Appliance Aggregation Architecture (A^3)

Dejan Milojicic[1], Philippe Bernadat[1], Rick Corben[2], Ira Greenberg[2],
Rajnish Kumar[3], Alan Messer[4], Dan Muntz[5], Eamonn O'Brien-Strain[1],
Vahe Poladian[6], Jim Rowson[1]
Mobile and Media Systems Laboratory
HP Laboratories Palo Alto
HPL-2003-140
July 3rd , 2003*

Email: dejan@hpl.hp.com

appliance, aggregation, distributed systems, architecture

Technology is advancing and bringing personal IT appliances into every aspect of our life, with the intention of making life easier and better for users. Wireless phones, digital cameras, and tablet PCs are examples of personal appliances available today. In the near future, it is envisioned that watches, jewelry, and clothes will be connected in a personal area network, providing new opportunities to owners of these devices. Unfortunately, technology has become so complex for an average user that it is more often a burden than a benefit. For example, applications are hard to configure, administer, and use; appliances can be stolen, losing information that is often more valuable than the appliance itself; data is often on one of the appliances that is not currently being used; and applications and interfaces differ from appliance to appliance.

We propose the notion of an ensemble of aggregated appliances that presents a user with a common model of ownership, shared state, shared applications, and shared functionality. A united ensemble makes it easier for users to control their appliances; it provides users with transparently synchronized data and continuous access to applications and functionality from any appliance. We bring back the ease of use of fewer devices and fewer services, enabling users to focus on their lives and their needs rather than on technology. Appliance aggregation converts technology back to being an enabler rather than a friction.

* Internal Accession Date Only Approved for External Publication
[1] HP Labs, [2] Independent Contractor, [3] Rice, [4] Samsung Research, [5] CoroSoft Inc., [6] CMU

Appliance Aggregation Architecture (A^3)

Dejan Milojicic[1], Philippe Bernadat[1], Rick Corben[2], Ira Greenberg[2], Rajnish Kumar[3], Alan Messer[4], Dan Muntz[5], Eamonn O'Brien Strain[1], Vahe Poladian[6], and Jim Rowson[1]

dejan@hpl.hp.com (contact author)

[1]HP Labs [2]Independent Contractor [3]Rice [4]Samsung Research [5]CoroSoft Inc. [6]CMU

Abstract

Technology is advancing and bringing personal IT appliances into every aspect of our life, with the intention of making life easier and better for users. Wireless phones, digital cameras, and tablet PCs are examples of personal appliances available today. In the near future, it is envisioned that watches, jewelry, and clothes will be connected in a personal area network, providing new opportunities to owners of these devices. Unfortunately, technology has become so complex for an average user that it is more often a burden than a benefit. For example, applications are hard to configure, administer, and use; appliances can be stolen, losing information that is often more valuable than the appliance itself; data is often on one of the appliances that is not currently being used; and applications and interfaces differ from appliance to appliance.

We propose the notion of an ensemble of aggregated appliances that presents a user with a common model of ownership, shared state, shared applications, and shared functionality. A united ensemble makes it easier for users to control their appliances; it provides users with transparently synchronized data and continuous access to applications and functionality from any appliance. We bring back the ease of use of fewer devices and fewer services, enabling users to focus on their lives and their needs rather than on technology. Appliance aggregation converts technology back to being an enabler rather than a friction.

1 Introduction

The number of IT appliances available to users is increasing as computing technology improves. These devices come in two forms: personal—those you own, and environmental—those in the locale. Examples include cell phones, cameras, server appliances, and printers. In the future, users will carry one or more devices in an environment full of embedded appliances. This large number of devices creates several problems for users. First, today each device is designed to run alone. Even if it does connect to other devices, the connection is simplistic, pair-wise, low-level, and typically proprietary (e.g., Microsoft's ActiveSync). If some data or a password is used on a device, it is difficult to use it from/on other devices. Second, by acting alone, each device is limited to its own interface. It is not possible to use the larger screen with a cell phone. A more general problem is the ability to use the right device or resource for the task at hand. Finally, the number of device types is overwhelming in different sizes, weights, and interaction models.

However, multiple devices also present an opportunity for additional functionality and performance. They can be used for combined functionality and their resources

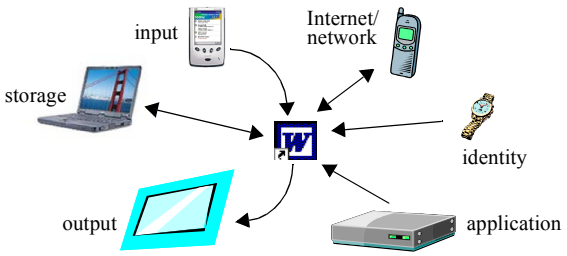

Figure 1: Example of Aggregation Use: Composed Appliance.

can be used in concert. For example, aggregating a PDA (address book, dialing, messaging), personal server (network connection), and a headset (voice I/O) to make a phone call instead of using a limited cell-phone (see Figure 1). A wireless LAN connection may provide improved voice quality and other media types, or the personal server may run complex tasks to save the PDA batteries. By combining personal and environmental devices together in an ensemble, devices become easier to use and offer more functionality and performance.

The rest of the paper is organized as follows. In Section 2, aggregation is put in a historical perspective. Section 3 presents a few scenarios. In Section 4, we dis-

Table 1: Evolution of Processing Infrastructure and Access

device type	"era"				
	early batch	multi-user	client/server	ad hoc utilities	planetary-scale computing
infra-structure	a computer a job at a time	a computer, multi-tasks	app. processor, file/print server	hardwired data centers	managed computer fabric
access devices	shared users, CDR, LPT	terminal shared by users	dedicated PC & workstation	separate PC, PDA, camera	ensembles

cuss A^3 characteristics. Section 5 presents the aggregation architecture and Section 6 presents prototype implementations. In Section 7, we discuss the benefits of A^3. Section 8 compares our work to related work and Section 9 summarizes the paper.

2 History

In some ways, appliance ensembles represent a natural evolution in information technology (see Table 1). If we consider computer history in terms of infrastructure and access, then we can identify a sequence of generations, each dominated by a particular architectural style. In the earliest days, one machine executed one task. Then we moved to multiple tasks on the same machine. Later, there was some functional disaggregation so applications ran on a primary processor supported by file servers, printer servers, etc. As the need to support massive Internet-based applications emerged, we have seen large data centers built from hundreds or thousands of computers wired together. More recently, Grids, as a notion of a managed fabric of computers have been developed as a more manageable, flexible way of achieving the same thing.

There has been a similar evolution in devices for accessing information (on-ramps and off-ramps). Initially, one device (e.g., line printer, card reader) connected to one machine was shared by many users working around the clock to lower the costs. Then, many users accessed one machine through individual (dumb) terminals. Later, dedicated PCs and workstations became the preferred access method. As the cost and size of IT devices continued to decline, individual users acquired multiple devices (e.g., PCs, laptops, PDAs) that worked well individually but not together. Finally, the concept of ensembles has emerged as a way to get more value from the profusion of IT devices.

3 Scenarios

This section illustrates levels of aggregation using four business scenarios. However, these could be easily extended to include home devices—video projector, RPTV, Hi-Fi, DVD, etc. People will encounter such devices at home but also in hotels, cars, and airplanes.

Level-1 ensemble ("always-available-appliances"): shared ownership and preferences; tailored for and works only for owner.

Jamie Decker is a reporter at HP Labs. Before heading out to interview Webb Conners,[1] an HP executive, she goes to the accessories room to get her equipment. She needs a video camera, a smart pen, a phone, and a tablet PC. Jamie's earring acts as her identity appliance—it authenticates her and permits her to enter the accessories room. Her identity is used to configure all her new appliances based on her preferences. In a few seconds before she leaves the room, all of the appliances preload the software required for today's assignment. It is the newest version of authoring software for imaging and annotation. The new accessories Jamie picks up become part of her ensemble along with the identity appliance that she wears as an earring. In addition, device preferences are shared across all her appliances: passwords, power-saving setting, time of the day, fonts, display background, etc. Any new appliance added to the ensemble automatically inherits the current ensemble preferences. While returning the old accessories, she notices that her old smart pen is missing together with the sensitive material stored in it. However, she is not worried, because if it cannot authenticate her as a user, the pen will not work and all the data stored in it is encrypted. If she gets close to the pen, she will be alerted to its location and will be able to return it to the accessory room later.

Level-2 ensemble ("always-available-information"): shared state of appliances. User can access data from any appliance.

The web browsers running on Jamie's tablet PC, her camera, and her phone all have same set of favorites and caches, and share the same data. If she starts editing and saves changes on one device, the changes are available on the others. The e-mail, instant-messaging, and telephony applications all share the same contacts. She took a few photographs with her camera before getting on a crowded train. On the train, she just had enough room to open her PDA to select the best photograph and rotate it to the correct orientation. She spoke a quick description about the subject of the photograph into her pen, and used it to annotate the picture. Later, in her office, she will use her tablet PC where the same photograph will be accessible to adjust the color and crop the photo before importing it into the interview.

Level-3 ensemble ("always-available-application"): single application running on all appliances.

1. Jamie Decker and Webb Conners are fictional people and the interview is a fictional activity.

Jamie's appliances run a new image authoring application developed especially for ensembles. Jamie doesn't view the application as being installed on a particular device. Whichever device she opens will have the editing application in the same state regardless of the device she was previously using. She doesn't have to save the file when switching from one device to another. On the train, using her tablet PC, she outlined the interview questions and interspersed them throughout Webb's bio and merger history pictures. While walking along the corridor to Webb's office, she opened her phone and corrected the punctuation. As she entered Webb's office, her tablet PC uses the very same application to display her questions on the screen. She is almost ready to start the interview.

Level-4 ensemble, "always-available-functionality": invisible applications, intuitive interfaces.

As she walks into the room, Jamie's ensemble automatically attaches a high-resolution video camera in the corner, speakers on the ceiling, and a few displays closest to the table. The camera automatically starts focusing on Webb and frames Webb and Jamie correctly as they move. It recognizes the context as an interview and knows who she is, and therefore can decide that it should film the other person as well. The audio and video clips from the outline get reproduced unobtrusively in the background as she proceeds with the interview. When the interview ends, the ensemble discovers the closest printer and prints the interview transcript while guaranteeing Jamie's privacy.

Returning to the train station, Jamie speaks additional impressions into her pen's microphone and the comments are automatically appended to the interview file. While standing on the crowded train, she previews her interview on the camera and does the first edits. When a seat becomes available, she opens her tablet PC, which has automatically loaded the video footage for editing. She does additional editing and annotates a note for the person doing the more detailed editing in the office. She marks it ready and the interview is automatically sent back to the newsroom. Having the same functionality and similar intuitive interfaces on all of her devices—voice input on the pen, buttons on the camera, and a keyboard on the tablet PC—made Jamie's task more efficient and enjoyable. She keeps thinking about Webb's words: "Today you came with a few appliances that harmoniously accomplished your task. Tomorrow, I hope you will only need to bring your earring and will be able to conduct the same interview even more effectively".

Table 2: Incremental Levels of Aggregation

Characteristic	Levels			
	1 (group membership)	2 (shared state)	3 (session)	4 (functionality)
Aggregate	ownership	+ system state	+ applic. state	+ service state
User Benefit	single login protection	+ shared files and connections	+ shared app space	+ shared functionality space
Transparency	appliance	+ external state	+ app requirements	+ user needs
Inherit	logon	+ VPN, cookies, caches	+ app preferences setup	+ user behavior
Illusion	private network	single computer/ OS	single app	single service
Access	secure paths	persistent data	app session state	services
Target	ensemble	apps	users	You!
Developers	low level system, hardware	+OS	+app model	+UI
Data propagated	membership	+ committed data	+ dynamic data	+ dynamic info
Changes propagated	when changed membership	+ transaction commit	+ user event	+ any event
User interacts with	appliance	appliance	ensemble	ensemble

4 Aggregation Characteristics

This section introduces the design considerations that must be addressed to achieve simplicity and other goals of appliance aggregation.

Aggregation Levels. In the scenarios, we defined increasing levels of appliance aggregation, from the simple ones, to more extensive levels. As the levels of aggregation increase, additional user benefits, transparency, and illusion are achieved. At the lowest level, *level-1*, appliances share ownership across the whole ensemble. At *level-2*, the aggregation is tighter and data is shared across an ensemble, e.g., in the form of caching or a distributed file system. At *level-3*, applications are accessible from any appliance in the ensemble. Finally, at *level-4*, the same functionality is accessible across an ensemble. Transparency increases from appliance-level, through external state and application requirements, to user needs. Illusion ranges from a private network, through a single computer/OS and a single application, to a single service.

Different levels are targeted to different beneficiaries: *level-1* targets an ensemble, *level-2* supports applications, *level-3* supports users, and *level-4* tailors an ensemble for a particular user. The different levels are addressed by different types of developers: *level-1* is addressed at low-level system and hardware, *level-2* at operating system, *level-3* at applications, and *level-4* at services. These and other aspects of aggregation are compared in Table 2.

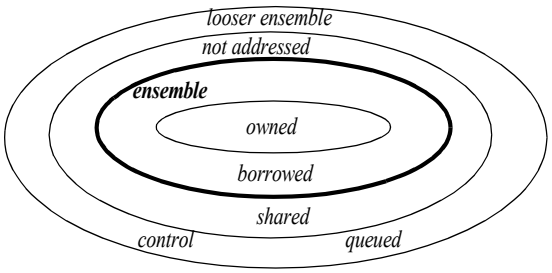

Figure 2: Ownership Model.

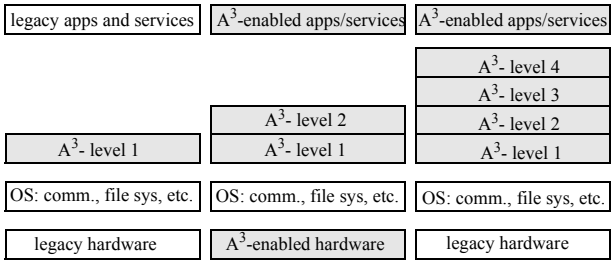

Figure 3: Appliance Aggregation Stacks.

Ownership Models. A user can interact with appliances in many ways. We developed an ownership model to help understand these relationships. Our goal is to define the concept of an ensemble and to determine how a user's ensemble would aggregate with appliances outside of the ensemble. The model specifies several broad categories of ownership and control (see Figure 2).

We identified two kinds of ownership, personal and environmental (organizational). An appliance owned by a person is automatically part of that person's ensemble. It typically contains personal information that needs to be protected from others. A user goes through some registration process to become the owner of an appliance. An environment appliance is publicly available. It is assumed that any information on an environment appliance is suitable for any person present in the environment. Examples are printers, projectors, and mounted cameras.

We specify five ways that a user can relate to an appliance. *Own* and *borrow* apply to personal appliances, and *share, control,* and *queue* apply to environment appliances.

- *Own*—If a user owns an appliance, the user has complete control over the appliance.
- *Borrow*—A user can borrow an appliance and control it as his or her own, except that the owner's data is protected and the user's data is purged upon return.
- *Share*—Some appliances can be shared in parallel or timeshared, e.g., a hotspot and a storage server.
- *Control*—Some appliances can be sequentially controlled. Examples include a projector and a camera.
- *Queue*—Some appliances can receive a queue of inputs. The principle example is a printer.

A user's ensemble is defined as appliances owned or borrowed by the user. To simplify the model further, shared appliances were removed from consideration. Appliances that support control or queuing cannot be part of an ensemble, and represent a looser form of aggregation.

User Models. An important aspect of appliance aggregation is to provide a user model that is as simple, if not simpler, than single devices. An ensemble of appliances attempts to give the illusion of acting as one appliance. While this illusion is a valuable one for the user as far as complexity is concerned, it can present a challenge.

- *Transparency should not be absolute.* For example, hiding details about the location of files from the user adds location transparency. However, this will be confusing to the user if arbitrary files seem to disappear/reappear depending upon which devices are currently present in the ensemble. Thus, unless all files can be replicated or logical error messages can be issued ("Sorry, file was located on PDA1"), then transparency will add more frustration than benefits.
- *Owning a device and joining an ensemble must be simple.* Before a device can join the ensemble it needs to be "blessed," causing it to be marked as owned. This must be simple, yet secure enough that a shop may do it or that a user can "bless" a device at home or at work. Likewise, devices must join the ensemble in a seamless fashion. For example, the simplest device may just have one button and a light to indicate joining the ensemble. Pressing buttons simultaneously on multiple owned devices makes them into an ensemble, with the light indicating success.
- *There must be a uniform view of personal and environment devices in the ensemble.* Users do not want to differentiate the network type of appliances, or the protocol used to access them, unless needed. Therefore, a shared projector should be usable the same way as personal devices. Limitations will exist in the level of functionality supported by some devices, for example, a user cannot own a shared projector.

5 Architecture

Appliance aggregation is executed on top of the existing appliance operating system (e.g., Linux or Windows Pocket PC), and in particular on top of its communication stack and file system, upon which it builds (see

Figure 3). The architecture is peer-to-peer rather than client-server, although there will be some centralized state, such as owner private certificates. Users can differentiate between different levels of A^3, marking products with differing levels of functionality. If a user buys a level-1 device and a level-2 device, the user should expect them to work together with level-1 functionality (the lowest common denominator). While A^3 can execute on legacy hardware, it can also take advantage of A^3-enabled hardware, e.g., in support of improved security. Similarly, while A^3 can support legacy applications and services, it can also take advantage of A^3-enabled applications and services, e.g., for improved caching, security, or performance.

The rest of this section defines A^3 terms, functional requirements, design principles, and components.

5.1 Definition of Terms

Within the scope of appliance aggregation, we define the following terms:

- *Appliance* is a device (communicating on a network) capable of aggregating with other devices; a smallest unit of aggregation.
  - examples include camera, PDA, laptop, watch, and hotspot
  - non-examples include communicationless, shared, and queued devices
- *Ensemble* is a group of appliances aggregated to perform a function greater than their parts.
  - examples include a camera, a PDA, and a watch used in concert to capture, annotate, and communicate pictures
  - non-examples include a client-server relationship, such as using a home PC to access Yahoo!
- *Appliance aggregation* represents an illusion of multiple appliances operating as a single entity for a period of time.
  - examples include use the screen of another device for display; execute a single application on/from multiple appliances
  - non-examples include execute a distributed algorithm on a workstation cluster; send a presentation to a printer

Even though we classify the traditional use of a printer as a non-example for appliance aggregation, there is nothing to prevent a looser form of printer aggregation, e.g., when there is a need to print over a longer period of time.

5.2 Functional Requirements

Appliance aggregation poses the following functional requirements:

- Discover, communicate with, identify, and qualify appliances.
- Establish a secure communication channel between appliances.
- Establish trust boundaries between two or more appliances.
- Provide access to aggregated resources by creating bindings between:
  a) user appliances,
  b) user and environmental appliances, and
  c) environmental appliances.
- Present a user with the illusion of a "bigger" appliance, consisting of aggregated resources from individual appliances.
- Partially and/or fully recover from temporary and permanent failure of a subset of aggregated appliances.

5.3 Design Principles

We define the following design principles for appliance aggregations:

1. An appliance is either in or out of an ensemble.
2. All appliances inside an ensemble are owned or borrowed.
3. A person can have multiple ensembles (e.g., personal and stereo).
4. An ensemble can be owned or not owned (can be borrowed).
5. Ensembles are flat for now (not hierarchical).
6. Inter-ensemble interactions are based on webservices.
7. Intra-ensemble interactions are based on secure communication, data synchronization, and aggregated behavior.
8. An ensemble as a whole has its own access policy.
9. Four levels of functionality are defined (see Section 4).
10. There is a nested set of APIs for increasing aggregation levels.

5.4 High-Level Architecture

This section describes our preliminary appliance aggregation architecture (see Figure 4).

Level-1 functionality relates to: identification, ownership, and authentication; discovery; admission control; membership; and security.

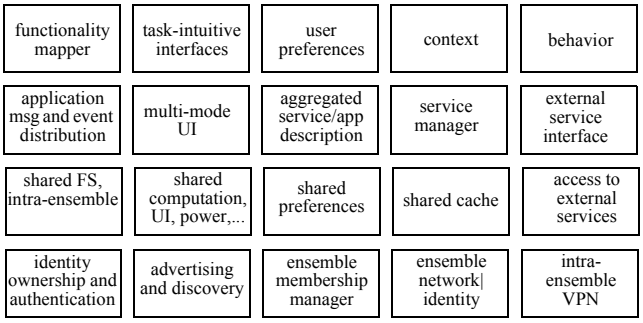

Figure 4: A³ High Level Architecture. *Levels starting bottom up consist of L1: identity and membership, L2: shared resource management, L3: shared application, and L4: shared functionality.*

Appliance identification and ownership. An appliance is uniquely identified by its producer. The identification is a certificate containing the brand, model and serial number issued by the manufacturer. An ownership mechanism at the hardware level signs data so that data originating from the given appliance is trusted. The identification information is immutable. A single user owns an appliance, but the owner may change over time. The owner's identification is registered in the appliance itself. Only the manufacturer or the current owner may overwrite this information. The appliance refuses to operate until its user is authenticated.

Appliance advertising and discovery. An appliance advertises itself when one of its network links come up by broadcasting its identity along with its owner's identity (certificates). If the owner is logged onto a nearby appliance, the appliance will attempt to enroll the newly connected appliance into its ensemble. Conversely, when a user logs onto an appliance, a multicast message is issued with its user identity and the neighboring appliances that the user owns would respond with their identity. The end user need not worry about network addresses and topology.

Ensemble management (admission control). At level-1, an ensemble is connected by the set of secure communication channels (VPN) between the appliances. An appliance is authorized to join the ensemble as soon as it establishes a secure communication channel with the appliance where the user is logged on. A membership protocol notifies ensemble members about changes in the ensemble population in real time. Optionally, appliances leaving the ensemble are detected when a communication failure occurs.

Secure communication channels. Establishing secure communication channels relies on signed messages. When two appliances communicate for the first time, they exchange a secret key. To ensure the link is established between an appliance and its owner, the remote appliance issues a symmetric key that it encrypts with the owner's public key and signs it with the device's private key. At the other end, only the owner can decrypt the key and authenticate the appliance. Extending the model for users to borrow a device requires a list of the authorized users signed by the owner. Each element of this list contains information similar to ownership identity, signed by an authorized user.

Level-2 functionality comprises the ability to share state and resources across the ensemble. Examples of shared state include files, cached data, browser bookmarks, calendars, and preferences.

Shared file system. Much level-2 functionality can be based on a shared file system. The file system provides ensemble members access to shared data and metadata. Special "device files" provide a hook for resource sharing, permitting one device to access the resource of another. The file system's semantics enforce an appropriate level of consistency for the shared data.

Shared computation, UI, and power. Device files in the shared file system provide the interface to access resources from other devices. A device file may be an RDF description of a resource with information about acquiring and using the resource. Sharing a user interface could mean borrowing a keyboard resource from another device, or displaying a given UI on different displays in the ensemble. Power can be shared by distributing computation and networking to optimize power usage.

Shared state and preferences. Preferences for each device are stored in the file system. When a new device is introduced into the ensemble, it can examine the current preferences of other devices, and attempt to choose preferences that apply to itself. For example, if a new PDA joins the ensemble, it may inherit the preferences of another PDA.

Shared cache. Cached data (e.g., browser information) can be stored in the shared file system, allowing ensemble members to share it. The ability to combine storage from different devices into a single directory (i.e., a "union" mount) could increase the size of the cache as the size of the ensemble increases, and eliminate redundant cached data within the ensemble.

Access to external services. If one device has access to an external service, other devices in the ensemble will be able access the service—preferably sharing authentication and authorization information with the first device. In some instances, the first device to access a service may act as a proxy for other ensemble members.

Level-3 functionality presents the user with the illusion that the entire ensemble is executing a single application. It consists of intra-ensemble communication, a shared user interface, service aggregation, service execution, and an aggregated external interface.

Application message and event distribution. An A^3 application is inherently modularized in order to execute in a distributed way on an ensemble. For these parts to communicate, some form of event propagation is required. The A^3 events have a limited range because they are only sent to ensemble members. These events might contain information or they are simply used to invalidate cached information on other appliances.

Multi-mode UI. To present the illusion of coherent execution across the ensemble, the user needs to interact with the application from a variety of appliances. Thus, the UI of the application has to support distribution and coordination. For instance, the user may assign a specific function to each appliance within an ensemble. User interface events from each single-purpose device have to be coordinated with other appliances to construct aggregated service requests.

Aggregated application description. An aggregated application is comprised of the collection of appliances in an ensemble. For example, a hi-fi functionality emerges by putting together speakers and a tuner. Conversely, an application is modularized so it can be executed on an ensemble.

Service manager. The service manager is responsible for balancing load and capability as an application is executed on an ensemble of appliances.

External service interface. An ensemble running an application presents an aggregated external interface so that global discovery and service composition schemes can use the ensemble as a unit.

Level-4 deals with the automatic aggregation of services to help a user accomplish a high-level task. To achieve this goal, we take into consideration a user's preferences and history of past behavior, and surrounding contextual information.

Functionality Mapper. We describe the functionality provided by an appliance as an <*action, data*> pair by extending the notion of MIME-types. To facilitate the automatic aggregation of functionality, we also describe the requirements of an appliance in the same format. Our runtime infrastructure can then use functionality descriptions to aggregate appliances and services to automatically meet the high-level task request.

Task-Intuitive User Interfaces. Because a user can accomplish a task with different appliances, not all user interfaces will be appropriate in each situation. At certain times, it would be better to use voice, while at other times it would be better to use a traditional keyboard, specialized buttons, or a user's intent. The purpose of this module is to select the best available UI for the task.

User Preferences. Mobility can bring several challenges to a user wishing to aggregate borrowed appliances in an unfamiliar environment. Increasing the number of appliances in an ensemble contributes to the combinatorial growth of the number of alternative able to perform a task. To help the user choose among these alternatives, we propose eliciting preferences from the user about appliance properties, and then helping the user select the optimal aggregation by scoring the aggregations according to the preferences. Because acquiring preference can be complex, we propose using declarative policies to encode a group of preferences together. Preference declaration can be provided by the system or defined by developers, administrators, and users.

Context. Context includes any information that can be used to characterize an ensemble environment and that can affect the users preferences. Examples include the user task, the ensemble devices, the user location, and the time of day. Context can be captured by using recent advances in sensor technology.

Behavior. The history of a user past choices, combined with the present context can yield significant information about the user preferences for appliance properties. If the user is not willing to specify preferences for some task, the preferences can be predicted by applying a classifier based upon the historical data and the context.

6 Prototype Implementations

We have implemented prototypes of each aggregation level.

Level-1. The goal of this prototype is to demonstrate the formation of ensembles based on appliance ownership. Each appliance has a pre-stored identity and ownership certificate. Discovery is implemented using the SLP protocol and a distinct scope for each ensemble. Visualizing the ensemble is performed through off-the-shelf web browsers, configured to operate through a local proxy. This proxy is responsible for gathering the ensemble members and establishing secure HTTP links with other appliances. This same proxy also acts as a web server for each device and allows each ensemble member to browse the appliance file space through the secure communication channel. When opening the web browser, the

owner authenticates his own files. A web page displays ensemble members and appliance descriptions.

Level-2. Our initial prototype, the Appliance Aggregation File System (AAFS), is based on NFSv2 (rfc1813) combined with namespace conventions, dynamic group configuration, and a leasing protocol. We use an AFS-style namespace where files are referenced via a specific device name ("cell" in AFS), for example, /aafs/dansJornada/somefile or /aafs/plasma/dev/display. We rely on synchronous I/O for a couple of reasons. Because appliances can leave a group without notice, it is important for applications to detect I/O failures and respond appropriately. Asynchronous writes make failure detection difficult or impossible. For asynchronous operations to be effective, there must be sufficient buffer space to store a significant amount of written data, which is not available on resource-constrained devices. Because a server may also leave and possibly never return, a client must be able to interrupt operations and unmount non-responsive servers. In a standard NFS configuration, with asynchronous operations, "soft" mounts and interruptible operations can lead to data corruption. Synchronous operations eliminate this possibility for properly written applications. Even though the operations are synchronous, they can either be interrupted or timed-out to prevent an appliance from being blocked forever. Leasing is used to address file sharing and it simplifies recovery when devices leave a group.

Level-3. We prototyped an approach to level-3 aggregation that we call Agile web services (see [17] for more details). An Agile web service is similar to a normal web service except that it executes locally on the appliance rather than on a globally accessible server. To execute such a service locally, each appliance must contain a runtime environment. Our Agile runtime consists of a proxy server, a cache, a template engine for generating HTML pages, and an action language engine for handling form POSTs. The *cache-template-action* combination provides the runtime components necessary for a classic *model-view-controller* architecture. The proxy enables a local browser to access the Agile service. In our level-3 prototype, we assign a user-interface "role" to each appliance. A display appliance might be assigned to show the on-line status of a user's buddies, for instance. As roles, applications, and application data change, the cache notification mechanism lets the appliance know when to regenerate page views. This way we distribute control without customizing inter-appliance communication. To demonstrate this idea, we created an application that allows people to share music while using text messaging to chat. We created a number of roles, such as a buddy list showing who is present, and a room list showing chat rooms in which the user is participating.

Level-4. We prototyped the CAFE system that provides user-centric aggregation of device functionality in a dynamic environment. The system automatically aggregates appliances based on existing description of appliances and user requests. CAFE facilitates the selection of the aggregation that best matches a user's preferences using declarative policies. In addition, it allows a user to express trade-offs between the quality of device attributes, user distraction, and aggregation stability. A policy suggestion mechanism is employed to recommend policies to the user based on the context and the user's aggregation history. The system has five main components: a Service Lookup Protocol provider for appliance presence and functionality registration, a user interface, an aggregator, a candidate selector, and a policy recommender. All five components run on a designated appliance, called the coordinator. A user interacts with the system using the servlet-based user interface. A user's request is represented as an <*action, data*> directive, e.g., <*play, MPEG*>. The aggregator calculates all possible aggregations for a given user task using a rule-based engine. The system provides the user with the ability to specify preferences for devices and device properties. These preferences are used to guide the selection of the best aggregation. If the user does not input preferences, the policy recommender module predicts preferences based on the user's past interactions with the system. User preferences are used by the candidate selector module to rank the aggregations. A more detailed discussion can be found in [8].

7 Benefits of Appliance Aggregation

Appliance aggregation presents several key benefits over standalone or proprietary appliances. It enables better products, more opportunities for innovation, as well as traditional benefits of an architecture.

Better products

- *Ease of use.* By reducing the barriers between devices they will be easier to use. It will be possible to access functionality and data arbitrarily and securely.
- *Better security.* It will be significantly easier to deploy and to enforce security with a clearly defined aggregation and use model. A^3 can enforce better protection of user data scattered over many devices. It can establish more robust ownership models.
- *Increased interoperability and personalization.* The aggregation of devices opens up opportunities for better matching of functionality to personal needs. Rather

than having a combined device, two more ergonomical and/or smaller devices may be dynamically composed for the precise task at hand. For example, a cellular radio, handset, and PDA can be combined to make a phone call.

Innovation

- *New combinations of appliances.* Add-ons from vendors and third parties can be used to combine devices in novel, useful, and interesting ways. For example, new software can make it possible to control the room air conditioning from an appliance.
- *New appliances.* By making it possible to dynamically combine appliances, new opportunities will present themselves for improved appliance designs that are not hampered by power supplies, I/O, and user models.
- *New applications, distributed models, and functionality.* New models of appliance interaction will enable new applications to emerge with new distributed models and new functionality.

Benefits of a common architecture

- *Leverage industry through open standards* (third party developers, tools, applications, and so forth). By having an open architecture, others can contribute in areas where individual companies are not strong or do not have an interest to lead, enabling others to innovate and contribute, e.g., in case of low volume production or specific markets.
- *Enable platforms on top of architecture.* Ensembles of appliances present opportunities for a programming environment platform supporting ensembles.
- *Easier creation of brands, control points, and ecosystems.* These are traditional benefits of any architecture.

An example that might present a beachhead business opportunity is authenticated appliance use of expensive mobile appliances in the enterprise market. Security is a prime concern in these markets both for data access and theft. Level 1 of our architecture provides support for authenticated single sign-on use, which can be invaluable for three reasons. First, without the ensemble password, no device can be used when stolen (e.g. laptops). Second, with the insecurity of wireless networks, some form of shared log-on can form the basis for network security (e.g. VPN) solutions. Third, past examples have shown (802.11b networking, high-end PDAs) that getting wide ranging solutions into the corporate market can eventually lead to wider adoption.

One concern is the cost of adding the functionality required for aggregation to appliances. Appliance aggregation requires at least one form of network support in each device as well as a software stack to support it. Generally, devices are moving towards becoming networked, so this cost is not exclusive to aggregation. In addition, by breaking down the software architecture into self-contained levels, cheap or simple devices need only support lower levels of the architecture. For example, pen appliances need only support level 1 of the architecture, because they do not need a file system.

8 Related Work

Several efforts have addressed making IT appliances work better together. Sony's MemoryStick, the Apple-initiated IEEE 1394 standard, and home networking standards such as HomeRF and PowerLine have all provided some basis for connectivity among consumer electronic devices. Infrared has been used for similar purposes between PDAs, laptops, and printers; and is now being replaced by Bluetooth. HP successfully introduced interoperability between measurement devices and promoted it as the IEEE 488 (HP IB) standard. These standards all share a fairly low-level view of appliance integration, supporting file transfer, and networking or wireless cable replacements. Lack of broad industrial adoption of these standards has resulted in market fragmentation, as exemplified by IEEE 1394 vs. USB, memory stick vs. flash memory, and interference in the wireless communication standards (802.11b and cordless phones).

UPnP [19] and RendezVous [15] are a standard and an emerging de facto standard at the networking level that enable personal computers and appliances to plug into the network, configure its addresses, and announce their presence and services. Above the networking level, a few systems have tried to provide some coordination of resources or functionality. The most notable examples are the Sony HAVi and Sun Jini. These systems provide support for remote control and coordination, but they do not couple the devices tightly together. Instead, they provide functionality similar to a traditional distributed system (remote object invocation, naming, and so forth).

The benefits of aggregation have been recognized in Grid Computing [6, 7] at the infrastructure level with respect to aggregation of processing, storage, and virtual wiring, as well as in a number of pervasive computing efforts [5, 13, 16, 18, 20] where data, I/O, and computing are distributed across many personal and environmental computers.

Epson and camera producers are trying to make a best match between printers and cameras in so-called Print Image Matching (PIM) [14]. IBM's Digital Jewelry [10]

envisions computers deployed on a user's body and connected in a personal area network. They require some form of ensemble support and advanced UI, similar to our proposal. There were a number of architectural efforts in the past [9, 11, 12].

A^3 is orthogonal to the super-appliance approach (integrated PDA, phone, GPS, and so forth) because even super-appliances can be aggregated with environment appliances for additional functionality and performance. It is important to point out that A^3 functionality need not primarily come in the form of software architecture, a lot of components can be deployed in hardware, firmware or downloadable drivers. It will be up to individual companies to choose the most suitable form of deployment.

9 Summary and Future Work

In this paper we presented a case for an appliance aggregation architecture (A^3) by motivating the needs for such an architecture, and by discussing possible architectural levels, ownership models, and user models. Then, we presented a four-level architecture and preliminary prototype implementations. Our work is an attempt to provide higher-level functionality, to allow increased cooperation between appliances. In addition we provide some extra functionality, such as multiple levels of security and single sign-on. Together these provide for closer and richer interactions between appliances in an ensemble.

After our initial designs and prototyping, we took this work to the GGF, where it continued as a part of the GGF Appliance Aggregation Research Group (APPAGG) in the P2P technical area. It is an attempt to bring together various companies in the appliance space to come up with a common strategy enabling appliance aggregation [4].

ACKNOWLEDGMENTS

We are indebted to Tim Kindberg for providing feedback on the initial versions of this paper. His comments significantly improved the content and presentation of the paper. Christos Karamanolis, Ed Perry, and Rosanne Wyleczuk contributed to the earlier phases of this effort.

REFERENCES

[4] Bhatia, et al. "Appliance Aggregation Architecture Terminology, Survey, and Scenarios", A GGF APPAGG Research Group Working Document, www.hpl.hp.com/hosted/ggf/AppAggSurveyApr03.doc.

[5] Dertouzos, M.L., "The future of computing," Scientific American, July 1999. oxygen.lcs.mit.edu.

[6] Foster, I. and Kesselman, C., "The Grid, Blueprint for a New Computing Infrastructure", Morgan Kaufmann, 1999.

[7] Foster, I., Kesselman, C., Nick, J., Tuecke, S., "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration," Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002.

[8] Kumar, R., et al. "User-Centric Appliance Aggregation," HPL Technical Report, HPL 2002-277, September 2002.

[9] IEEE Recommended Practice for Architectural Description of Software-Intensive Systems.

[10] Miner, C., "Pushing Functionality into Even Smaller Devices, "CACM, Volume 44, Number 3 (2001), Pages 72-73.

[11] Morris, C., and Ferguson, C., "How Architecture Wins Technology Wars," Harvard Business Review, 1993

[12] Nokia, "MITA – Mobile Internet Technical Architecture", Nokia White Paper.

[13] Norman, D. A., "The Invisible Computer," Cambridge, MA, MIT Press, 1998.

[14] P.I.M., Print Image Matching, www.printimagematching.com/

[15] RendezVous, http://www.apple.com/macosx/jaguar/rendezvous.html

[16] Satyanarayanan, M. 2001., "Pervasive Computing: Vision and Challenges," IEEE Personal Communications, August 2001.

[17] Rowson, et al., "Agile Computing," HPl Technical Report.

[18] Thackara J., "The Design Challenge of Pervasive Computing," Interactions, vol 8, no 3, May/June 2001.

[19] UPNP, http://www.upnp.com/.

[20] Weiser, M., "Some Computer Science Problems in Ubiquitous Computing," CACM, July 1993, 75-8.