Project Aura: Toward Distraction-Free Pervasive Computing

The most precious resource in a computer system is no longer its processor, memory, disk, or network, but rather human attention. Aura aims to minimize distractions on a user's attention, creating an environment that adapts to the user's context and needs.

David Garlan, Daniel P. Siewiorek, Asim Smailagic, and Peter Steenkiste
Carnegie Mellon University

As the effects of Moore's law cause computing systems to become cheaper and more plentiful, a new problem arises: increasingly, the bottleneck in computing is not its disk capacity, processor speed, or communication bandwidth, but rather the limited resource of human attention. Human attention refers to a user's ability to attend to his or her primary tasks, ignoring system-generated distractions such as poor performance and failures. By exploiting plentiful computing resources to reduce user distraction, Project Aura is creating a system whose effectiveness is considerably greater than that of other systems today.

Aura is specifically intended for pervasive computing environments involving wireless communication, wearable or handheld computers, and smart spaces. Human attention is an especially scarce resource in such environments, because the user is often preoccupied with walking, driving, or other real-world interactions. In addition, mobile computing poses difficult challenges such as intermittent and variable-bandwidth connectivity, concern for battery life, and the client resource constraints that weight and size considerations impose.

To accomplish its ambitious goals, research in Aura spans every system level: from the hardware, through the operating system, to applications and end users. Underlying this diversity of concerns, Aura applies two broad concepts. First, it uses *proactivity*, which is a system layer's ability to anticipate requests from a higher layer. In today's systems, each layer merely reacts to the layer above it. Second, Aura is *self-tuning*: layers adapt by observing the demands made on them and adjusting their performance and resource usage characteristics accordingly. Currently, system-layer behavior is relatively static. Both of these techniques will help lower demand for human attention.

Envisioned scenarios

To illustrate the kind of world we are trying to create, we present two hypothetical Aura scenarios. Although these might seem far-fetched today, they represent the kind of scenarios we expect to make commonplace through our research.

In the first scenario, Jane is at Gate 23 in the Pittsburgh airport, waiting for her connecting flight. She has edited many large documents and would like to use her wireless connection to email them. Unfortunately, bandwidth is miserable because many passengers at Gates 22 and 23 are surfing the Web. Aura observes that, at the current bandwidth, Jane won't be able to finish sending her documents before her flight departs. Consulting the airport's wireless network bandwidth service and flight schedule service, Aura discovers that wireless bandwidth is excellent at Gate 15, and that there are no departing or arriving flights at nearby gates for half an hour. A dialog box pops up on Jane's screen suggesting that she go to Gate 15, which is only three minutes away. It also asks her to prioritize her email, so that the most crit-

1536-1268/02/$17.00 © 2002 IEEE

ical ones are transmitted first. Jane accepts Aura's advice and walks to Gate 15. She watches the election returns on the TV there until Aura informs her that it is close to being done with her messages, so she can start walking back. The last message is transmitted during her walk, and she is back at Gate 23 in time for her boarding call.

In the second scenario, Fred is in his office, frantically preparing for a meeting at which he will give a presentation and a software demonstration. The meeting room is a 10-minute walk across campus. It is time to leave, but Fred is not quite ready. He grabs his PalmXXII wireless handheld computer and walks out of the door. Aura transfers his work from his desktop to his handheld, and lets him make his final edits using voice commands during his walk. Aura infers where Fred is going from his calendar and the campus location tracking service. It downloads the presentation and the demonstration software to the projection computer and warms up the projector. Fred finishes his edits just before he enters the meeting room. As he walks in, Aura transfers his final changes to the projection computer. As the presentation proceeds, Fred is about to display a slide with highly sensitive budget information. Aura senses that this might be a mistake: the room's face detection and recognition capability indicates that there are some unfamiliar faces present. It therefore warns Fred. Realizing that Aura is right, Fred skips that particular slide. He moves on to other topics and ends on a high note, leaving the audience impressed by his polished presentation.

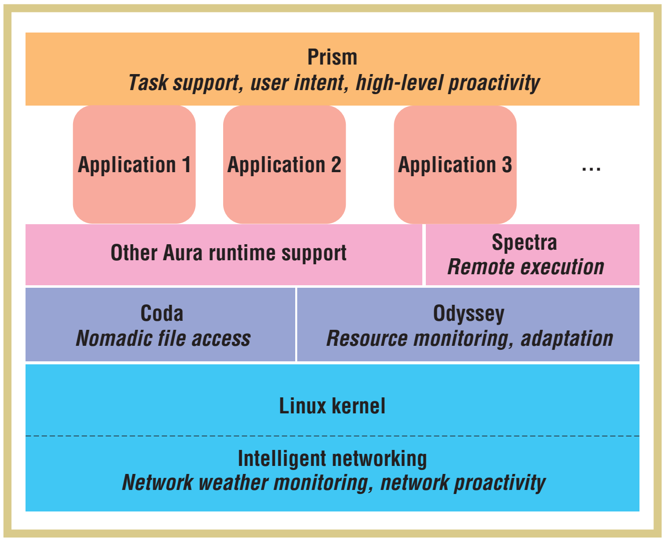
These scenarios embody key pervasive computing ideas.[1] In scenario 1, Aura is proactive in estimating how long it will take Jane to send her documents. In addition, it combines knowledge from different system layers, determining both wireless congestion (a low-level system consideration) and the flight's boarding time (an application or user-level concept) to let Jane complete her email transmission. Furthermore, Aura takes advantage of smart spaces, using the services its environment provides to determine wireless conditions at other gates, arrival and departure times and gates, and distances between gates.

Scenario 2 shows how we can easily transfer the execution state across diverse platforms—for example, from a desktop to a handheld and from the handheld to projection computer. It exemplifies self-tuning through Fred's ability to edit the handheld using speech input rather than a keyboard and mouse. Aura shows signs of proactivity when it infers that Fred is headed for the room across campus, warms up the projector, transfers the presentation and demonstration, anticipates the upcoming budget slide, and senses danger by combining this knowledge with the inferred presence of strangers. Aura uses smart spaces when it consults the location tracking and online calendar services to infer Fred's destination; warms up the software-controlled projector; and uses the camera-equipped room with continuous face recognition to warn Fred that he is about to present sensitive information.

Project Aura

The component technologies presented in these scenarios are not complex. The hardware technologies (such as the laptops, handhelds, wireless communication, software-controlled appliances, and room cameras) are readily available, as are many of the component software technologies—location tracking, face recognition, speech recognition, and online calendars. Unfortunately, the whole is much greater than the sum of parts. Research is needed not just on the building blocks of pervasive computing but in their seamless integration.

In Project Aura, which started about two years ago, we are developing the system architecture, algorithms, interfaces, and evaluation techniques needed to realize the Aura vision. Figure 1 shows the architecture we've adopted, including an Aura client's components and their logical relationships. The text in italics indicates each component's role. Coda and Odyssey were created prior to Aura but are being modified substantially to meet pervasive computing demands. Odyssey supports resource monitoring and application-aware adaptation,[2] and Coda provides support for nomadic, disconnectable, and bandwidth-adaptive file access.[3] Spectra is an adaptive remote execution mechanism that uses context to decide how to best execute the remote call. Prism, discussed later, is a new system layer that is responsible for capturing and managing user intent. It is layered above applications and provides high-level support for proactivity and self-tuning.


Prism
Task support, user intent, high-level proactivity
Application 1
Application 2
Application 3
...
Other Aura runtime support
Spectra
Remote execution
Coda
Nomadic file access
Odyssey
Resource monitoring, adaptation
Linux kernel
Intelligent networking
Network weather monitoring, network proactivity
Figure 1. The Aura architecture.

Cyber foraging

Aura uses *cyber foraging*[1] to amplify the capabilities of a resource-limited mobile client and thus improve user experiences. Compute servers or data-staging servers located near the client provide this amplification. We call such a server a *surrogate* of the Aura client it is temporarily assisting. We expect a surrogate to have good connectivity to the Internet and wireless LAN connectivity to Aura clients nearby. We have started using surrogates both as compute servers[4] and as data-staging servers. For brevity, we discuss only data staging here.

Staging data on surrogates can reduce the impact of end-to-end Internet latency on interactive file-intensive applications. This impact is easily seen in situations such as reading mail using mail clients such as exmh

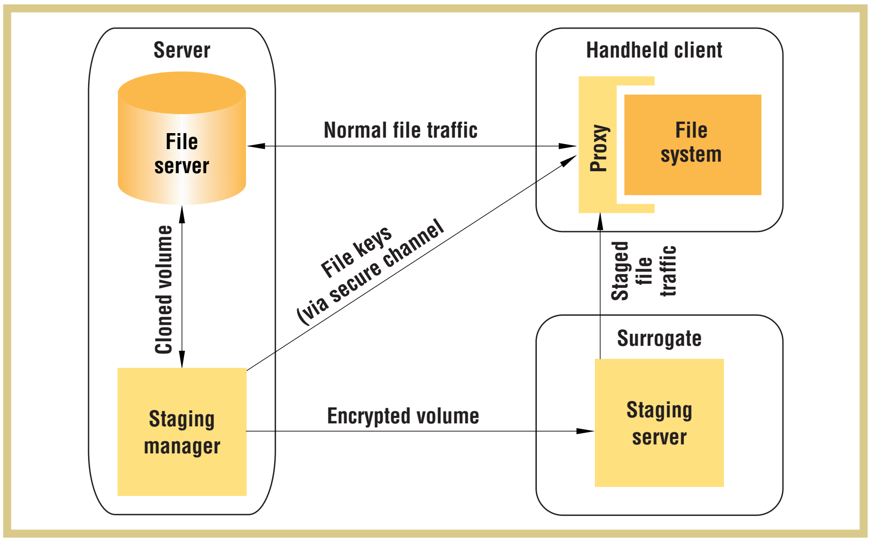


Figure 2. The data-staging architecture.

in Linux or browsing a remote directory tree with Explorer in Windows. In such applications, a flurry of cache misses on relatively small files can result in annoying delays and sluggish behavior. Merely improving bandwidth does not help because such interactive applications are latency-limited rather than bandwidth-limited.

Although aggressive use of caching can mask latency, there are some important circumstances in which cache misses are unavoidable. First, a resource-poor mobile client might not have a cache large enough to fit all relevant data. Second, the client might experience periods of disconnection during which updates of interest to the user might have occurred. Upon reconnection, there will be cache misses as the client accesses this data. Third, some uncached files might unexpectedly become relevant to a user. For example, the user might receive a cell phone call requiring that he or she access files previously considered unimportant. Because cache misses are unavoidable, our approach is to reduce their performance impact through data staging.

In Aura, the Coda file system provides nomadic file access,[3] and we recently extended Coda to exploit data staging. Data staging applies the well-understood concept of prefetching[5,6] to pervasive computing environments. Our current implementation stages data in relatively coarse-grained *snapshots* of file system data. Each snapshot corresponds to a volume, which is a predefined partial subtree of the file system name. This name space typically contains a group of related files. Each snapshot is a consistent read-only view of the file server state at some point in time.

Figure 2 shows how our data-staging architecture is split across three computers: the server, surrogate, and handheld. The orange components represent the preexisting distributed file system. The yellow components show our modifications to support data staging. The staging server on the surrogate is an unmodified Apache HTTP/1.1 Web server. The proxy intercepts and redirects file system traffic. If a request is for data contained on the surrogate, the proxy directs the request to it. Otherwise, it forwards the request to the distant file server. The proxy also performs translation between the file system protocol and HTTP.

The staging manager on the distant Coda server oversees snapshot creation. It contacts the file server to create a snapshot, then encrypts and transmits it to the staging server. The backup creation, encryption, and network transmission are pipelined to reduce latency. We now manually perform surrogate discovery and snapshot creation, but we'll automate these as Aura development progresses.

Our staging architecture avoids the need to trust surrogates, using an approached characterized as "caching trust rather than content."[7] In other words, we require clients to cache sufficient information to validate the contents of a file even if they cannot cache the entire file. Our implementation uses MD5 checksums for validation. The proxy uses these checksums to verify the integrity of files that the surrogates give it.

To ensure privacy, we use end-to-end encryption of snapshots. Our current implementation uses the Data Encryption Standard by default; it also supports triple-DES and other private key encryption schemes. The staging manager can choose a per-file or per-volume encryption key when creating a snapshot. Key distribution is simpler with per-volume encryption, but per-file keys offer greater resilience to compromise. Key distribution is done through direct client-server communication, without any involvement of a surrogate. MD5 checksums are included in the key distribution process. Typically, a user will cache keys for volumes of interest before he or she departs on a trip. If unforeseen data accesses arise on the trip, the user can obtain keys via a secure client-server channel such as *ssh* (secure shell).

Preliminary performance results confirm the benefits of data staging. The experimental setup consists of a laptop connected to a surrogate using an 11-Mbits-per-second wireless local area network. The connection between the surrogate and the remote server is a 100 Mbps Ethernet on which a delay of 30 ms was emulated using the NISTnet network emulator. For a first benchmark, we modeled a user searching an email archive of 250 messages using 1.67 Mbytes. Data staging reduced the cumulative delay that the user must wait before messages are displayed from 59 seconds to 32 seconds. A second benchmark modeled a user browsing a large video database with 345 images (354 Mbytes of data). Data staging reduced the time to load each image in succession using djpeg (an application that decompresses jpeg images) from 70 to 44 minutes.

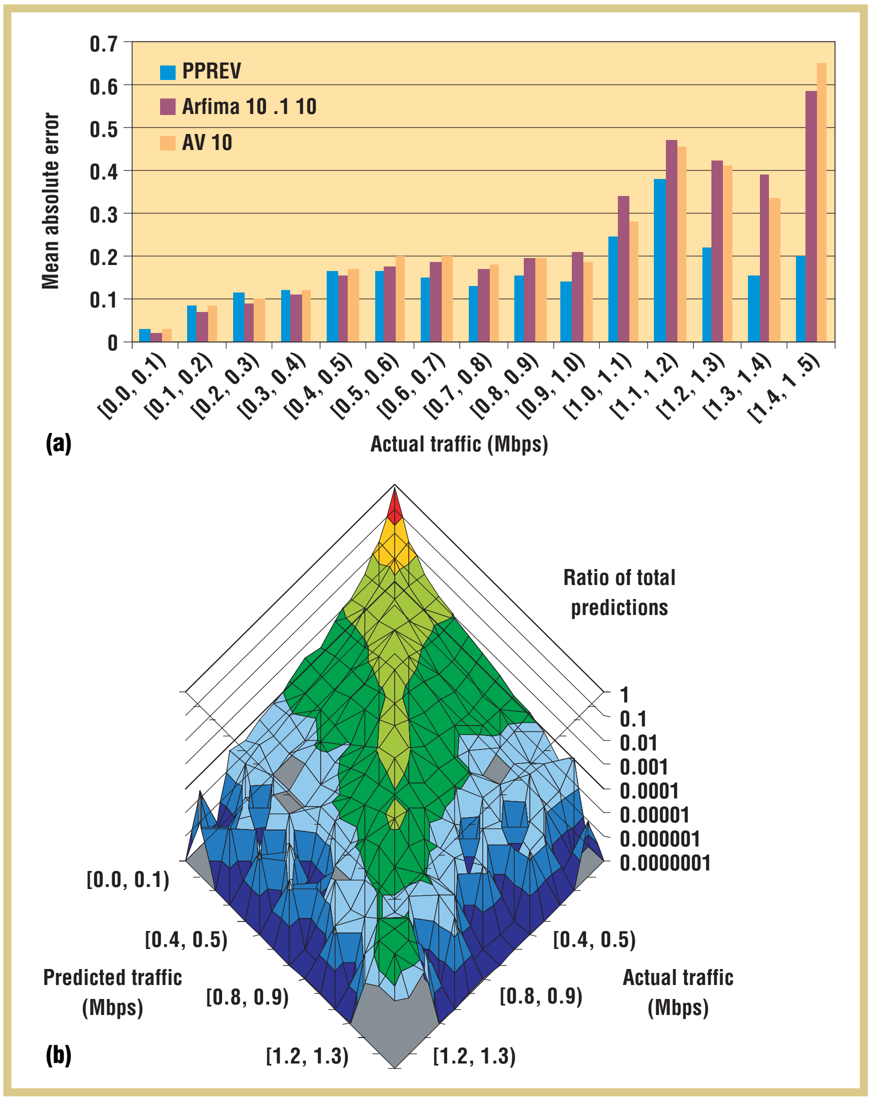


Figure 3. The accuracy of network utilization prediction: (a) comparative performance and (b) accuracy of the PPREV model.

A wireless bandwidth advisor

Network-aware applications can use reasonable estimates of future available bandwidth to make informed decisions, such as server selection. To support such applications, the research community has developed several bandwidth advisor services[8–10] that provide information on network conditions. Users of such services include end-user applications (such as Jane's mail client), file systems,[3] or middleware services.[2]

In the wireless realm, where bandwidth is often scarce, the opportunities for leveraging estimates of future available bandwidth are greatly increased. At the same time, there are some unique challenges associated with developing a wireless bandwidth advisor. First, application throughput in a wireless network is affected not only by the nature of the competing traffic (as in wired networks) but also by effects such as noise and cross-cell interference. Furthermore, in a wireless network, network performance depends on the device's physical location.

The components of our bandwidth advisor (for IEEE 802.11 wireless networks) are divided into two categories: monitoring and prediction. Two monitoring components periodically gather information from wireless access points using SNMP (simple network management protocol). The AP Segment Service collects incoming and outgoing traffic rates and related information, such as error and collision rates, while the AP Device Service obtains cell population information by querying each access point's bridge table. Recent information is held in memory to provide data needed to satisfy client requests. In addition, all data the monitoring components gather is stored to disk. Prediction components can then operate on this data online or offline.

As a first step toward providing an accurate application throughput prediction, we need to know how heavily the cell will be used. We have investigated using simple linear models to predict future use from recently observed past values. Using the AP Segment Service, we gathered data from several access points in Carnegie Mellon University's School of Computer Science as well as CMU's business school over a period of several weeks during April and May 2000. During this time, CMU's wireless network operated at 2 Mbps. We took samples in these traces every 10 seconds.

Inspecting the data confirmed many of our intuitive suspicions: many cells were almost entirely idle while others were heavily used. Moreover, utilization tended to change at hourly intervals and can be correlated with the same interval in other weeks and with the number of stations in a cell. Given that classes typically run for 60 and 90 minutes and start at regular times, this was not a surprise.

We examined the accuracy of the following three models at predicting one sample ahead in our traces (details on these predictive models are presented elsewhere[11]):

- *PPREV* predicts future values to be the same as the most recent value observed.
- *AV* (average value) uses an evenly weighted average of the several previous observations (10 for the results presented here).
- *Arfima* (auto-regressive fractionally integrated moving average) computes future values as a dynamically weighted average of past values (again, we show results for 10 previous samples). Unlike AV, this model applies a different weight to each past sample and updates these weights as it runs to better fit currently observed error.

Figure 3a shows a typical set of results that compare the accuracy obtained with these three predictive models; our metric is the mean absolute error for predictions. The mean absolute error is shown as a

TABLE 1
Accuracy of location measurements.

Confidence (%)	Strength (dBm)	Range (ft)
68.6	+/– 0.939	+/– 5
95.4	+/– 1.146	+/– 10
99.9	+/– 2.817	+/– 15

function of the ranges of actual traffic observed. For example, the first set of bars shows the mean error for each predictor when little traffic was present. The results clearly show that AV is inferior to the other methods. For actual traffic values above 0.5 Mbps, PPREV is the clear winner, while Arfima does better for traffic values less than this.

We can easily explain this result. In cells that are idle or have low utilization, the traffic is usually just background traffic—for example, ARP (address resolution protocol) and periodic management traffic. Arfima is good at picking up patterns in traffic, so it works well in low-utilization cells. However, in high-utilization cells, traffic is very bursty, so PPREV works better because it adapts quickly to changes in traffic conditions. Figure 3b uses a 2D histogram to show the PPREV model's performance. Many of the data points lie on the diagonal (predicted = actual), indicating that the model works well for this data set.

The results presented here are for predictions on a short time scale (10 seconds). The wireless bandwidth advisor also collects historical data. For predictions further into the future (such as deciding where you can get the best bandwidth in 15 minutes), we are evaluating whether extrapolation from historical data will yield more accurate results.

The service described is just a first step. We are in the process of more carefully characterizing the bandwidth advisor's accuracy. One aspect of this effort is to determine how we can best combine near-term utilization measurements with longer-term historical data. Another issue is that the wireless network has been upgraded to 11 Mbps. This complicates network utilization prediction because hosts can now send at different rates, depending on noise conditions. We also want to report estimated application throughput, based on utilization estimates. This is a difficult problem, especially in a wireless network where hosts can use different transmission rates and can optionally use power management.

The WaveLAN-based people locator

Location information is a key parameter of context awareness. Our implementation of a people location service is based on signal strength and access point information from the IEEE 802.11 WaveLAN wireless network covering the CMU campus. Its availability was a significant factor in determining the underlying technology for the location service. Other systems based on radio frequency, ultrasound, and video require an investment in infrastructure and hardware. GPS-based systems tend to have poor indoor coverage and require each client to be equipped with a GPS unit, which adds weight and consumes power. Our goal was to develop a cheap, scalable, and easy-to-use location service that a variety of clients, ranging from wearable computers to laptops, could use.

We have developed two algorithms for location sensing: CMU's pattern-matching algorithm (CMU-PM) and the triangulation-based remapped interpolated algorithm (CMU-TMI).

CMU-PM determines location by measuring the signal strength from a computer to all available wireless access points. It compares these measurements to a table containing a unique reading of signal strengths for each location. To train the system, the user enters his or her location into the computer. The algorithm takes and averages signal strength samples and stores them in a table that it can reuse across sessions. During use, the algorithm compares measure values to those in the table and computes differences. It assumes the entry with the smallest difference is the current position (see Figure 4).[12] The client requests the location of a target from a server. It may use a caching mechanism or send the request to the target user. The target's computer determines its location and sends the results to the server, which are then passed to the client.

The Radar algorithm gathers signal strength data from access points,[13,14] while

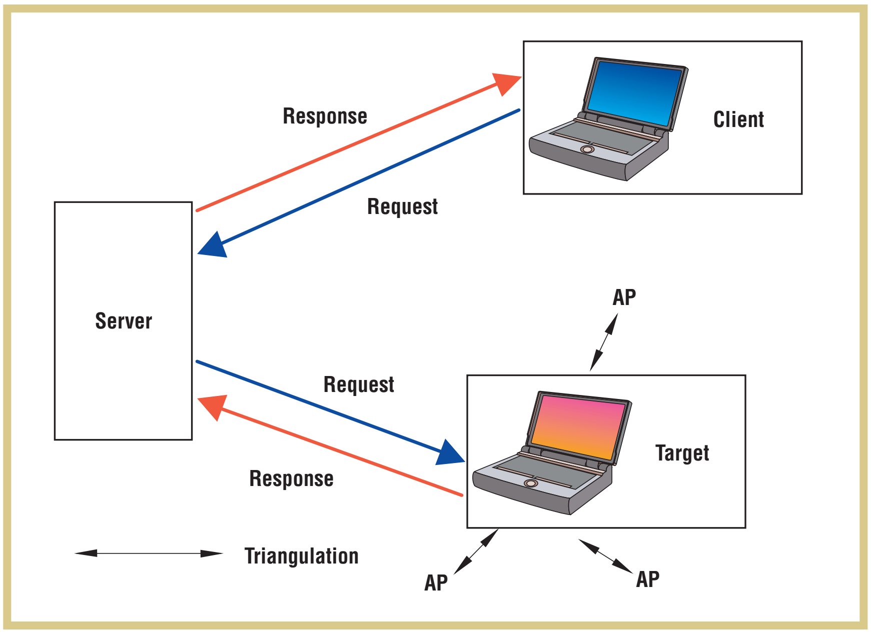
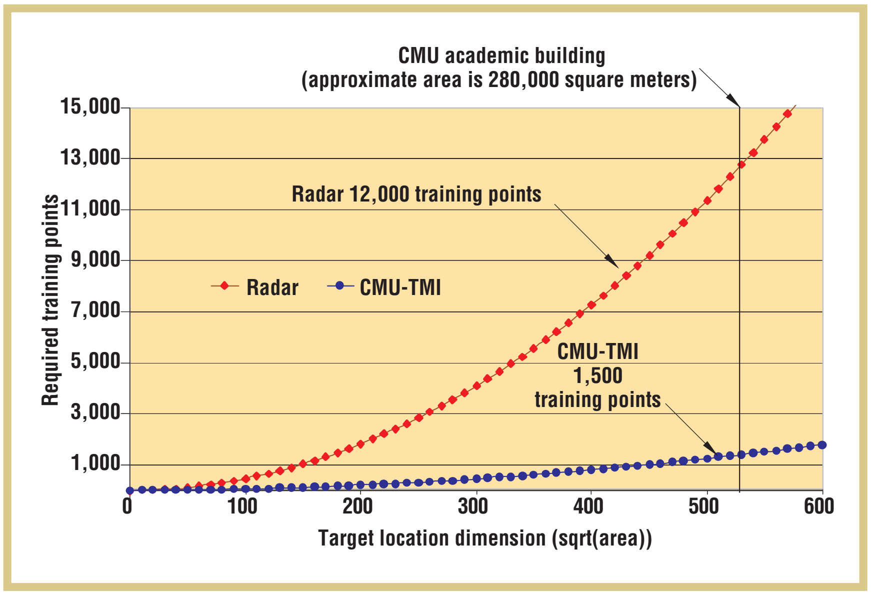


Figure 4. Client-side location service architecture (AP stands for access point).

Figure 5. The training complexity of algorithms.



CMU-PM has clients gather the data. The advantage of these algorithms is that, given a sufficiently dense table of patterns, we can often pinpoint a user's position to within five feet (see Table 1), while the accuracy is almost always 15 feet or better. We initially trained the system with an equal distribution of signal strength measurements taken with a laptop at waist height. Recording the signal strengths from a stationary location over several hours showed signal strength with a standard deviation of 2.13 dBm. Opening a door produced a change of 10 dBm. Other handheld devices present in the test area caused an effect of up to 5 dBm. We conducted most experiments where there was line-of-sight—for example, along the length of hallways in university buildings. We also included places in a 2D plane and extended them to 3D. Signal strength readings taken during the day were nearly identical to readings taken at night. The presence of people did not appear to affect the measurements. However, this algorithm, as well as Radar, requires many training points.

The high training overhead motivated us to design a new algorithm, CMU-TMI. When designing CMU-TMI we had two specific goals: accuracy within a few feet, and scalability, such that every returned value does not require a trained data point. This approach required interpolation between trained values.

CMU-TMI performs two transformations on the raw data of signal strengths. It calculates the client's position on a continuous coordinate grid, assuming that signal strengths map directly to distance (triangulation in Figure 4). It then maps the resulting coordinates onto real space coordinates using a set of trained values. Because both transformations are continuous, the result is interpolative on the trained data and is significantly more accurate than the area divided by the number of trained values.

CMU-TMI requires less complicated training than Radar, as Figure 5 shows. We need to know the physical position of all access points, and to generate the signal space positions, we must generate a function that maps signal strength to distance. We calculated this empirically from observations. Finally, to map the signal space positions onto physical space positions, we must generate a set of trained points. These are offset vectors from signal space positions onto physical space positions, and we calculate them by performing the algorithm at a known location and then recording that location.

Accuracy for the CMU-TMI falls between that of the CMU-PM and Radar algorithms for low error distances. However, it generates better results when errors are greater than four feet. The greater accuracy at high distances is a direct result of the nature of the algorithms. Errors in CMU-PM or Radar are often large, because they are typically the result of an incorrect training point being returned. In contrast CMU-TMI's errors are continuous, so although small errors are common, it is unlikely that the returned position will be far from the user's actual location.

Although the results are usually accurate, we have observed that certain environmental elements can affect location system's output. Using a Lucent WaveLAN 802.11 card installed on the side of a laptop, rotating the laptop would move the reported physical location more than the 6-inch radius of the rotation. Placing a hand over the antenna has shown a reduction of signal strengths reported from access points. We've tested the system with a minimum of six additional wireless units operating on the same channel, within close proximity to the location client. We conducted these measurements in a building with multiple offices, laboratories, and people.

Location sensing is useful for mobile users, who often have significant power constraints. We designed our algorithms to consume minimal power. We evaluated the battery life for a Jornada 680 handheld computer while running the location sensing algorithms. These algorithms force the wireless network card on the device to scan nearby access points, depleting the battery. For our experiment, the device was fully charged, and it ran until fully discharged. It performed location calculations once every 10 seconds. CMU-PM decreased battery life by 8 percent; CMU-TMI decreased it by 6 percent.

We need to perform more experiments to quantify the impact of various environmental factors on the people locator. We are also experimenting with the scalability of this approach, increasing the number of users by an order of magnitude to over 100 users, including several academic classes. We hope to further increase its efficiency and scalability by incorporating caching and location prediction.

Capturing user intent

Two of Aura's most important capabilities are supporting user mobility and shielding users from variations in resource availability. When a user moves from one environment to another, Aura attempts to reconfigure the new environment so that the user can continue working on tasks started elsewhere. As resources in an envi-

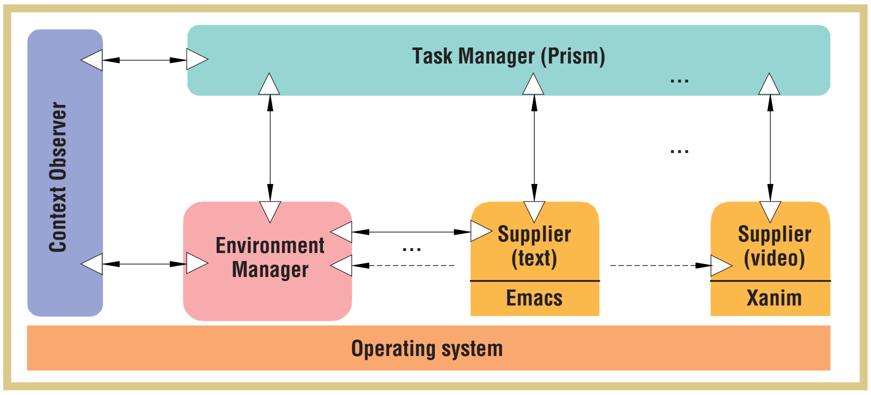


Figure 6. Prism's architecture.

ronment change (such as wireless bandwidth), Aura attempts to adapt ongoing tasks to accommodate the change, possibly reconfiguring certain services or replacing one service with another.

For Aura to achieve these goals, it is crucial that the system maintain some representation of user intent. Without this capability, determining which system actions will help rather than hinder the user is almost impossible. For example, suppose a user is viewing video over a network connection whose bandwidth suddenly drops. Should the system reduce the video's fidelity, pause briefly to find another higher-bandwidth connection, or advise the user that the task can no longer be accomplished? The correct choice depends on the task.

In a major departure from existing systems, Aura introduces a new layer of system abstraction: the *task layer*. This layer, called Prism, sits above individual applications and services but below the user (see Figure 1). By explicitly representing user intent, the task layer makes available to the rest of the system a powerful basis on which to adapt or anticipate user needs. In the two scenarios presented earlier, for example, the system must know such things as the user's plans for moving from one location to another, resource requirements for the user's future computing activities, and the user's preferred privacy policies.

Key ingredients of Prism's architecture are

- Explicit representations of user tasks as coalitions of abstract services
- Context observation that lets Prism configure tasks in a way that is appropriate for the environment
- Environment management infrastructure that assists with resource monitoring and adaptation

Each of these capabilities is encapsulated in a component of the architectural framework: the *Task Manager*, *Context Observer*, and *Environment Manager*, respectively (see Figure 6). A set of components called *Service Suppliers* carry out the services needed to support a user's task. Finally, Prism's infrastructure supports interactions between those parts, built on top of existing middleware such as remote procedure call or Corba.[15]

Figure 7 illustrates an application of Prism's architecture, where the Aura environment at Fred's home cooperates with the Aura environment at his office to migrate tasks between the two locations. When Fred leaves one environment, the local Context Observer points out that fact to the Task Manager. The Task Manager then checkpoints the state of the running services in a platform-independent fashion and causes the local Environment Manager to pause those services. This information, along with Fred's task state, is stored in a distributed file space. When Fred enters his office environment, the local Context Observer notices the fact and informs the local Task Manager. The Task Manager reinstantiates the tasks by finding and configuring service suppliers in the new environment. This reconfiguration attempts to maximize the use of local resources, subject to various resource utility functions specified by the task.

Prism's architectural framework has several important benefits. By representing user tasks explicitly, it provides a placeholder to capture user intent that can guide the search for suitable configurations in each environment. By representing tasks as service coalitions, Aura can determine when to support the essential services in a task, instantiating them jointly or otherwise providing early warning to the user when that is not possible. By providing an abstract characterization of the services in a task, the infrastructure can search heterogeneous environments for appropriate matches to supply those services. By providing the environment with self-monitoring capabilities, the infrastructure can detect when task requirements (such as minimum response time) are not being met and can deploy alternative configurations to support the task.

Our current prototype supports migrating tasks between two types of environments: Windows and Linux platforms. For example, suitably-wrapped Microsoft Word and Emacs become suppliers of text editing services on their respective platforms, while Media Player and Xanim

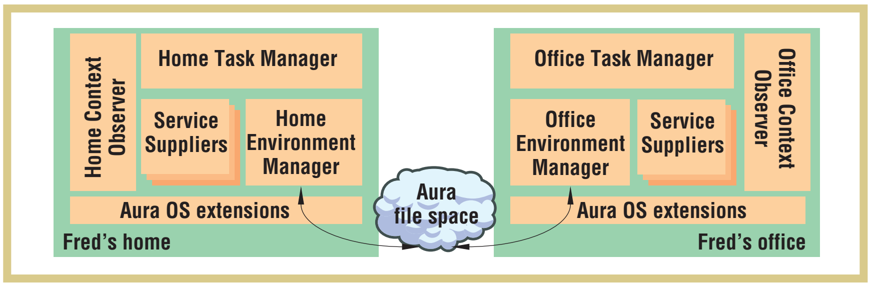
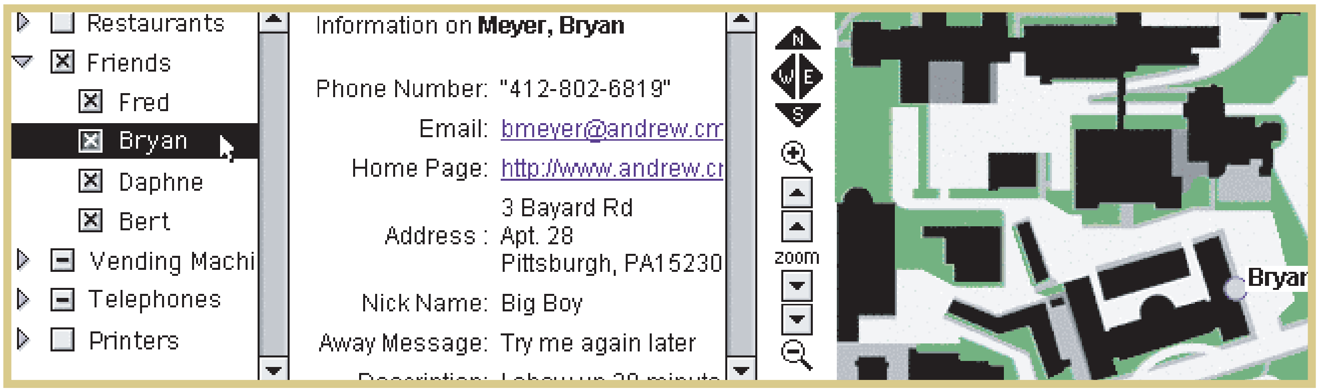


Figure 7. Task mobility in Aura.



Information on **Meyer, Bryan**

Phone Number: "412-802-6819"
Email: bmeyer@andrew.cm
Home Page: http://www.andrew.cr
Address : 3 Bayard Rd
Apt. 28
Pittsburgh, PA15230
Nick Name: Big Boy
Away Message: Try me again later

Figure 8. Portable Help Desk visual interface.

become suppliers of video-playing services. The current implementation of the Environment Manager has rudimentary service registry abilities built on top of existing solutions such as Jini[16] and Salutation[17] and relies on distributed file systems such as Coda[3] for file access across environments. Currently, we are extending Prism to support other environments, such as those based on PDAs and smart rooms. We are also developing mechanisms to manage the interaction of several users' tasks.

Context-aware applications

A key component of our research is developing applications that use the Aura infrastructure. One set of these applications, called *Handy Andy*, has been designed with the goal of supporting on-campus collaboration. Two of its core applications are the *Portable Help Desk* and *Idealink*.

Portable Help Desk

PHD is a context-aware application built on two fundamental services: spatial awareness and temporal awareness. Spatial awareness includes a user's relative and absolute position and orientation. Temporal awareness includes the scheduled time of public and private events.

PHD lets a user determine the location of teammates on campus as well as information about them. It displays maps of the immediate area, which indicate resources and nearby people. It also provides other services, such as notifying the user of the closest available printer or where food might be available. This capability builds on Aura's people location services.

PHD has both a visual and an audio interface. Each interface supports users in different contexts. A user who is walking around campus is likely to be less distracted by the hands-free speech interface, while a stationary user might want the richer visual interface. Both interfaces, however, are driven off the same database of information and underlying Aura services.

Figure 8 illustrates PHD's visual user interface. People and resources are selected in the left pane, the results of the queries are presented in the middle pane, and locations of people and resources are displayed in the right pane. For the queries made in Figure 8, the speech interface's transcript would be

User: "Locate Bryan."
Speech PHD: "Bryan is located in Hamburg Hall."
User: "What is Bryan's phone number?"
Speech PHD: "Bryan's phone number is 412-802-6819."

PHD delivers relevant information to the user in both a proactive and user-driven manner. It delivers proactive information to users when they are interacting with Aura infrastructure resources, such as printers. For example, when a user begins a print job, PHD alerts the user if there is a large print queue and suggests a nearby printer with a shorter one. PHD can also suggest a printer near the destination of a user en route. An example of a user-driven interaction is a request from a design group to locate a missing colleague.

Idealink

Idealink is a virtual collaboration environment that facilitates planned and ad-hoc collaboration among mobile users. Its goal is to provide easy access to information needed to initiate and conduct collaborative design meetings. It provides users with features that let them communicate their ideas to others via a shared distributed whiteboard. In addition to providing standard pen and text tools, it supports multiple channels, enabling simultaneous collaborative sessions. This feature lets teams in a large class use Idealink simultaneously without interfering with other teams' whiteboards. Figure 9 contains a screenshot of the Idealink interface.

Idealink is integrated with PHD to retrieve information related to users' preferences and schedules. PHD knows what meeting is taking place by consulting a user's calendar; this also lets the system automatically determine who should be included in the Idealink session. Aura's pervasive computing infrastructure stores Idealink preferences, including the tool palette layout and keystroke combinations that the user selects. Idealink combines each user's additions to the session and distributes these updates to each client. At the end of the meeting, Idealink archives the session.

In pilot user studies, we found that an Idealink-based design task takes less time than with a traditional whiteboard (see Table 2). In addition, participants made fewer errors. We asked two groups of four participants to collaborate on the design of a stereo remote control. We instructed them to layout a remote control with a predetermined list of features, working until they came to consensus about a final design. The participants were undergraduate students in mathematics, computer science, human-computer interaction, electrical engineering, and English. We recorded the time taken to complete the task and the number of errors they made during the process. Errors include things such as one participant asking

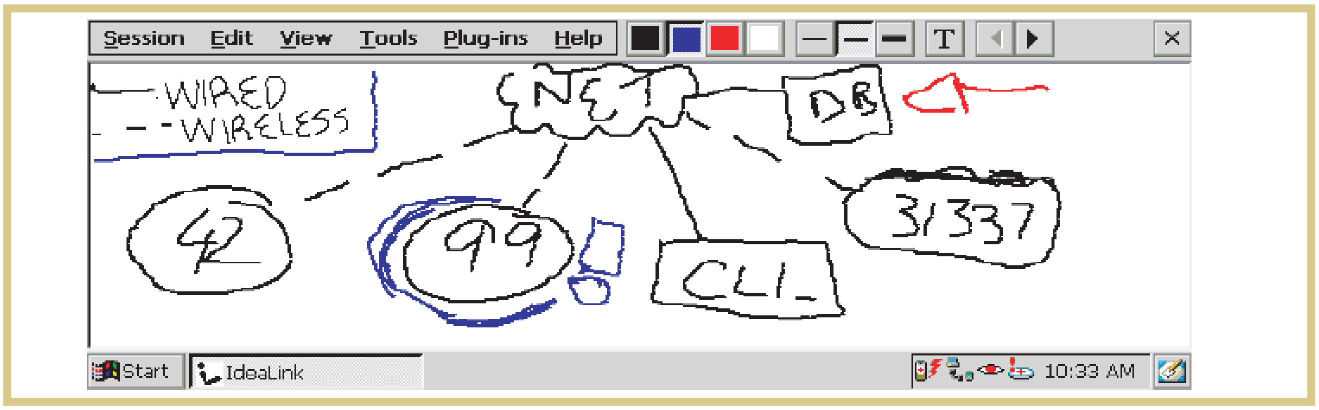


Figure 9. Idealink application.

TABLE 2
Comparing Idealink with whiteboard performance.

Metric	Idealink	Whiteboard
Group 1: Time	10:14	17:13
Group 2: Time	15:26	17:17
Population Standard Deviation	2:30	~0
Group 1: Errors	1	3
Group 2: Errors	4	6
Population Standard Deviation	1.5	1.5

another participant to explain what was written because the original content was not understandable, redrawing an illustration to make it more understandable, and competition to use the whiteboard.

Idealink let the participants be more effective by allowing interactions not possible with a traditional whiteboard. For example, because it lets users edit simultaneously, they could more easily reference and modify the shared design. Also, interaction was more evenly distributed among participants using Idealink. In particular, unlike face-to-face meetings in which a few people tend to dominate the discussion, with Idealink, more people could interact electronically. In addition, it let meeting participants work in parallel, review each other's efforts, and converge on a common result. Its replay feature let participants mix and match different versions of an illustration that evolved over time.

In the near future, we will expand this experiment, increasing the number of participants to eight groups. This will help us verify these results and might reveal additional Idealink benefits and drawbacks. We also plan to integrate a meeting moderator's assistant, which will guide participants to follow specified meeting and collaboration practices by reminding them of steps that they should take during the meeting. We'll supplement this capability with templates that easily categorize captured sketches and that can capture a series of ideas while sketching design. We'll also add voting mechanisms for anonymously voicing opinions during a meeting and a mechanism for categorizing and selecting the best ideas.

Integration and deployment

An important component of the Aura project is developing a prototype for nontrivial use. To evaluate some aspects of Aura on a larger scale, we are deploying parts of it on the CMU campus and developing several applications that the campus-wide community can use. The campus is an attractive deployment environment, not only because we are familiar with it and it is easily accessible, but also because it has features that are targeted by Aura. For example, it has mixture of "smart rooms" (for example, conference rooms that have been upgraded with a variety of devices and sensors) and device-poor environments (such as lawns and parks). It also has a large community of mobile users, who routinely rely on the campus-wide wireless network for access to information and computing systems.

Our efforts toward Aura deployment have focused on two main areas. The first is a set of contextual information services.[18] These services provide information about the entities of interest in a pervasive computing environment (devices, people, physical spaces, and networks) and also about their relationship (such as "Where is Joe?"). Although some of this information is static, some of the more interesting information is dynamic and will be used by Aura to perform self-tuning and adaptation.

The second area of deployment is in developing applications that exploit the Aura infrastructure. In addition to the applications we described earlier, we are developing prototype implementations that could be used in the Jane and Fred scenarios.

In the future, we plan to extend Aura's capabilities to more fully integrate the components we've described and to make them available to the campus community. We also plan to continue developing applications such as PHD and Idealink.

A final thrust of the project will be to evaluate Aura's impact using human factors studies to evaluate both its components and the integrated Aura system. Cognitive task analyses yield psychologically valid descriptions of peoples' task performance.

They are guided by formal descriptions of the task (assumptions and goals) and constraints on task performance. In our user studies, people perform multiple tasks, from fairly simple ones, such as organizing team meetings, to significantly more complex tasks such as refining the design of complex components. Our user studies testbed is fully instrumented to record team member activities, including time on task, keystrokes, mouse clicks, and tasks completed. Aura's focus on reducing user distraction drives our choice of metrics, including task execution time, errors, number of context switches, and task quality. P

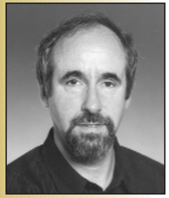
ACKNOWLEDGMENTS

We acknowledge Mahadev Satyanarayanan for his role as a key principal investigator in Project Aura. Much of the work described in this article is directly due to Satya and his students, and he coauthored earlier drafts of this article. We also thank the students and colleagues in Project Aura—in particular, Joshua Anhalt, Rajesh Balan, Jason Flinn, Glenn Judd, David Kogan, Matthew Kornyak, Takahide Matsutsuka, SoYoung Park, Vahe Poladian, Raj Reddy, Bradley Schmerl, Shafeeq Sinnamohideen, and João Sousa. DARPA contracts N66001-99-2-8918 and F30602-00-2-0616 have sponsored Project Aura. Opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of DARPA.

REFERENCES

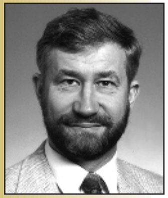
1. M. Satyanarayanan, "Pervasive Computing Vision and Challenges," *IEEE Personal Comm.*, vol. 6, no. 8, Aug. 2001, pp. 10–17.
2. M. Satyanarayanan, "Mobile Information Access," *IEEE Personal Comm.*, vol. 3, no. 1, Feb. 1996, pp. 26–33.
3. M. Satyanarayanan, "Evolution of the Coda File System," to appear in *ACM Trans. Computer Systems*, ACM Press, New York, 2002.
4. J. Flinn, D. Narayanan, and M. Satyanarayanan, "Self Tuned Remote Execution for Pervasive Computing," *Proc. 8th IEEE Workshop Hot Topics in Operating Systems*, IEEE Press, Piscataway, N.J., 2001.
5. F. Chang and G. Gibson, "Automatic I/O Hint Generation Through Speculative Execution," *Proc. 3rd Symp. Operating System Design and Implementation*, Usenix, Berkeley, Calif., 1999, pp. 1–14.
6. D. Duchamp, "Prefetching Hyperlinks," *Proc. 2nd Usenix Symp. Internet Technologies and Systems*, Usenix, Berkeley, Calif., 1999.
7. M. Satyanarayanan, "Caching Trust Rather Than Content," *Operating System Rev.*, vol. 34, no. 4, Oct. 2000, pp. 32–33.
8. P. Dinda, *Signal Prediction and Its Application to Real Time Scheduling Advisors*, doctoral dissertation, School of Computer Science, Carnegie Mellon University, 2000.
9. R. Wolski, N. Spring, and C. Peterson, *Implementing a Performance Forecasting System for Metacomputing: The Network Weather Service*, tech. report TR-CS97-540, Computer Science, Univ. of California, San Diego, 1997.
10. P. Francis et al., "An Architecture for a Global Internet Host Distance Estimation Service," *Infocom '99*, IEEE Press, Piscataway, N.J., 1999, pp. 210–217.
11. P. Dinda et al., "The Architecture of the Remos System," *10th IEEE Symp. High Performance Distributed Computing*, IEEE Press, Piscataway, N.J., 2001, pp. 383–394.
12. A. Smailagic et al., "Towards Context Aware Computing," *IEEE Intelligent Systems*, vol. 6, no. 3, May/June 2001, pp. 38–46.
13. P. Bahl and V.N. Padmanabhan, "Radar: An In-Building RF-Based User Location and Tracking System," *Proc. IEEE Infocom*, IEEE Press, Piscataway, N.J., 2000, pp. 775–784.
14. P. Bahl, V.N. Padmanabhan, and A. Balachandran, *Enhancements to the Radar User Location and Tracking System*, tech. report MSR-TR-00-12, Microsoft Research, Redmond, Wash., 2000.
15. *The Common Object Request Broker: Architecture and Specification*, 2.6 ed., Object Management Group, Needham, Mass., 2001.
16. K. Arnold et al., *The Jini Specification*, Addison-Wesley, Reading, Mass., 1999.
17. *Open Source for Service Discovery*, Salutation Consortium, 2001.
18. G. Judd and P. Steenkiste, "Providing Contextual Information to Ubiquitous Computing Applications," submitted for publication, 2002; www.cs.cmu.edu/~aura/docdir/contserv02.pdf.

the AUTHORS


David Garlan is a principal investigator on the Aura project and is an associate professor of computer science in Carnegie Mellon University's School of Computer Science, where he works in the areas of software architecture, formal methods, and pervasive computing. He currently manages the ABLE Project, which focuses on architectural design tools and support for self-adaptive systems. He received his PhD in computer science from Carnegie Mellon University. Contact him at garlan@cs.cmu.edu.

Daniel P. Siewiorek is a Buhl University Professor of computer science and electrical and computer engineering at Carnegie Mellon University. He is also director of the Human Computer Interaction Institute. His research interests include systems architecture, reliability, modular design, wearable computers, and context-aware computing. He received his BS in electrical engineering from the University of Michigan and his MS and PhD, both in electrical engineering, from Stanford University. He is a member of the IEEE Computer Society, the ACM, Tau Beta Pi, Eta Kappa Nu, and Sigma Xi. Contact him at dps@cs.cmu.edu.

Asim Smailagic is a senior research scientist at the Institute for Complex Engineered Systems and Department of Electrical and Computer Engineering at Carnegie Mellon. He is also director of the Laboratory for Interactive Computer Systems at CMU. He received the fulbright postdoctoral award at Carnegie Mellon in computer science in 1988. His research interests include pervasive and wearable computing, advanced computer architectures, rapid prototyping, and wireless communications. He has a BS in electrical engineering and an MS in computer science from the University of Sarajevo, and a PhD in computer science. Contact him at asim@cs.cmu.edu.


Peter Steenkiste is an associate professor of computer science and electrical and computer engineering at Carnegie Mellon University. His research interests are in the areas of networking and distributed systems. He has done research in high-performance networking, high-performance distributed computing, and network quality of service. He currently manages the Libra project, which supports scalable network services deployment through service composition. He is also a principal investigator on the Aura project. He has an engineering degree from the University of Gent, Belgium and an MS and PhD electrical engineering from Stanford University. He is a senior member of the IEEE and a member of the ACM. Contact him at prs@cs.cmu.edu.