An Architecture for the Integration of Physical and Informational Spaces

Scott M. Thayer and Peter Steenkiste

Carnegie Mellon University, 5000 Forbes Avenue
Pittsburgh, PA 15213
{sthayer, prs}@cs.cmu.edu
http://www.cs.cmu.edu/~aura

Abstract. While computer processing power, storage space, and bandwidth capacities are experiencing exponential growth, individual human processing capabilities are not increasing significantly. Pervasive computing creates an environment that offers a wealth of computing resources, I/O capabilities, and sensors. This offers an opportunity for applications to interact with and monitor the physical environment and to provide a task-centric and mobile infrastructure for the modern user. However, this rich environment can also be overwhelming and distracting to users, in part because of a disconnect between the physical infrastructure observed by users and the information space seen by applications. In this paper we introduce AIPIS, an architecture for a technological bridge between the physical and informational realms of the human and the computer, respectively. The purpose of this bridge is two-fold: (1) to provide to users a hands-free computing environment that automates much of the drudgery associated with use of computers, and (2) to focus human attention to only the critical aspects of task execution that require their input. We also describe the implementation of the Aura desktop, a first prototype of the AIPIS architecture.

1 Introduction

Much of today's computing systems and software are application centered. Commercial pressures demand well-defined, shrink-wrapped products that can be readily and repeatedly marketed to consumers in part drive this situation. With great diversification in the software market, the typical computer is loaded with software from dozens of different vendors. Some application integration is available, but typically it is within a select bundle of applications from a major vendor with proprietary interfaces. The problem with this model is that while performing tasks, users typically do not work within the framework of a single application or even an application bundle. On the contrary, users typically coordinate diverse applications in the daily execution of tasks and projects under their charge. For example, the preparation of a multi-author conference paper typically involves email and phone applications, spreadsheets, word processors, schedulers, scientific simulation and mathematics packages, photo editing, typesetters, etc. The user wants a seamless interface between applications that can be tailored to each particular task and configured with the current set of

user preferences. Even with the recent, rapid progress resulting in increasingly powerful computing platforms and software, the state-of-the-art in system-wide, task-level interfaces to computing and software applications from distinct vendors is somewhere between ftp and file copy. The level of distraction and frustration associated with using a computer to coordinate a reasonably complex task can be disconcerting for the veteran and novice user alike.

An application-centered approach can never achieve the level of efficiency and utility necessary to liberate people from computer drudgery in a task-based world. This is a natural consequence of an approach that attempts to satisfy the needs of an average user base with a common, commercial software base. In Aura [1,2,3], we embrace an approach that attempts to weave applications into an infrastructure that is pliable enough to represent user preferences and mobile enough to support user on the go and intelligent enough to operate as viable resource in the execution of day-to-day tasks [12]. Our approach to realizing this vision centers on the following principles:

Embedded Users. Users are embedded in a physical world and Aura must operate within the limitations, constraints, and distractions of that world.

Mobile Users. Users are mobile and move within the physical world during the execution of tasks.

Task-Centered. Most users are not application or even bundle centered – they are task centered and currently lack an intelligent interface that helps in the automated execution of comprehensive user tasks.

One challenge is that we have to bridge the gap between two currently distinct domains: informational and physical. Applications interact various sources and sinks of information streams, while users interact with devices that have certain properties (physical location, capabilities). Typically, these domains are not tightly coupled and as a consequence, current computer interfaces inhibit the optimal benefit to the user. In this paper we present an architecture called AIPIS, for Architecture for Integrating Physical and Informational Spaces, that bridges the physical and informational spaces in such a way that it becomes easier to develop applications that can interact with humans in a more natural manner. In the next section we first give an overview of the CMU Aura project, which provided the motivation for AIPIS. We then present the AIPIS architecture and we describe a prototype implementation. We conclude with related work and conclusions.

2 Aura: Ubiquitous Invisible Computing

The Aura project is evolving a ubiquitous invisible computing infrastructure that supports mobile users in performing every day tasks. Ubiquitous means that Aura is present everywhere, although the level of support may vary significantly. For exam-

ple, device-rich smart rooms will offer a wider range of modes of interaction than elevators and parks. Invisible means that Aura will minimize the level of distraction that it imposes on users. For example, it will only interrupt the user when necessary.

In this section we give a short overview of Aura, starting with a motivating example. The remainder of the paper focuses on one aspect of Aura, the integration of physical and informational space.

2.1 Motivating Example

Bob is on a plane, working on a large project that involves multiple documents. When the captain announces that they are about to land, he logs out. Aura automatically saves all the documents that are part of the task, and transmits them to a server.

When Bob walks into his office, he uses a finger print reader to authenticate himself. Aura knows that he is alone in the office and restores the files that he was working, opening them at the right point, on the large display in his office. Aura also opens his mail, since Bob always reads mail first. While Bob was using a keyboard and mouse on the plane, Aura switches to voice control since it is more appropriate for the current task executed in the privacy of his office. Aura can synthesize speech to provide an audio alert to Bob of certain events or even for more complex operations such as reading back e-mail so Bob can have additional interaction with Aura while moving around in his personal space.

At this point, Bradley walks towards Bob's office. Aura recognizes Bradley since he is wearing an RF badge, but since badges do not provide strong authentication[1], Aura decides that the physical environment is no longer secure, and it iconifies Bob's e-mail client, thus hiding the confidential message that was being displayed.

Bradley drops by to work with Bob on a presentation that accompanies his project. Bob and Bradley continue to use voice control to navigate the PowerPoint presentation, but Bob uses the touch screen for modifications, while Bradley uses his PDA to annotate Bob's changes. When Bob and Bradley leave for lunch, Aura captures the state of the project and stores it on a server so it is ready to be restored wherever Bob decides to work on it next.

The current Aura prototype supports this scenario in a controlled environment. In the remainder of this paper we elaborate on the specific mechanisms used.

2.2 Overview of Aura

[1] RF badge identification is only effective if each individual is wearing an appropriate badge. We are extending Aura with video capabilities so we can detect people not wearing badges.

The motivating example shows some of the features of Aura. First, we want to support a rich set of I/O interfaces for users to interact with the computer system. This makes it possible to pick the most appropriate form of interaction for the job at hand (voice, keyboard/mouse, touchscreen, PDA). However, the change between modes of interaction should be seamless. Second, the system should be "context aware", i.e. it should know about the environment so it can automatically take appropriate actions, e.g. select the appropriate I/O device or hide confidential information. Third, the system has a notion of what the user is trying to achieve. In Aura this is explicitly represented in the form of a task, which is a first class object. Finally, Aura has a number of goals that are not explicit in the example and that are not central to this paper. For example, applications in Aura automatically adapt to resource availability, e.g. by using servers if the cycles are not available locally. Similarly, QoS support can be used to improve predictability in a resource-poor environment.

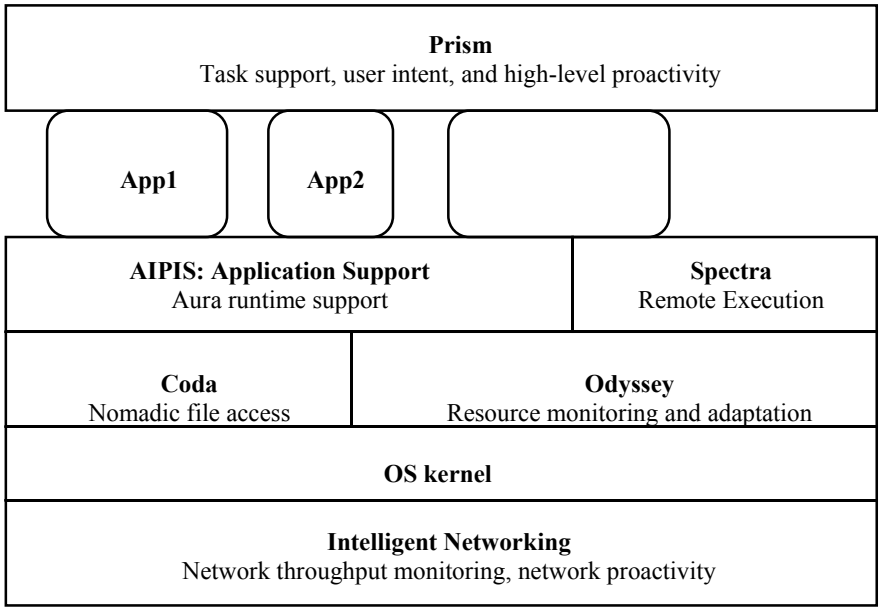

Figure 1. Aura's task-level architecture.

Figure 1 shows the Aura architecture. The bottom layers are concerned with identifying and managing the resources in the computing and physical environment. This includes support for networking, including network monitoring [20], and node management. At a slightly higher level, Aura also supports intelligent information access through Coda [18] and Odyssey [18], and adaptive remote execution capabilities through Spectra [21]. The top of the picture represents user tasks and preferences. The combination of low level system input and high-level user input allows the Aura system to make decisions about what to do (what operations could be of help to the user) and how to do it (what I/O devices to use, what CPUs to use, etc.) In this paper, we focus on the middle layer, i.e. the application support, which ties the physical

environment to the computing environment in the context of executing an everyday task, such as preparing a presentation.

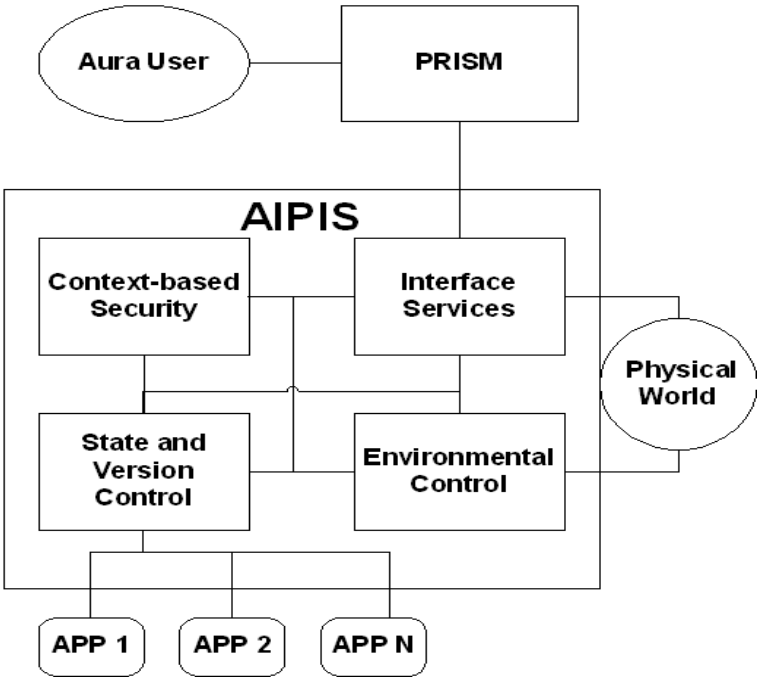

Figure 2: AIPIS: Integrating physical and information spaces.

The application support component interacts with the physical environment in several ways:

- It has to handle I/O from a variety of devices and channel that information to the right application or to the system itself. Specifically, our prototype supports voice control, touch screen, and PDA input, besides the usual keyboard and mouse input.
- It channels input from sensors in the environment to applications that are interested in the information. In our current system, examples include input from badges, finger print readers, motion detectors, etc.

In the remaining sections, we describe how we coordinate the flow information between the physical environment and the computing systems.

3 Application Services

Aura is concerned with the automation of everyday tasks for the mobile user. This concept is distinct from application and application-bundle automation services that

many vendors provide. For example, Microsoft provides common API's that allow the seamless transfer of information and control between application bundles, e.g. MS Office. What is missing from these is a set of rich automation services that provide run-time support to applications such that they can leverage information provided by a task-level support engine, e.g. user intent and preference within the current context, the proactivity needed to support mobile users, and QoS support during infrastructure transitions.

3.1 AIPIS Architecture

The value of providing applications with a context-based or task-based mechanism for application support is well understood [4, 11, 8]. In Aura, task level support is provided by Prism. Prism provides support for saving and restoring of a task in different environments, where a task is a set of documents that are needed as a group to meet a certain goal. An early version of Prism for the Windows environment is described in [4]. Aura's Architecture for the Integration of Physical and Information Spaces, AIPIS, extends these results by providing task-focused, context-based run-time support to Aura aware applications and application bundles. The four fundamental blocks of AIPIS, shown if Figure 2, coordinate between the Aura task layer, Prism, and the individual applications to provide the following:

Application Interface Services. This component supports inter-application data exchange and control protocols. It is also responsible for the human-machine interface component.
Environment Control Services. This component realizes Aura's bridge from the information domain to the physical environment, e.g. lighting, temperature control, access control, external displays, etc.
Context-based Security Services. This component provides an integrated control mechanism that provides context-aware access to physical, computational, and information-based resources.
State and Versioning Services. This component manages the allocation/de-allocation, transport, crash recovery, merging, and configuration of Aura aware applications.

The modules work in a loosely coupled, agent-based framework centered on the application interface, which monitors control input from the physical world, including user inputs, and the other AIPIS modules and provides this information to Aura's task-layer, Prism. An example of such information, provided by the security module, would be the privileges and identities of the set of known viewers of a presentation, which contains some classified data. The state and versioning module also provides to AIPIS the classification of each data set that is to be displayed in Aura, before it is displayed and also the current suite of information protection modes available within AIPIS, e.g. data hiding. The environmental module provides the set of devices, e.g. displays, physical locks, notifications, etc. that Aura can leverage to protect the data from the unauthorized viewers. The application interface aggregates this information and relays application, environmental state, context, and security information to

Prism. These control signals are reconciled by Prism with attention to minimizing impact on the current task and routed to AIPIS through the Application Interface and on to versioning, environment, and security modules such that the appropriate actions are executed such that the classified data is not exposed to unauthorized viewers. For example, Prism could relay to AIPIS that sensitive data must not be shown in its current form to a subset of users without proper privilege. AIPIS's security module could then chose instruct the application to remove the data from the presentation, the environmental manager could be notified to switch displays, or the state manager instructed to terminate the application depending on the high-level direction from Prism.

3.2 Application Interface & Environmental Control

Application interface & environmental service modules provides a uniform access to input, output, and external control functions for Aura applications. The associated capability that any application can include for each of the managers provides is:

Application Input Manager. Speech recognition, handwriting recognition, gestures, eye tracking, data mining, etc.

Application Output Manger. Speech synthesis, display management, environmental settings, mobile messaging

The input manager coordinates a set of data sources that can provide input to applications. Input data is typed and data sources of the same type are basically indistinguishable from the perspective of the application. For example, keyboard input, speech synthesis, and handwriting recognition all provide ASCII input. Similarly, gesture recognition or eye tracking will provide input of type "location". The application input manager controls what input source is used for an application. This decision can be based on many factors, e.g. user preferences, availability of specific devices in the environment, explicit user specification, etc. Note that users can use a specific data source for input to several applications. For example, speech input can be used control PowerPoint, a mail client, and the OS (e.g. application selection). It is up the input manager to keep track of what application is active. We elaborate on this in the next section when we discuss our prototype.

The output manager has a similar role as the input manager, but it controls the output mode for applications. For example, it controls what display to use for screen output.

One can think of the data sources and output devices as I/O services. The input and output managers are basically responsible for deciding dynamically which I/O services applications should currently use based on user level and context information. They can on the fly reconfigure the I/O services as desired.

3.2 Context-based Security & State and Versioning

The security issues in a pervasive computing environment are fundamentally different from those in a traditional distributed computing environment. The reason is that various ubiquitous I/O devices create new opportunities for privacy and security violations. For example, large touch screens provide a convenient way for users to interact with the system, but they make it easier for third parties to get unauthorized access to information. Similarly, speech synthesis can be a convenient output device, but it also similar privacy concerns.

Fortunately, devices can also help in addressing some of these new security and privacy risks. For example, fingerprint devices can provide convenient strong authentication while cameras, motion detectors, and badges can be used to identify the presence of other people. The challenge is to make this sensor information available to the applications that need it.

Security in Aura is concerned with access to resources, e.g. space, devices, applications, and data. The current instance of the security monitoring systems has three fundamental components that provide context-based security to Aura. They are:

Resource Monitor. Controls access to physical spaces, devices, and applications. Access to resources in AIPIS is usually provided by a challenge of the person requesting resources with a password or ID badge –based verification

Information Monitor. The information monitor provides the same level of protection for information that the resource monitor provides. The most advanced levels of security in Aura assume the lowest common denominator from the group that is to view Aura data and will employ data hiding or fuzzification to protect the release of unauthorized data to the group

Context-Monitor. AIPIS currently employees a simple context awareness that aggregates group permissions and then decides the optimal sub-set of resources that AIPIS is allowed to engage. In cases where security violations are present (sensitive data is exposed to unauthorized viewers), AIPS adapts resources and information to fit the current group security profile. The mechanisms employed are:

- *Display.* Transport and hiding
- *Input.* Mode suppression
- *Control*. Soft lockout of sensitive applications.

AIPIS uses an application state and transport mechanism that is described by Wang and Garlan in [4]. We describe how we use its capabilities in the next section.

4 Prototype Implementation: Aura Desktop

The Aura Desktop is a PC application that provides a hands-free gateway to many of the features and capabilities of an Aura environment, including AIPIS run-time support. In its current manifestation, the Aura Desktop provides voice-control of four common Microsoft applications, PowerPoint, Outlook, Word, and Excel. These applications support a wide range of documents. The user can also select from a suite of optional physical security devices including RF and IR tags, fingerprint readers, and context-based security services provided directly by AIPIS. Security options are configured during compilation. In addition, a context preserving mechanism is provided that allows users to maintain the state of applications across Windows machine boundaries, including PDAs, and infrastructure transitions thus enabling mobile users to work from any resource equipped with the Aura Desktop.

4.1 Application Interface & Environmental Control

Besides keyboard input and textual output on the screen, the Aura Desktop uses state-of-the-art speech recognition and synthesis to provide both application navigation capabilities and to provide feedback to the user in a hands free context. Speech recognition is accomplished using a native Sphinx II OCX [15] with a small dictionary. The top-level voice interface is designed to reliably provide the control of email and presentations at the expense of generality. This allows Bob to us voice control for working on this presentation in his office in the scenario of Section 2.1. Applications such as MS Word require continuous recognition with a much larger dictionary; they are currently not supported but they could be added easily to the AIPIS framework. Speech synthesis was implemented using a client/server implementation of the popular Festival speech synthesis system [16]. This is used to read back e-mail to Bob in the scenario of Section 2.1

The services provided to applications that leverage our voice-control interface are:

PowerPoint Services. The Aura Desktop implements speech-based presentation navigation, editing, and creation features such that drudgery of a large number of manual tasks associated with these capabilities is eliminated.

Document Archive. The Aura Desktop can be configured such that user documents (presentations, images, word and excel files, etc.) are stored in any network accessible file folder.

Outlook Services. The Aura Desktop provides the user with a voice-controlled interface to email as well. It also provides the ability to synthesize the email text and to recite each email to the user. In speech control mode, the user is freed to perform other tasks while sorting emails. In addition to recitation, a baseline set of sorting features is implemented.

Excel & Word. The Aura Desktop has minimal support in the form of (1) transport, (2) instantiation, (3) document hiding for security purposes, and (4) termination.

Digitized speech is provided asynchronously to the AIPIS application interface module from Sphinx. This speech is then parsed for clues relating to the application and the associated action(s) the user is trying to effect. Once application-context is decoded speech strings are sent to an application specific analysis routine that extracts any necessary data and action contexts from the string or inserts any implied data into the action sequence. For example, the integrated AIPIS application interface module may receive the command string, "Open the Aura Presentation" from the Sphinx OCX. AIPIS parses the string for context: (1) the presence of the keyword "presentation" implies PowerPoint is the application to use on a Windows system, and (2) from the keyword open in conjunction with the application context, AIPIS searches the set of known presentation directories for presentations that contain references to "Aura". AIPIS then selects the best match and engages the state and versioning module to transport the application to the appropriate combination of location, processor, and display.

For graphical input and mouse control, the Aura desktop also supports PowerPoint Commander and Pebbles interfaces [13, 14], besides the usual mouse and keyboard control. This makes it possible to support basic navigation and annotation of PowerPoint from a PDA. This mode of interaction is sometimes more appropriate than the keyboard and mouse mode, for example, when mobility is important such as during a presentation. In our scenario of Section 2.1, this allows Bob and Bradley to work together on a presentation using (besides voice) three different modes of interaction: keyboard and mouse, touch screen capabilities, and a handheld PDA. They can pick their most appropriate mode of interaction based on what operation they plan to perform and where they are physically present.

AIPIS makes extensive use of the Microsoft COM/D-COM interfaces to implement application control within the MS Office suite. Speech recognition and synthesis modules are controlled through a sockets-based interface using client/server architecture. In both cases, the implementation of the communication between applications in AIPIS resides in C++ wrapper classes such that details are hidden from the main AIPIS control loop.

4.2 Context-based security

AIPIS employs an integrated security mechanism that leverages passwords, RF-based authentication [17], motion detection, and finger print readers. The security module uses a two-tiered system for access where maximum access is provided by reliable security mechanisms such as passwords or fingerprint ID readers. The RF badge is treated strictly as a weak form of access that allows applications with sensitive information to be suppressed when a weak user is identified. For example, an accountant is reviewing a spreadsheet with current and recommended salaries for university staff.

The Aura Desktop can be configured to automatically suppress (un-map it from the display) when an unauthorized employee enters the viewing space. This was also demonstrated in our scenario in Section 2.1. Since Bob used a fingerprint reader for authentication, so he had unlimited access to all information and applications. However, since Bradley was only identified using a badge (weak security), confidential information was hidden. Of course, once Bob recognizes Bradley, he can use the touch screen or voice control to display any information that he thinks is appropriate.

Security in AIPIS uses an asynchronous, event-driven scheme coupled with a security state matrix that encodes all the relevant relationships between user and application. For example, a user with weak access might be granted navigation privileges in PowerPoint, but would be blocked out from Outlook until he/she provides password or fingerprint ID. Each new event is decoded and the state matrix is updated. The security state vector for any affected application is then reconciled by taking an action that alleviates the violation, e.g. the email application is terminated and the unauthorized user is blocked from trying to access it.

A Security Warning Indicator (SWI), see Figure 3, is provided that warns the user when an unauthorized person has entered a private workspace. The SWI has hooks for the RF badge reader as well as external devices that the user might integrated, e.g. a video camera running motion detection, for example. Also, a text list of current users is displayed in the User Access Panel. In this panel, three levels of access are represented:

*Maximum***.** The specified user has been authenticated through a physical device such as password or fingerprint ID. Full access to all Aura Desktop features is provided.

Weak. The specified user has been identified with an external authentication device such as RF/IR tags or video recognition. Access to a navigation features is provided.

Unauthorized. An unidentified or identified but unauthorized individual has entered a private workspace. No access to the Aura Desktop features is provided to that individual.

4.3 State and Versioning Support

Aura's current state services center on the task transport necessary to support a mobile user such that application state, including preferences and data, track the physical movement of the user through an Aura enabled space [4]. Users can define a task as a set of documents that belong together. Aura keeps track of the task state, including not only the contents of the documents, but also information such as cursor position and window layout. Aura can then restore the task on any Aura-capably Windows platform, using one of four applications (MS Word, Powerpoint, Excel, and Outlook) to handle the documents. Even the current prototype is restricted to Windows, this is already a powerful capability. For example in our scenario, Bob's tasks are automati-

cally moved from his laptop to his office desktop system. Aura's state manager provides the seamless transition necessary, as application transport is trigger by movement in or out of Aura cells or user request. In the default case, Aura will try to maintain the default status of the user's computing environment as closely as it can across platform boundaries providing ubiquitous access to applications for the mobile user.

Work is in progress in extending task support so it works across heterogeneous platforms, e.g. Windows and Linux. This is a lot more challenging since often different applications may have to be used and document translation may be necessary.

4.4 GUI and Configuration

Figure 3 shows a screen shot of the Aura Desktop management GUI. It provides a fairly minimal interface for managing speech synthesis and security and it is designed primarily to support experimentation and evaluation.

Users can customize a remote Festival server with a favorite voice model by setting the IP address and port name in the "Festival Speech Synthesis Server IP Address" box (lower right corner of Figure 3). The Aura Desktop will then send all future synthesis requests to this preferred service provider.

Users can select the RF badge service by specifying its IP address in the lower left corner. The top of the window shows the security of the user's current physical context. The window in the middle lists all nearby users and their status. The rectangle in the top right hand corner turns red if any unauthenticated users are in the vicinity and is green otherwise.

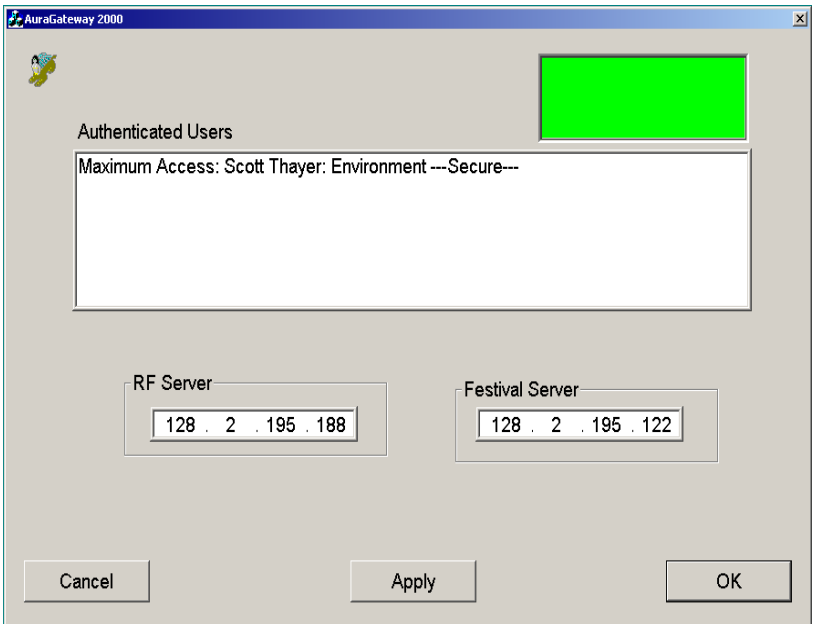

Figure 3. Screen shot of an instance of Aura Desktop GUI

5 Related Work

Besides Aura, a number of other projects are exploring pervasive computing. MIT's Project Oxygen aims to "communicate with people using natural resources, such as speech and vision", to support users on the move, and to monitor and control the environment [6,7]. Georgia Tech's InfoSphere endeavors to "achieve nothing less than radically enhancing human understanding through the use of information technology, by making it dramatically more convenient for people to interact with information, devices, and other people." They forward the idea of universal *Information Utility* and explore the architecture in areas of "rapid decision making" and learning. They define success in terms of the effectiveness of the "InfoSphere" to amplify human intellect [9]. University of Washington's Portolano project "seeks to create a test bed for investigation into the emerging field of invisible computing [10]. Aura is distinct from these efforts in that it seeks to service the needs of a mobile users engaged in everyday tasks as they move through infrastructures of varying quality.

This paper describes the infrastructure between Aura's task layer and the user applications that are registered within Aura. This infrastructure, AIPIS, provides context-based security, versioning, interface, and environmental services sufficient to coordi-

nate the application base such the mobile user can continue to engage the world with relative independence from infrastructure.

6 Conclusions

We have presented an initial architecture and implementation of a system for hands-free computing that transforms the focus of computing architectures from application-based to task-based implementations. This is a critical first step in harnessing the tremendous potential of a truly human-centered approach to computing. In additional to a task bias, our architecture works to seamlessly integrate the physical-reality of humans and the information spaces that encode much of our identity, knowledge, and resources. Only with an efficient coordination of both physical and information domains can the full potential of computing be realized. We, as users, must be allowed to move throughout both the physical and information worlds with minimal constraints and burdens. Further, we require a personal infrastructure that is capable of taking maximal advantage of the modalities presented to us at each point in the physical-informational continuum such that we are empowered to engage tasks independent of location, bandwidth, and the chains of deskbound computing.

This Aura application architecture is a first step in the realization of ubiquitous, hands free, and mobile computing tailored to user needs. In this first implementation, both personal and desktop computing devices have been coordinated to automate a simple task level interface for the creation and delivery of presentation materials. This infrastructure utilizes speech-recognition and synthesis; MS PowerPoint, Word, and Excel; portable computing; personal location and identification services; and application transport software to generate the capability where a user working at a desktop computer can create and edit a presentation from a voice-driven interface. The user can then utilize the Aura infrastructure to tweak presentation material en route to the speaking engagement with either his personal computing resources or salient and Aura aware resources embedded in the environment. Upon arrival at the engagement, Aura will install an audience appropriate version of the presentation material and allow the user to navigate using a natural voice or PDA interface.

References

1. Small, J., Smailagic, A., Siewiorek, D.,"Determining User Location For Context Aware Computing Through the Use of a Wireless LAN Infrastructure", December 2000
2. Narayanan, D., Flinn, J., Satyanarayanan, M., "Using History to Improve Mobile Application Adaptation", Proceedings of the Third Workshop on Mobile Computing Systems and Applications, Monterey, CA, December 2000
3. Satyanarayanan, M., "Caching Trust Rather Than Content", Operating Systems Review, Volume 34, No. 4, October 2000
4. Wang, Z, Garlan, D. "Task-Driven Computing" Technical Report, CMU-CS-00-154, School of Computer Science, Carnegie Mellon University, May 2000

5. http://www.cs.cmu.edu/~aura/
6. http://oxygen.lcs.mit.edu
7. Michael L. Dertouzos, "The future of computing", *Scientific American*, July 1999.
8. Anind Dey and Gregory Abowd, "The Context Toolkit: Aiding the Development of Context-Aware Applications", IEEE 2000 International Conference on Software Engineering, Limerick, Ireland, June 6, 2000.
9. http://endeavour.cs.berkeley.edu/
10. Esler, M., Hightower, J., Anderson, T., and Borriello, G. "Next Century Challenges: Data-Centric Networking for Invisible Computing: The Portolano Project at the University of Washington", Mobicom 99.
11. Dey, A.K., et. al. "The Conference Assistant: Combining context-awareness with wearable computing". Proceedings of the 3rd International Symposium on Wearable Computers (ISWC '99, pp. 21-28.
12. Jim Waldo, "Mobile Code, Distributed Computing, and Agents", IEEE Intelligent Systems, pg. 10-12, Jan 2001
13. Brad A. Myers. "Using Multiple Devices Simultaneously for Display and Control." *IEEE Personal Communications*, Special issue on "Smart Spaces and Environments." vol. 7, no. 5, Oct. 2000. pp. 62-65.
14. Brad A. Myers. "User Interface Software Tools," *ACM Transactions on Computer-Human Interaction*. vol. 2, no. 1, March, 1995. pp. 64-103.
15. http://fife.speech.cs.cmu.edu/speech/sphinx/
16. http://www.cstr.ed.ac.uk/projects/festival/
17. http://www.versustech.com/
18. Braam, P. J.: The Coda Distributed File System, Linux Journal, Vol. 50, May 1999
19. Noble, Brain, System Support for Mobile, Adaptive Applications, IEEE Personal Communications, Vol. 7, No. 1, Feb. 2000
20. A. DeWitt, T. Gross, B. Lowekamp, N. Miller, P. Steenkiste, J. Subhlok, and D. Sutherland: *"ReMoS: A Resource Monitoring System for Network-Aware Applications"* Carnegie Mellon School of Computer Science, CMU-CS-97-194.
21. Jason Flinn, Dushyanth, Narayanan, and M. Satyanarayanan, *"Self-Tuned Remote Execution for Pervasive Computing"*, Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII), Schloss Elmau, Germany, May 2001.