1. Jane is in the living room working from home. She has a range of applications displayed on her living room screen, such as a VoIP phone, and a word processor application. Moments later, Jane's house-mates enter the living room. Since Jane is busy working and talking to her work colleagues, she transfers her desktop session from the living room screen to her study screen, and resumes her work in the study.

Scenario 2.2. Next morning, Jane needs to attend a conference in Tokyo. After arriving at her hotel room, a symbol on Jane's phone starts to flash in an unobtrusive manner. Jane now knows she's within a 'smart space'.

Using her phone, Jane selects the smart space menu, which has now become 'active' by the phone implicitly merging itself into the hotel space. After an authentication procedure between Jane's smart phone and the smart space, Jane is presented with a menu list of services available to her. One of these services is 'Home Desktop'. Jane selects this menu, which then causes her home desktop to appear on a terminal screen in the room. She uses an instant messaging application to tell her boyfriend that she has arrived safe and sound.

Scenario 2.3. After arriving at the conference, Jane walks into the conference theatre. Immediately, Jane's smart phone is added to the conference space, with Jane being alerted by the phone displaying a space symbol. Jane uses her phone to browse the space, and finds a conference proceedings section. Whilst being seated, Jane requests the conference proceedings. The conference space then requests a prioritised list of formats supported by Jane's phone and, as result, sends the details in HTML form.

Each of the scenarios differ in terms of their context. However, they all have one thing in common, which involves the transfer of application sessions within smart space environments. Furthermore, the transfer process is done at the touch of a button, hence being invisible to the user. We believe scenarios such as these can only be realised by examining different approaches to state migration, together with their strengths in favouring certain scenarios. Three approaches are now described.

3. APPROACHES TO SESSION TRANSFER WITHIN SMART SPACE ENVIRONMENTS

Any system concerned with session transfer must clearly identify the objects needed for mobilisation. We refer to these objects as application sessions: anything an application may transfer, handle and present to devices within a space. Sessions may include personalised state, such as the state of a user's desktop environment (scenarios 2.1, 2.2), or the current state of an application. Sessions may also be stateless and de-coupled from a user or application, such as a file or snippet of web content, e.g the conference session within scenario 2.3, where the session consists of an ordinary file belonging to a shared space, together with a web page. Application sessions may be requested or pulled by a user in a space (e.g. using a mobile device) and then pushed using relevant protocols. Sessions may also be pushed according to contextual information, such as a user's location within a space (e.g. moving across rooms in a building).

Techniques for migrating and pushing application sessions will now be provided, along with their strengths in favouring specific smart space scenarios and application types.

3.1 Server based redirection

Server based redirection is concerned with running users' application sessions within a local, centralised server environment, and redirecting input/output channels, such as display and sound, to and from networked devices. This approach is taken in remote desktop systems (11) (12), where personal computing environments are directed to whatever device a user wishes to use. These systems include specialist protocols that split a host computer's low level input/output devices into separate network aware channels. Input/output is therefore piped over these network channels to a thin-client residing in a device. In some ways, server based redirection resembles previous dumb terminal to main-frame models.

Server based redirection for smart space environments will typically be used to move complex user sessions to and from devices within a smart space, as described in scenario 2.1. Since application and user environments will reside within a server node, server-based redirection will allow the transfer of runtime application state, such as a desktop, to any device in the local space. A change of context, such as moving to a different room in the building, could cause the server node to redirect and transfer all output to the user's new device.

Although server based redirection can capture and transfer real-time session state, such as a user's desktop environment; network connectively and latency must remain optimal, which is adequate for transfer in local spaces, but may cause problems when connectivity is unavailable; such as accessing server sessions from more remote locations. Another approach, virtual machine driven transfer, transfers actual session state between session servers located in separate spaces, therefore being more suited to nomadic and remote interaction. This method will now be examined.

3.2 Virtual machine driven transfer

Early work in mobile agent systems envisioned software agents migrating themselves between devices to perform user tasks, such as information retrieval. Important work here included the need to migrate agents between heterogeneous device architectures, which in turn was addressed by employing interpretative environments, such as virtual machines.

Process migration techniques using virtual machines can provide an alternative form of session state transfer, suited to scenarios of uncertain network connectively and latency, together with times when one may wish to fully import their personal computing environment into another space, therefore making full use of any local resources in the space. One such system, Internet suspend/resume (4) provides virtual machine based state transfer using commercial strength virtual machine technology. Virtual machine monitors (VMMs) are deployed to examine the volatile state of a user's computing environment. When a state transfer is required, e.g. from a user's home PC to a work PC, the VMM will push all volatile state to a distributed file system - whilst the user makes the transition from home to work. A VMM on a target host will then be notified, and pull any relevant state, hence restoring a user's active computing environment as it was left. Combined with smart spaces, virtual machine transfer may be used in nomadic situations where a user is continually moving between spaces and wishes to pull his computing environment, as he left it, into the current space (scenario 2.2). This way, utilisation of powerful local resources is maximised, rather than wasted by treating these resources as thin-clients, as in server based redirection. Figure 1 illustrates virtual machine based transfer. As shown, a user has a session, B within intelligent building A. The user then moves to another intelligent building environment, B, and requests session B. Using virtual machine transfer, the user's session is imported into intelligent building B, and displayed on a terminal within the user's space. Both

Figure 1: Virtual machine driven transfer across smart spaces

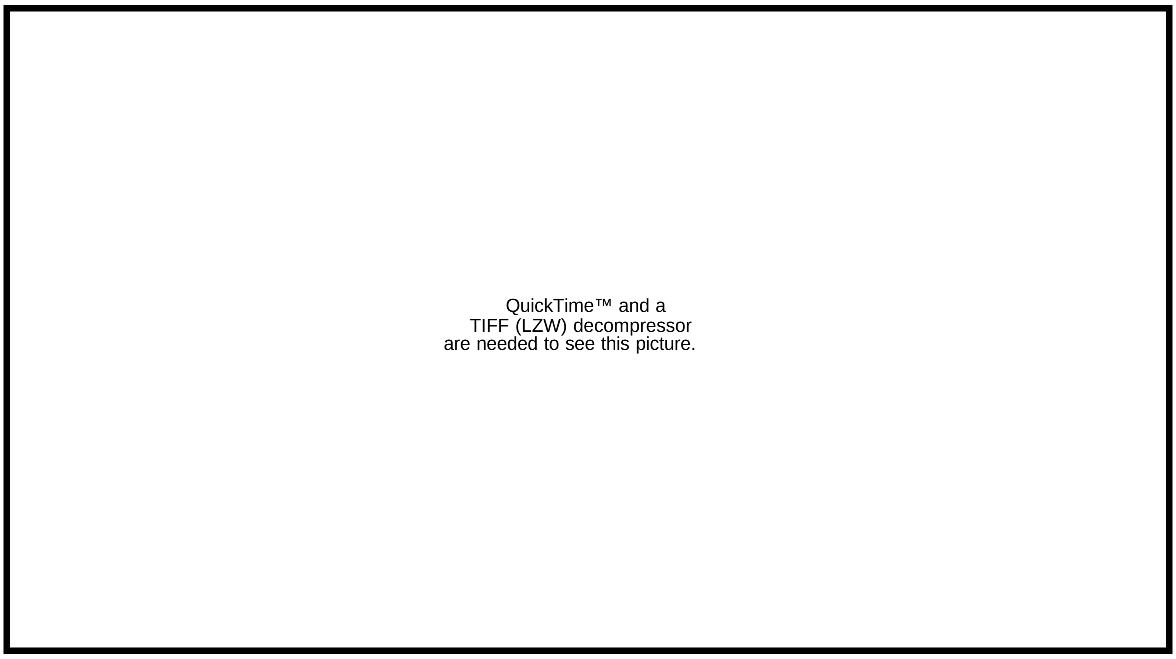

QuickTime™ and a
TIFF (LZW) decompressor
are needed to see this picture.

Figure 2: Transferring data objects to a mobile device in a smart space

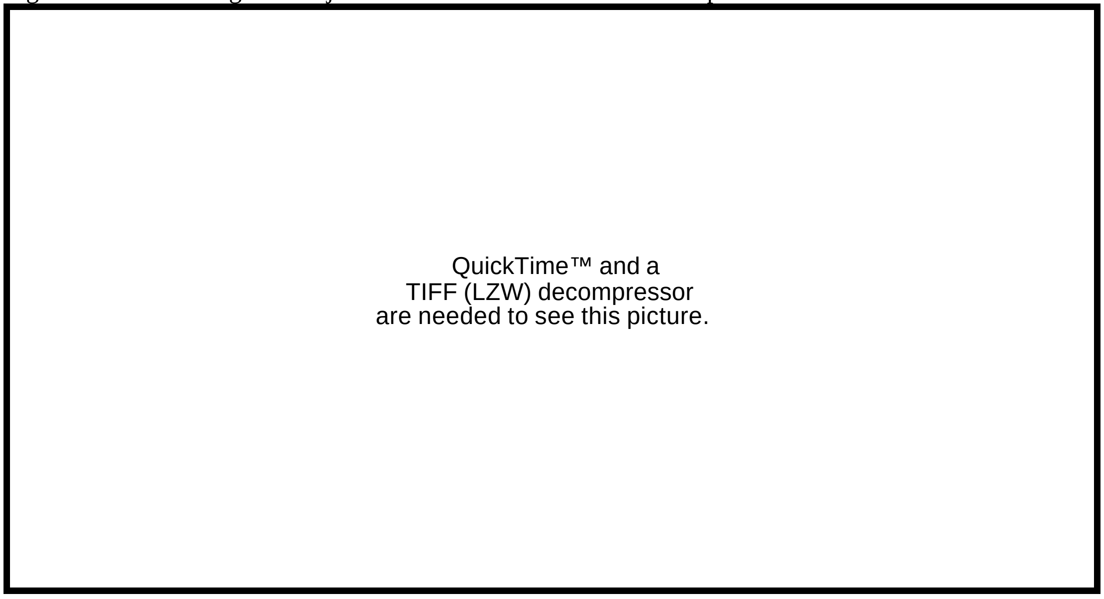
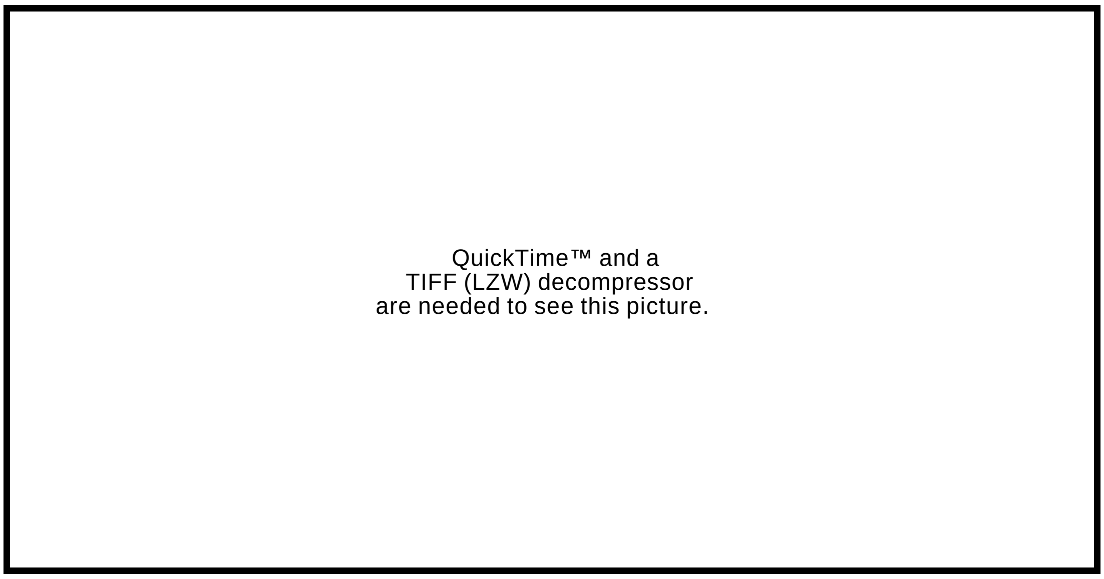

QuickTime™ and a
TIFF (LZW) decompressor
are needed to see this picture.

Figure 3: Controlling redirection of a user session within a smart space



server-based redirection and virtual machine driven transfer prove useful for transferring sessions with complex state, such as a user's personal computing environment. Sometimes, especially in public space scenarios, we may wish to transfer smaller, finer grained sessions called data objects. Typically, data objects will correspond to the state of a particular application or to local files or URLs. Transfer of data objects will now be examined.

3.3 Data object transfer

Data object transfer is concerned with capturing small, fine-grained data objects and pushing these to a relevant device. To do this, protocols must be utilised that support pushing of objects, such as OBEX Push (10) or WAP-Push (13) for instance. Each data object must also be augmented with META-DATA describing itself, e.g. its MIME type value. This allows the receiving device to process a data object accordingly, just as a web browser handles MIME types.

Embedding data objects into a space requires some form of logical encapsulation regarding available data objects for transfer. We call this encapsulation a data box. Figure 2 includes a data box for a conference space. As shown, the data box groups various data objects that point to files or individual application sessions. For example, object B and object C point to a URL describing the conference agenda and a PDF file of the conference proceedings. object A points to a running application, X, which keeps track of the active speaker. This data object is therefore referring to the run time state of application X, i.e. current speaker information. A device within the space could request object A, causing the space to extract the run-time state of application X, via object A, and transmit this information as a VCard, using a suitable PUSH protocol.

Devices entering a space will handle a certain set of protocols for object transfer, therefore requiring the space to determine which protocol to use when transferring an object. For example, a space may ask a smart phone device for the various PUSH services it supports. The space could then determine that the device supports the OBEX File Push protocol, hence choosing to push a data object over OBEX.

Each of the outlined approaches is suited to different usage scenarios. Server-based redirection is typically used for directing the output channels of a user's server session to devices within a space (scenario 2.1). Virtual machine transfer is used by nomads to import their computing environment into a new space, therefore allowing session movement across spaces, such as between built environments (scenario 2.2). Finally, data object transfer is useful for seamlessly pulling snippets of information associated with a space (scenario 2.3) Overall, the challenge lies in taking these approaches and their respective strengths in favouring particular migration techniques, and combining them together under the context of smart spaces, such as intelligent buildings. OTIS is our sample framework for achieving this, and will now be described by outlining its main components.

4. OTIS: OBJECT TRANSFER IN SMART SPACES

OTIS will encapsulate all session transfer services within a smart space, and provide adequate support to transfer application sessions within and across spaces. Each of OTIS's components will now be described:

4.1 Session transfer

All processing related to transferring application sessions is handled by the session transfer (ST) component. This component includes control mechanisms to carryout the following:

—Server based redirection: Redirect the output channels from a server-based session to appropriate devices within a space. Figure 3 illustrates the control mechanisms used for server-based redirection. All user sessions are stored within a session server. When a user wishes to move a session to a different device, the server simply drops the connection to the current terminal device, and re-instantiates a connection to the target device by requesting the thin client to connect to the user's server session. An appropriate protocol is used to send display and audio output to the terminal device, together with input events from the terminal device to the server session. Note that terminal devices may take the form of PADs, Tabs or Boards, as described by Weiser (5).

—Virtual machine transfer: When a user moves to a completely new environment, such as a different building, this component will provide control mechanisms to import users' computing environments into the current space. Control mechanisms will typically be based on already existing approaches such as Internet/Suspend and Resume, whereby a user's computing environment will be stored in a distributed file system. The session transfer component (ST) will then act as a virtual machine monitor by accessing the distributed file system, importing a user's computing environment into an appropriate VM, and creating a server-based session for the user. Since the imported environment will be represented as a server-based session, server-based redirection will be possible between devices within the local space. The ST will also provide mechanisms for restoring a user's server session back to the distributed file-system, therefore allowing retrieval from another environment.

—Accessing space specific data objects: A data-box will hold any space specific data objects intended for transfer. The ST component will then mediate access to and from the data-box by interacting with the mobile device mediator component. Using the mobile device mediator, the ST component will be able to determine the various PUSH protocols supported by a device. It will then use an appropriate service to transfer a data object over an appropriate protocol. For example, a web service could be used to send a data object corresponding to a URL, with WAP-Push as the underlying PUSH protocol.

4.2 Mobile device mediator

Mobile device mediators (MDMs) will typically be used to allow mobile devices to enter and merge into a space. Users will typically use their mobile devices to retrieve data objects available within the current space (scenario 2.3). Alternatively, a mobile device could be used as a remote control for seamlessly triggering the retrieval of a user's computing environment into a stationary device in the space (scenario 2.2). The MDM works by performing sever beaconing via one of its sensors, therefore detecting any mobile devices within the current space. Different wireless technologies may be used depending on the granularity of a space. For example, one may wish to split a room into lots of small tiny zones using sensing technology such as RF-ID. Alternatively, the boundary of a space could span the whole building using Wi-Fi technology, or employ bluetooth like technology; since the range of bluetooth corresponds to the theory of our behaviour being associated with the room that we are in. Once a device has been detected and authenticated, the MDM queries the context model and retrieves all session services (e.g. available data objects) within the space. These are then sent to the mobile device for display. A user may select a particular session, causing an event to be sent to the MDM, which then relays the event to the ST component thus causing a data object being sent to the mobile device, or a user's computing environment being transferred to an appropriate device within close proximity to the user.

When pushing data objects to a mobile device, the ST component may require the type of underlying PUSH protocols supported by the device. Using service discovery mechanisms, such as bluetooth SDP (7), the MDM may discover any PUSH protocols supported by the device, and relay this information back to the ST component, therefore allowing the ST to adapt appropriately.

4.3 Context model

Many forms of context are required when transferring sessions using the identified approaches. It turns out that each approach requires its own type of context from a space:

4.3.1 Modelling server based redirection. Using server-based redirection, a user may wish to transfer their current, active session, to another device within the local space. To do this, the context model must represent all active devices within the local space. Typically, protocols such as UPnP (6) may be used to discover devices by multi-casting service discovery requests within the local space. The context model may then cache returned XML service advertisements, to provide a snapshot of current device context within the local space.

4.3.2 Modelling virtual machine transfer. Using virtual machine based transfer, users may request their personal computing environment to appear on a terminal situated near themselves. In order to do this, we must be able to model the location of a user with respect to the terminal they wish to use. Many technologies may be used for this task. For example, near-communication (3) may be used, where a user simply touches the relevant terminal using a mobile device. A request containing the information regarding the user and the device touched, is then sent to the session transfer component, which retrieves the user's environment and uses server-based redirection to direct output to the device selected by the user.

Alternatively, stereo computer vision may be used to identify users (9), and place them within a geometric model, where entities are positioned in a space, along with their relationships and more importantly, their 'extent'. Using a geometric model, the session transfer component may determine all terminals situated close to the user and transfer a session's output to these terminals.

Since the ST component will act as a VMM, and access a distributed file system, appropriate user credentials must be available for establishing a secure connection. Typically, these credentials will be stored on a user's personal device, such as a smart card.

4.3.3 Modelling data objects. A data box may hold various data object sessions linked to applications, URLs or files. To make these part of a space, these sessions must be represented in a form that allows the MDM to make their availability known to a mobile device entering a space. One way to model data objects is to use the W3C standard, CC/PP: a framework for contextualisation, where profiles may be defined based on various objects such as people, devices, locations and applications etc. Profiles are organised into components, which are described using attributes. Since CC/PP is based on the resource description framework (RDF), components and attributes are identified using XML name-spaces; thus allowing profiles to be formed from heterogeneous components. CC/PP may be used to develop a profile for data object sessions. A session could be described by the approach taken in modelling context using CC/PP (2):

```
—[SessionProfile
        [User [SessionID, URI]]
        [Application [ URI ] ] [Location [URI]]
        [Display [URI]] [Transformer [URI]]
        ]
```

Here, a session profile includes pointers to the session author, along with a session identifier. An application URI describes the type of session and may correspond to a MIME type. Location points to the actual data object, which may be a file or a call to an application's interface. Display is used by the MDM for displaying data object presence, hence allowing the user to select an object for transfer. Finally, transformer allows data returned from the location URI to be transformed into a specific format, e.g. converting personal info into VCard format. Typically, many 'transformer' components will exist, with the session transfer component being able to select the correct transformer, depending on the transfer format required.

5. COMPARISON TO OTHER APPROACHES

We now compare OTIS to two similar pervasive computing frameworks: AURA and GAIA.

5.1 AURA

Out of all research being conducted on application migration within ubiquitous computing environments, the AURA project (8) seems to relate most closely to this work. AURA is a ubiquitous computing infrastructure specialised towards giving users the ability to seamlessly move their computational tasks across environments, such as buildings. AURA monitors a user's activities and location by deploying a context observer. When a user changes context, such as moving to another environment, AURA uses a task manager, PRISM, to control migration of any tasks to the new environment. PRISM interacts with a local environment manager to determine whether tasks can be restored using the resources and task suppliers of the new environment. If so, a user's environment is restored by using a distributed file system to retrieve any personal state. AURA is thus concerned with a computer environment which follows a user by deploying separate PRISMs in each new environment.

Examining AURA's migration techniques, all applications intended for migration are abstracted into high-level tasks. These tasks are then mapped onto service suppliers which wrap existing applications. When task migration is required, a user's current tasks are captured and saved using each active application's service supplier. Within a new location, a PRISM restores tasks by consulting an environment manager for any local suppliers that can handle the tasks a user wishes to import. Since applications are abstracted into high-level descriptions, application migration across completely heterogeneous environments is possible, such as moving from Word in Windows to Emacs in UNIX. The main problems with this approach come from the need to define custom suppliers or wrappers for every application supported, and then map these to high level task definitions. This may be a problem when considering the many applications being used, and introduced into everyday computing environments. Building wrappers around every existing application can also be a complex and time consuming process. We know this from experience, as we have built a prototype system, Tele-Web, that allows web sessions to be tele-ported between different web browser applications. Here, the task of abstraction is a 'Web session', as defined in HTTP. The aim is to build wrappers around individual web browser applications, and map these to an abstracted web session definition. Wrappers allow access to each browser's web specific session state, such as Form Field Values, Bookmarks and Cookies etc. Two application specific wrappers were built for both Internet Explorer and a custom Java based web browser. In order to achieve the relatively simple process of transferring web state from Internet explorer to the Java browser, much effort was required in terms of building individual application wrappers since each browser holds internal state in a different way. Furthermore, some applications provide only limited access to their internal object models; the ones that do are certainly not intuitive and take time to understand. In time, we decided that the best way to move application state in pervasive computing environments was to combine a mixture of state transfer techniques, as supported by OTIS. Finally, we believe AURA's ability to move application tasks between environments such as Windows and Linux may prove powerful; however, users generally familiarise themselves with applications over time, and may feel slightly puzzled if their session is restored in a different application, which incorporates a completely different usability model.

OTIS supports migration of a user's complete environment as they left it, therefore moving all applications belonging to a user. Since migration across spaces is performed at a virtual machine level, the need to implement custom wrappers for each application is eliminated. OTIS does require each environment to run user sessions within a VM environment and deploy OTIS components for state transfer in a smart space. This is however, analogous to AURA requiring its own components, such as PRISMS and Environment Managers to be deployed in each environment. OTIS is also different from AURA in that it takes a different approach by combining the strengths of different state migration techniques and integrating these within the context of smart space environments. The framework behaves like AURA when using virtual-machine transfer, but provides other migration techniques for use within a space, such as server-based redirection and data object transfer, which are required to support scenarios, such as 3.1 and 3.3 respectively. In terms of interaction, AURU tries to anticipate a user's next location, and automatically transfer an environment to that location. OTIS includes a mobile device mediator to allow a user to summon a personal computing environment by using a mobile device as a remote control (scenario 3.2). Since mobile devices are now incorporating contact less smart card technology, it appears perfectly feasible to hold a user's credentials on a smart card, such as user-name/password and encryption keys for accessing a distributed file system using VM transfer.

5.2 GAIA

GAIA (1) is a comprehensive ubiquitous computing infrastructure for the creation of 'active spaces'. The main idea behind GAIA is to provide a programmable metaoperating system that co-ordinates software entities and heterogeneous devices contained in a space. A key part of the GAIA framework is its context-aware file system, where context is addressed through file name-spaces and files are made available from personal (remote server) and space based storage services (local server). This concept is very similar to data object transfer and the data-box concept, although our framework emphasises the use of a mobile device mediator (MDM) to detect mobile devices, such as smart phones, and push data objects by selecting relevant PUSH protocols. GAIA's context aware file system on the other hand expects users to mount their own file systems into a space, together with creating mount points into the local space. This is quite different from the more seamless interaction scenario found in 2.3.

Generally, GAIA aims to provide a ubiquitous computing middle-ware, rather than a specific service as with OTIS. We believe GAIA's context aware file system could be combined with OTIS's MDM and data-box, hence allowing data objects to be transferred to a mobile device using contextual information.

6. DISCUSSION

We have introduced OTIS, a framework created from our earlier work in making web browser sessions teleport their active state between heterogeneous devices. OTIS has been compared with similar frameworks, such as AURA and GAIA, and differs by taking a different approach to state migration within smart space environments. We believe that determining which framework is best suited to application transfer in pervasive computing, requires careful, user-centred evaluation examining the following:

—Do users find OTIS's session transfer techniques useful? What needs to be changed?

—Do users want sessions to appear instantaneously, or do they prefer to manually request sessions?

—Do users prefer the OTIS or AURA approach to session transfer? Can these approaches be combined to offer superior behaviour?

These are the type of questions we expect to guide future research by implementing parts of OTIS and introducing server-based redirection and data-object transfer within our new intelligent building testbed: the iDorm2. Currently, a sample OTIS data-box is being implemented that allows a bluetooth enabled device to interact with a space via an MDM, and request any relevant data object sessions. OBEX and WAP Push are currently being used as the underlying protocols to PUSH objects to a device.

Overall, we have outlined the benefits of OTIS in combining different state transfer techniques, and believe the challenge lies in creating a system that can support the distinct advantages that server-based redirection, virtual machine transfer and data object transfer bring to pervasive computing environments. In time, we aim to evaluate our prototype systems using different usage scenarios. We hope this will give us insight into evolving our framework for future work. We are also interested in exploring the broader, social aspects of how people interact with, and use space within intelligent building environments.

REFERENCES

1. M. Roman, C. Hess, R. Cerqueira, A. Raganat, R. Campbell, and K. Nahrstedt. 2002. Gaia: A middleware infrastructure to enable active spaces. *IEEE Pervasive Computing*, 1(4):74–83.

2. J Indulska, R Robinson, A Rakotonirainy, and K Henricksen. 2003. Experiences in using cc/pp in context aware systems. *Proceedings of the 4th International Conference on Mobile Data Management*, pages 247– 261.

3. ECMA International. 2004. Near field communication white paper. Technical report, NFC Forum, 2004.

4. M. Kozuch and M. Satyanarayanan. 2002. Internet suspend/resume. In *Fourth IEEE Workshop on Mobile Computing Systems and Applications*, pages 40-46

5. M. Weiser. 1991. The computer for the 21st century. *Scientific American*, 265(3):94–104.

6. White Paper. UPnP device architecture. 2000. Technical report, UPnP Forum.

7. S. Avancha, A. Joshi, and T. Finin. June 2002. Enhanced service discovery in bluetooth. *IEEE Computer*, 35(6):96–99.

8. D. Garlan, D. Siewiorek, A. Smailagic, and P. Steenkiste. 2002. Project aura: Towards distraction-free pervasive computing. *IEEE Pervasive Computing*, 21(2):22–31.

9. B. Brumitt, B. Meyers, J. Krumm, A. Kern, and S. Shafer. 2000. Easyliving: Technologies for intelligent environments. In *Proc. 2nd Int'l Symp Handheld and Ubiquitous Computing (HUC 2000)*, pages 12–27.

10. Extended Systems. 2003. Obex protocol v1.3 technical specification. Technical report, Infrared Data Association.

11. T. Richardson and K. Wood. 1998. The rfb protocol, version 3.3. Technical report, ORL, Cambridge.

12. Microsoft Whitepaper. 2000. Remote desktop protocol (rdp) features and performance. Technical report, Microsoft Corporation.

13. WAP Forum Whitepaper. 2002. Wap 2.0 technical white paper. Technical report, Open Mobile Aliance.