HomeOS: Context-Aware Home Connectivity ‡

Neal Rosen, Rizwan Sattar, Robert W. Lindeman, Rahul Simha, Bhagirath Narahari
Department of Computer Science, The George Washington University
{nrosen | rizwan | gogo | simha | narahari}@gwu.edu

Abstract[*]

The trend of cheaper ubiquitous and pervasive technology has reached a critical point. A standardized integration of devices in the context of a home environment will be possible in the near future. HomeOS is an attempt to provide a standard interface for a context-aware application and device integration system. HomeOS offers three key features: cross-platform application portability; application following; and management of multimedia streams.

HomeOS allows applications and home management services to run on a central server written in Java, and displays interface widgets on remote devices capable of acting as terminals. By implementing pre-defined Java interfaces, applications and devices can be integrated into HomeOS, which handles the details of routing events to the current location of a designated user.

1. Introduction

In 2001, The George Washington University, with seed money from America Online, began an effort to design the infrastructure to support applications in the home of the 21st century. This paper presents work we have done in designing a context-aware application support layer for connected homes.

Every person has a different vision of the Home of the 21st Century. While the time has not yet arrived when domestic robots and artificial intelligence are commonplace in the home, other technological advances provide fertile ground for research and development; devices are getting smaller, smarter, and most importantly, cheaper; Java technology can be found in devices as small as a thumbnail; our cell phones and PDAs possess significant processing power; and wireless access points now provide our homes with a cheap and effective way to network our devices and our computers. With the current state of technology in our homes, it is only requisite to look toward *integration* as a key to unifying the home experience. It is now possible to make the various devices that pervade our homes work together. Context-aware, personalization-capable, home-automation systems will be possible in the very near future.

The integration of devices with different technologies, conflicting standards, and incompatible interfaces can be approached the same way as the construction of a single computer. The "motherboard" connects all devices through standard ports and slots, the operating system (OS) provides a unified way of accessing these resources, and applications translate input into usable output. We can view the hardware infrastructure of the networked home as the motherboard connecting physical input and output devices. The OS to support the nature of typical interaction in the home environment, what we call the Home Operating System, or *HomeOS*, has distinct requirements that we address in this paper.

2. Sample Scenario

Allison returns home from work as HomeOS has signaled the X10 system to turn on the garden lights in the dimming light of dusk. Upon approaching the front door, radio frequency identification (RFID) sensors pick up her identification tag, embedded in her shoes, and automatically allow her to enter. A display panel at the entrance has a notification that she has four unread e-mail messages. She walks to the kitchen and accesses the panel on the front of the refrigerator to where the e-mail notifications have migrated. She begins putting her new groceries inside as the fridge automatically updates its inventory based on the tagged items.

Sitting down on her couch, Allison finally accesses the e-mail notifications now on her TV screen. An e-mail message is converted to speech and starts to stream through the TV speakers. At one point while listening to her e-mail, it is automatically paused and she is notified that someone is at the front door. The TV display simultaneously begins to stream a video signal from the video camera observing the front door. As she moves to the door, her e-mail continues to be read aloud, following her to the speakers placed near the entrance.

[*] This project was supported in part by NSF grant MRI-0216137 and by a grant from America OnLine

3. Previous Work

Microsoft Corporation has long been interested in home automation, and one of their smart homes reflects the direction in which they are headed. In the Microsoft Digital Home [1], visitors are shown that they can have their food scanned by bar code into the microwave, which will adjust the settings for appropriate heating of food. Similarly, food can be scanned in to display possible recipes based on the input ingredients. Visitors to the home can even leave voice messages on a flat panel computer attached to the front door. These are very similar to the kinds of services that the HomeOS will support.

The critical difference between the Microsoft Digital Home and our HomeOS-based system is in object identification and location. Our system will have an installed RFID sensor network to help it discover all tagged objects in a home, without the need for a user to scan objects in. Furthermore, the sensor network can be used to locate tagged-object locations inside the home, whereas Microsoft's approach uses object identification with bar code reading and biometric sensors. Another critical difference is that all the smart devices (the microwave, the recipe display, *etc.*) are not connected with each other in the Microsoft example. Such applications would exist centrally on a server in HomeOS, allowing other devices and applications access to these services, eliminating the requirement of the user having to be at a particular station to execute a particular command.

Georgia Institute of Technology's Aware Home Research Initiative [2] attempts to keep track of a person's activities inside the home. This is especially useful in a home for an elderly individual, or in an institutionalized care facility. Children of the elderly occupants in the home are able to keep track of their parents through a remote reporting device in the form of a digital family portrait. The system at the home uses sensors to record sleeping and eating habits, and movement inside the house (*e.g.*, doing "chores").

Stanford University's ICrafter [3] also attempts to provide applications to the user through various appliances. Developers can create context-aware services that are portable across various devices through a UI framework. HomeOS expands on this idea by allowing applications to exist centrally and be accessed remotely, similar to the way the X Windowing system exports displays. This offloads the work done by resource-limited devices, while still providing a sophisticated array of services.

HomeOS addresses some of the major issues in home automation (*e.g.*, context-awareness and following, object identification and location) by using an RFID sensor network. With objects (and people) being tagged, the HomeOS can serve this information to all applications that need them, allowing them to easily implement context-sensitive features for the home environment. Other emerging technologies, such as ultra-wide-band techniques, can be easily integrated into the current system, as the tracking data is simply taken from a database.

4. HomeOS

The HomeOS is a large-scale Java application which runs on a centralized server. Its primary purpose is to provide a standard communications system between all smart devices in a home and to be an event handler for these devices. Applications and services run on HomeOS to provide functionality for controlling devices and various aspects of the home environment.

A computer operating system allows users to interact with applications that have been installed, and also use those applications to communicate with peripheral devices. Much like its computer counterpart, applications can be installed on HomeOS, and it allows these applications to talk to the various devices connected to it.

Figure 1 shows a diagram with multiple devices and terminals connected to the central HomeOS server. The person locator provides information to the system about the location of persons within the home environment. FridgeApp, MicrowaveApp, and MyApp are three different applications running on the HomeOS server, and are accessed from remote terminals.

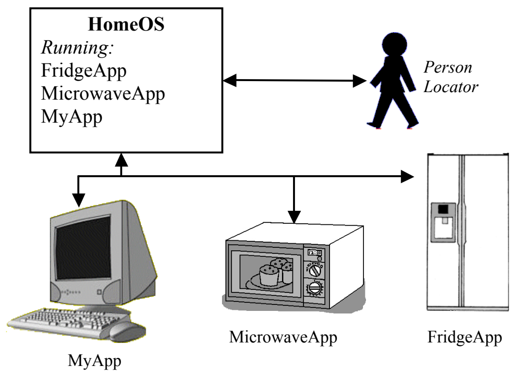

Figure 1: HomeOS Network

The applications typically installed in HomeOS are for controlling the home environment, accessing personal productivity information such as e-mail or an appointment calendar, and playing streaming media. A HomeOS application would allow its user to control the heating system in the house, manage the lights, and access the security system. While a typical operating system might be used to activate a Web-cam and display stored files, HomeOS allows security cameras around the house to be monitored, and lets the user control their microwave, retrieve a list of the contents of the refrigerator, *etc*.

HomeOS also tracks occupants within the home through the use of RFID technology. RFID tags are embedded into all objects, that may need to be located, with RFID antennas deployed throughout the home. Recent RFID supply-chain initiatives by WalMart and others strengthen the prospects that RFID may become as ubiquitous in the near future as bar codes are today. With the ability to locate objects with RFID tags, a powerful infrastructure is laid for future applications to take advantage of this kind of information.

5. Applications

Applications in HomeOS have the following properties:

- Run centrally on the HomeOS server.
- Do not have a standard user interface, but rather are accessible remotely through any terminal in the home.
- Either provide home management and personal information services (heating control, address book), or represent smart devices in the home environment (remotely controllable microwave, a refrigerator capable of reporting its contents).

When a new controllable device is integrated into the HomeOS system, an application must be developed for accessing that device to provide an interface between a user and the device via a remote terminal. For example, an application that would be developed for controlling a smart microwave would provide a user the ability to specify a length of time to run the microwave and to notify the user when the time has elapsed.

Though there are many possible applications for HomeOS that are used mainly for accessing smart devices, other applications need not communicate with any device at all. These applications provide remote access to one or more data resources made available by the system. An example of this might be a centrally located appointment book that may be accessed from any remote terminal, or an e-mail client that talks to an IMAP server. Another example would be an application to monitor the movement of persons throughout the home environment (perhaps to keep an eye on a young child's movements).

6. Devices

Any programmable device or appliance with a network connection can be integrated into the HomeOS system. These devices may include microwaves, refrigerators, radios, lights, JavaPhones, cameras, doorbells, motion detectors, and smoke detectors. Currently this project is being tested with a microwave (simulated with an iPaq), a radio (using a radio transmitter connected to a PC), lights (using X10 devices), a camera, a doorbell, and a PingTel JavaPhone.

For each new device integrated into the HomeOS system, a new component (a device interface) is developed to provide a means for communication between HomeOS and the remote device. This allows the system to be extended to communicate with devices using various types of network protocols (TCP, FTP, HTTP, RTP, *etc*.). Also, devices may be connected over a wireless or wired network.

A device which meets a minimum set of requirements may also be implemented to act as a remote terminal, allowing a user to access the applications of the system through the device. The minimum requirements include some sort of display to show, at a minimum, text, and some input method. The display can be graphical or text-based. Input capability could be provided using a keyboard, a pointer device (such as a mouse or stylus), a set of buttons (like on a microwave or JavaPhone), or some other input method.

7. Kernel

The kernel provides a standard interface through which devices and applications interact. The kernel has the following properties:

- Enumerates and maintains the applications and devices registered with the HomeOS system
- Keeps track of the locations of users within the home environment
- Provides home environment information to applications, such as types of devices available, the location of its users, *etc*., and
- Enables applications and devices to communicate with each other

8. Registry

HomeOS maintains much information used to assist in event handling and data communication between devices and applications. This information includes device and application states, person and object location information, and user information. The registry is a module in the HomeOS Kernel Layer that provides all components of the HomeOS server to access this information.

The registry interfaces with a central home database, which maintains this system information.

This registry allows HomeOS to determine where to send data, detect when a user enters or exits the home or moves locations, access user preferences which may be utilized by applications, and locate tagged objects in the home environment.

9. HomeOS Layers

As shown in Figure 2, the major components of the HomeOS system are divided into the three layers: a Device Layer, a Kernel Layer, and an Application Layer. Looking at the figure from the top down, the applications communicate with the Application Manager (in the Kernel Layer). The Kernel Layer passes data between the Application and Device Layers. Each device has a separate device interface associated with it, which passes data to its device and receives data over a network.

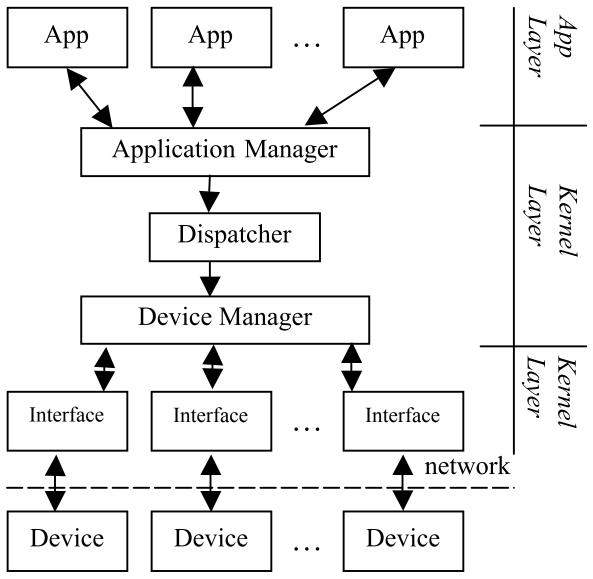

Figure 2: HomeOS Architecture

9.1. Device Layer

The Device Layer provides the interface for adding a device-specific interface component to the system for new devices. This layer also enables communication between the HomeOS server and the remote devices. The Device Layer contains all the device interfaces, which implement a standard software interface ("DeviceDriver") to provide data communications between the devices and the kernel.

9.2. Kernel Layer

The Kernel Layer passes data between the Device Layer and Application Layer. The data being passed across this layer is stored in a wrapper data structure called a DataObject. This provides a level of abstraction to the system, allowing any type of data to be passed. Data to be propagated through the system is wrapped in, and extracted from, a DataObject data structure by device interfaces and applications.

In order for an application to send data to a device or other application, it passes the data and the destination device to the Application Manager. The Application Manager passes this information to the Dispatcher. The Dispatcher then queries the registry to determine the appropriate device interface or application to receive the data. If the data is to be sent to another running application, the Dispatcher passes the DataObject back to the Application Manager, which then determines the appropriate application to pass on the data.

If the destination of the data is a device, it can be specified in one of three ways. A specific *device ID* can be specified to pass the data to a particular device. The Dispatcher passes this information to the device interface of the device with the matching device ID.

Second, a *location* can be specified. The Dispatcher then queries the registry to determine the capabilities of all devices located in the specified area, and sends the data to the device best able to process the information. This is useful, for example, if an application wants to send an audio message to a particular room or location. The device in that room with the best audio quality is chosen as the destination. The Dispatcher then sends the data to the device interface associated with that device

Another way to direct data is by specifying a *person* as the destination for a DataObject. The Dispatcher is also able to query the registry to determine the location of the specified person and then determine the best device in that person's location to which to send the data. In order to determine the location of persons within the home environment, the Dispatcher uses another kernel component called the PersonLocator.

The PersonLocator component queries the registry database which maintains the current location of all individuals within the home environment. The database is updated by a program connected to a series of RFID receivers located in strategic areas within the home environment.

Passive RFID tags are worn by each person in the home environment (possibly in a wallet or a shoe). When a person wearing a RFID tag enters or leaves the home environment, or changes locations within the home environment, the registry is updated to reflect this change.

9.3. Application Layer

The Application Layer consists of the applications developed for the HomeOS system and the HomeOS

Desktop remote desktop application. This layer provides an interface for applications to communicate with the Kernel Layer, to pass data to and from the devices and other applications, and to provide access to the resources made available by the Kernel Layer.

Applications are integrated into the system in a way similar to how device interface components are integrated. A parent component, HomeOSApp, is extended by applications to provide the interface to the Application Manager in the Kernel Layer. The HomeOSApp module provides an interface for sending and receiving DataObjects, querying the devices connected to the system for information about the devices, and determining the location of persons in the home environment.

The HomeOS Desktop is an application that provides a remote terminal session for each user logged in to the system from a remote terminal. While the user is logged in, the Desktop sends a user interface description to the remote terminal of the user associated with that Desktop instance and receives user events back. The event is processed and an updated user interface is retransmitted to the device running the terminal.

Through the use of the PersonLocator component, the HomeOS Desktop applications can determine the location of their current user within the home and direct the terminal session to a device nearest the user with terminal capability. This allows the user to move about the home while having his/her terminal session follow them throughout the home. For example, an application implementing a streaming-media player can send its output to the devices nearest the owning user.

10. XML-Based User Interface language

The applications developed for HomeOS run on the HomeOS server, and have no standard user interface. These applications are only accessed remotely from a terminal. Therefore, a UI description language was designed to send a description of what needs to be shown on the terminal device, and the client for the output device displays the UI in whatever way is most appropriate.

As stated earlier the minimum requirements for a device with terminal capability are rather basic. There must be some form of display that can show text, and an input mechanism such as a keyboard, mouse, stylus, physical buttons, *etc.* Some terminals may only have a console display with keyboard input; while others may have full GUI capabilities (soft buttons, text boxes, dialog boxes, *etc.*). Most terminal devices will fall somewhere in between these two examples.

The UI language had to be general enough for all types of terminals to be able to handle. We decided upon an XML-based UI language with four basic UI elements. A description of each of these elements, along with examples of how they might be implemented on a terminal with GUI capability and a terminal with text only capability is shown in Table 1.

Table 1: UI Element Mappings

UI Element	Description	GUI Example	Text Example
Message	Provides text to be displayed	Dialog box	Text to console
Data Entry	Prompt text & single- or multi-line text entry	Label with text field	Show prompt & wait for input
Choice	Prompt text & list of choices	Label & radiobuttons or checkboxes	List choices & wait for input
Button	Strings to display	Buttons	List buttons & wait for input

11. Media Stream Management

Streaming audio and video data throughout the home environment provides additional flexibility by allowing residents to move freely about the house. In place of a visual message or alert displayed on a terminal, an audible message may be generated from text, and directed to the location of the intended recipient. This may be used by a microwave service notifying its user that the microwave has finished running. E-mail messages may be converted to speech and streamed to the audio output device nearest the user, dynamically following the user as he or she moves about the home. Other potential sources of streaming audio include a music source (such as an MP3 player), a doorbell, security systems, and intercoms.

In addition to audio streaming, many applications could make use of video streaming capabilities. A camera positioned at the front door could be used as part of a security system sending visual data to a monitor inside the home. The camera outside the front door may also be triggered by the doorbell to send the video feed to the family-room television so the resident can immediately see who is at the front door.

With many services running in the home that make use of media streaming, collisions are likely to occur. For example, the resident may have music playing when an e-mail arrives. The audio e-mail message would be difficult to understand while the music is playing. Also, someone may ring the doorbell, interrupting the e-mail message and providing further confusion for the user trying to distinguish between the multiple audio streams. To resolve these types of collisions, a stream type and priority value are associated with each media stream. This information is used by the system to perform user-specified actions based on what types of streams are playing when a new stream arrives for playing. One possible action is to pause one stream while another is playing, and resume once the second stream is finished. Other actions include lowering the volume, muting, and stopping a particular stream. Figure 3 shows a scenario where an audio e-mail message arrives while a music stream is playing. The volume of the music stream is lowered while the e-mail message plays. Once finished, the volume is restored to its previous setting.

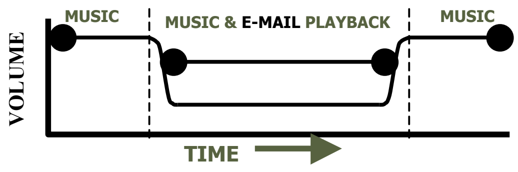

Figure 3: Stream Collision Example

Whereas it might make sense to have music lowered in volume when an e-mail message is played, it would not be appropriate to lower the volume of an e-mail message while the doorbell alert is playing. A better solution is for the e-mail to be paused while the other audio stream is playing, and then resume once the other stream has finished. These actions are organized into a matrix, as shown in Figure 4. The rows represent a type of stream currently playing and the columns represent a new stream to be started. The actions in the matrix are used by the HomeOS system to determine what action to perform on the streams in response to collisions. A similar matrix could exist on a system-wide basis, or each individual could have a personalized matrix, based on their preferences.

	Music	**Message**	**Info**	**Alert**
Music	Pause	Mute	Mute	Mute
Message	No action	Pause	Pause	Pause
Info	No action	No action	Pause	Pause
Alert	No action	No action	No action	Pause

Figure 4: Action Matrix for Audio Streams

If a Music stream is running and an audio stream of type Info arrives, the Music stream is muted while the Info stream is running, and then the volume is restored once the Info stream has finished.

12. Applications within the HomeOS

While the HomeOS creates the infrastructure for communications between connected peripherals, its real success will depend on the kind of applications that are developed for it. We have implemented several applications in order to test our infrastructure design. We have completed the initial version of the HomeOS server, implemented versions of simple UI clients using the Java Swing library and text-only techniques, and are in the midst of implementing X10 and PingTel JavaPhone device interfaces. In addition, an audio-only media stream management facility has been implemented, including an e-mail audio reader and an mp3 player. We have also implemented a text-to-speech alert monitor that devices can send a simple ASCII message using sockets, and the alerts will be handled gracefully. This system is in the process of being integrated with the HomeOS.

There are several planned applications under consideration for our home lab. One allows visitors to leave video messages through a video camera at the front door if no one is home. Upon returning home, the occupant will have a list of video messages waiting for playback on the nearest terminal.

The HomeOS applications could also offer more life-style-supplementing services. Simple TV program reminders could be set into a TV Guide application which would then notify occupants when a selected television program is on. Instant messaging could be implemented to exist on any terminal in a home. Capable of following its intended user around the house, this would free the user from having to be at a single location to send and receive messages.

13. Conclusion and Future Work

The HomeOS is a framework for connecting devices found in a home setting in a transparent manner. In addition, it supports applications that are accessible from interface terminals by occupants who are free to move about in the home environment. These applications include those designed to control connected devices, as well as those designed to support access to personal and entertainment data, such as e-mail and digitized music. Currently, our focus is on improving the robustness of the HomeOS. Future work will focus on increasing the number and scope of applications available for the HomeOS.

14. References

[1] Ross, R. "The digital home of tomorrow", Toronto Star, February 2003.

[2] Abowd, G. A. Bobick, I. Essa, E. Mynatt, and Rogers, W. "The Aware Home: Developing Technologies for Successful Aging", Proceedings of AAAI Workshop and Automation as a Care Giver, July 2002.

[3] Ponnekanti, S., Lee, B., Fox, A., Hanrahan, P., Winograd, T. "ICrafter: A Service Framework for Ubiquitous Computing Environments," Proc. of UbiComp2001, pp. 56-75.