Enabling Pervasive Collaboration with Platform Composition

Trevor Pering, Roy Want, Barbara Rosario, Shivani Sud, Kent Lyons
Intel Research Santa Clara
{trevor.pering, roy.want, barbara.rosario, shivani.a.sud, kent.lyons}@intel.com

Abstract. Emerging pervasive computing technologies present many opportunities to aid ad-hoc collocated group collaboration. To better understand ad-hoc collaboration using pervasive technologies, or *Pervasive Collaboration*, a design space composed of three axes (composition granularity, sharing models, and resource references) is outlined, highlighting areas that are only partially covered by existing systems. Addressing some of these gaps, *Platform Composition* is a technique designed to overcome the usability limitations of small mobile devices and facilitate group activities in ad-hoc environments by enabling users to run legacy applications on a collection mobile devices. The associated *Composition Framework* prototype demonstrates a concrete implementation that explores the applicability of existing technologies, protocols, and applications to this model. Overall, Platform Composition promises to be an effective technique for supporting collaborative work on mobile devices, without requiring significant changes to the underlying computer platform or end-user applications.

Keywords. Pervasive technologies, ad-hoc collaboration, mobile devices, resource sharing, platform composition.

1. INTRODUCTION

Collocated collaboration is a highly dynamic and social activity where groups of people share information and create shared artifacts. *Pervasive Collaboration* enables users to share information using highly mobile and capable pervasive computing technologies such as smart phones, Mobile Internet Devices (MIDs), laptops, and large-screen displays. For example, the wealth of devices commonly found in corporate meeting environments could be used to enable more interactive group presentations. Studies of specific collaborative applications such as gaming [33], photo-sharing [4], and general file-sharing [35], detail how such platforms enable new collocated experiences for specific applications. However, in order to support a wide range of devices and generalized applications, collaborative middleware solutions have spanned a diverse design space consisting of collaboration granularity, sharing models, and resource referencing, leading to a number of different supporting middleware solutions.

Platform Composition is a technique that integrates standard computing components to support effective collaborative work by wirelessly combining the most suitable set of resources available on nearby devices. It is particularly well suited for ad-hoc tasks on wireless mobile devices such as laptops and advanced mobile phones: seamlessly incorporating fixed infrastructure such as projection displays. While these devices are becoming pervasive, their small form factors often impose usability challenges for supporting collocated group work. Composing devices enables them to act as a uni-

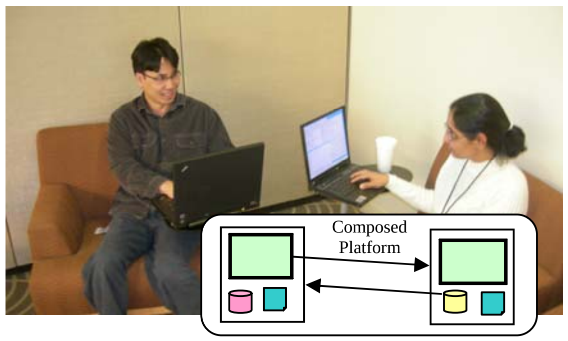

Figure 1: A Platform Composition showing the logically composed platform created out of the display and storage services from each user's individual device.

fied platform, enabling users to more easily overcome their individual limitations. The *Platform Composition* concept specifically refers to connecting devices together using standard distributed network protocols in such a way that existing familiar applications can be run unmodified. For example, it could be used to combine the file shares and displays of two laptops in order to run a standard application like Photoshop, enabling two users to more easily collaborate. This concept builds on the general notion of composition tailorability [34], but focuses on adapting the supporting computing platform, instead of changes to the end-user applications.

In order to understand how commonly available platforms and services support the general concept, the *Composition Framework* prototype provides a specific implementation of Platform Composition along with the user interface necessary to invoke standard platform services. By design, it is orchestrated around utilizing existing standards to support familiar applications on ad-hoc sets of devices. Although the underlying services and protocols used to share data among devices are not themselves new, the system instead focuses on the centralization and coordination of the sharing process. Experience with the Composition Framework highlights the efficacy of Platform Composition and informs a discussion of how existing systems can be modified to better support mobile ad-hoc collaboration.

The key contribution of this work is a focused understanding of how pervasive technologies can be used to support small-group collocated collaborations. The *Composition Framework* is the implementation of the *Platform Composition* concept, which is designed to support *Pervasive Collaboration*. Each of these three aspects provides an individual contribution to understanding pervasive collaboration applications:

1. *The Pervasive Collaboration Design Space* gives a concrete structure within which to understand how different pervasive technologies support collaboration.
2. *The Platform Composition concept* addresses several gaps in the pervasive collaboration design space by allowing the creation of dynamic ad-hoc collaborative device ensembles that support interaction using familiar applications.

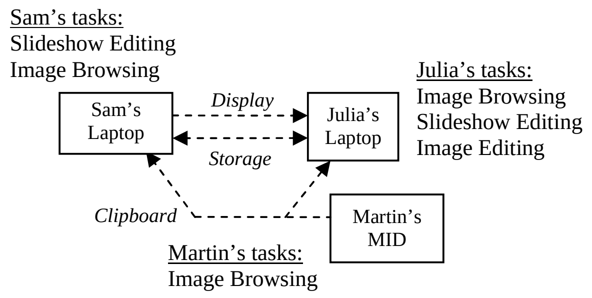

Figure 2: Example sharing setup for the introductory scenario, showing how display, storage, and clipboard can be shared among mobile devices to form a composite collaborative working environment utilizing a variety of applications.

3. *The Composition Framework prototype* highlights how existing services can be adapted to support collaboration as well as provides a system from which to collect observations on how users relate to the system.

Motivating Scenario

In the following detailed scenario, depicted in Figure 1, motivates Platform Composition by describing how a few friends meet in a café to create a birthday slideshow for their friend Kim:

Sam arrives first and starts to create a slideshow on his laptop adding some text and using some images he has on his system. By the time Julia arrives, he has a basic slideshow put together. To see his work, Julia mirrors Sam's display onto her laptop. She immediately thinks of a couple of funny captions she wants to include and adds the text while Sam discusses the images he selected.

Julia mentions that on the bus ride over she had spent some time browsing through her photo collection and found a couple of additional images to include. She shares her file collection with Sam's computer and uses the shared display to show him the specific images she wants. Sam nimbly drags and drops the pictures into the document from the shared folder.

At that point, Martin, a mutual friend, enters the café and stops by to say 'Hello.' He sees them working on the slideshow and realizes he has a great image in his email they could use. Using his Mobile Internet Device, he browses though his inbox, to find the image. They connect their clipboards together, allowing them to copy/paste the images between devices, both into the document Sam is working on and into Julia's personal collection.

Sam and Julia continue working on the document as Martin goes off to order his coffee. Julia feels that some of the images need a little editing: Sam makes his file share available to her, and she uses an image editing application on her machine to touch-up the photos. All done, they put their systems away and head off to Kim's birthday party, implicitly disconnecting their shared services.

In this scenario, Sam, Julia and Martin compose their systems to create a logical aggregate platform using several different platform-level services, highlighted in Figure 2. They dynamically shift between sharing their display, file storage, and clipboard between their computers in order to accomplish their task as a group. Furthermore, they use a collection of standard applications to edit the main document, share personal content, and edit individual images using different resources from all three individuals' devices.

Currently, such interactions are problematic due to the numerous steps needed to share distributed content and difficulties resulting from interacting in mobile environments. For example, one commonly employed solution is using email to exchange images between users, even though they may be sitting right next to each other. Email requires people to *a priori* decide which specific images to share and incurs an interaction overhead through an external (to the task) email application, and routes all information through supporting infrastructure.

Alternately, collaborative sharing can be moved into the physical world, either through the motion of people or devices. Two people can share one display, but it can be difficult for multiple people to crowd around one screen. Similarly, sharing a USB memory stick between two computers would limit data sharing to a batch transfer of files, preventing any dynamic sharing of data between devices. As a result, these approaches are functional but do not necessarily offer the best collaborative experience.

THE PERVASIVE COLLABORATION DESIGN SPACE

This section highlights three system design issues (Figure 3) that characterize mobile collaboration systems. First, *composition granularity* impacts the nature of collaboration spaces formed by users and the steps they must take to integrate with legacy applications. Second, *sharing models* governs how users interact with the underlying data and how shared resources are managed. Third, *resource referencing* refers to the different ways resources can be named in a system, underscoring both the infrastructural needs of the system and mechanisms by which users relate to technological constructs. A clear understanding of these various design options is useful to understand how current systems exploit various aspects of pervasive technologies.

Composition Granularity

Within the context of modern operating systems, resources can be shared between systems at a number of different granularities. There are three primary levels of granularity which can be used to support the interactions between systems:

Events: Fine-grain sharing can be accomplished through events, which take on the form of small, individual occurrences that stream together to form higher-level actions. User interface events, such as mouse movement or button presses are prototypical examples of event level sharing: each individual event is very small and short lived, but by combining multiple events together a cohesive stream of actions can be formed to interact with a remote system. The iRoom [13] system utilizes event level sharing in a collaborative room environment. It employs a centralized infrastructure for sharing events among mobile devices and fixed infrastructure to create a unified distributed work environment.

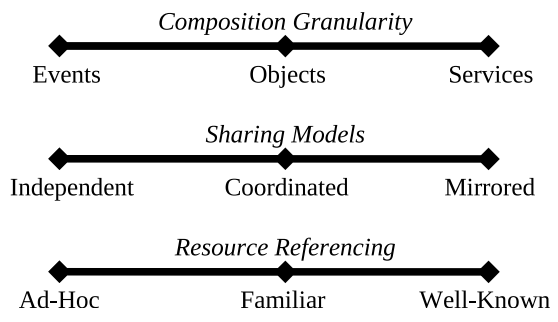

Figure 3: Composition design spaces. Although representing distinct axes, each spectrum follows the general pattern ranging from small, individual elements to more encompassing coordinated constructs.

Objects: Medium-grain sharing relies upon the sharing of individual objects between systems. These objects represent persistent individually meaningful entities that can be acted upon in a variety of ways. Unlike event sharing, objects are persistent, and represent more data than is directly transferred. Casca [6] supports object-level sharing and enables users to create shared "converspaces" in which objects such as files and printers are placed to be made available to other users; i-Land [32] similarly allows objects to be managed within a collaborative space with many display surfaces. Multibrowsing [14] exposes web pages as objects layered on top of the iRoom event mechanism: persistent web objects are encoded as events, which incorporate transient properties such as the target display screen.

Services: Coarse-grain sharing is accomplished through sharing entire services from the platform, which are integrated with the underlying operating system to provide transparent application access to the associated resources. For example, access to a network file share represents a collection of file objects available to an application in a transparent manner. Like object sharing, service sharing has the quality of persistence; however, it is less specific in the semantics about what is being shared and more flexible in terms of legacy application support. Sharing services is currently possible piecemeal: users can use basic utilities to share services between their devices, but there is no coordinated application for this capability.

Collaboration systems such as the iRoom, Casca, and Platform Composition each provide the user with a different primary mechanism for sharing, encompassing a trade-off between fine-grain control using events and simpler construction using coarse-grain sharing. The iRoom, for example, makes it very easy to remote control systems with keyboard and mouse events, but will incur an extra step in order to share a file directory between systems. Casca makes sharing collections of files among users easy, but would require extra mechanisms to share keyboard and mouse events between systems. Platform Composition, on the other hand, provides transparent access to remote system level services, but requires user interaction in order to share events or objects between systems. Composition granularity, therefore, becomes a primary differentiator between these systems.

Application Sharing Models

Similar to the different levels in which sharing can be induced between systems, individual collaborative episodes can be managed on different levels [2]. These levels are similar to the composition granularity described in the previous section, but manifests themselves more directly in users' work practice in terms of mechanisms for conflict resolution and data synchronization. Application sharing models can be broken down into three levels:

Independent: An application can share data independently of other systems in such a way that changes are independently applied to each system and loosely propagated to other instances. For example, one implementation of a calendaring application might handle meeting requests between users but operate on different underlying databases (one per user). As a result, the instance of one appointment will not be inexorably linked to others. Independent sharing is supported by many variants of distributed single-user collaboration-transparent systems, such as email or personal calendars. Independent sharing offers no direct mechanism for conflict resolution, since there are two separate entities that only loosely correlate – relying on higher-level resolution mechanisms.

Coordinated: Sharing multiple views of the same underlying data enables coordinated sharing. Using the same calendaring example, the application could open up multiple views of a shared calendar, such that all views show the same underlying data but from different perspectives (e.g., maybe showing different days). Coordinated sharing allows multiple people flexible access to shared data, allowing shared context with independent interaction; conflict resolution is a key aspect of coordinated sharing, but is often handled on an application-specific basis. Coordinated sharing is supported by a wealth of *collaboration-aware* [17] systems, exemplified by projects such as the Coda filesystem [28] and TeamRooms [27].

Mirrored: Finally, applications can be shared by exactly duplicating all aspects of presentation, from the underlying data set to the visual representation and interaction. This level of sharing enables users to share interaction context while working on a common dataset. For example in a calendar application supporting mirroring, users could be using the mouse to point to a specific day and say "How about we schedule a meeting here?" Here, the system sharing is supplying the necessary related context. This interaction allows multiple people to coordinate directly, decreasing the interaction overhead but preventing independent operations. This level of application interaction is supported by services such as VNC [26] that support remote-desktop collaboration.

A challenge for collaborative systems is to provide support across multiple application sharing models [2], essentially enabling users to use the appropriate model at different stages of their overall task, exposing a trade-off between flexibility in manipulation through independent sharing with an emphasis on communication for mirrored sharing. For example, in a calendaring application, users may wish to initially work with *independent* systems to understand personal commitments, use a *mirrored* view to discuss potential scheduling options, and then a *coordinated* conclusion to record the group consensus. Furthermore, it is important for a system to reflect the mode of sharing to the user, so that they do not form incorrect mental models of the underlying system. Towards this end, the different services made available through Platform Composition provide a unified approach to encompassing multiple sharing models, and provide the user with the ability for independent, coordinated, and mirrored sharing.

System Resource Referencing

Individual resources, such as devices and services, can employ a variety of means to discover, address, and reference each other in distributed systems. Referencing is important because it directly effects how users interact with the resources made available by the system. There are three rough models for resource referencing:

Ad-Hoc: Systems which have never seen each other before can use *ad-hoc* mechanisms to identify resources. Conversationally, this translates to "use this device" where a user can select the device from a set of nearby devices, discovered by such techniques as UPnP, ZeroConf, or Bluetooth wireless scan. Ad-hoc references are advantageous in that they allow access without requiring prior setup, allowing for the *implicit* discovery of new resources, although it can suffer from confusion if there are many similar resources available.

Familiar: Referring to something as "the one I used before" invokes a *familiar* reference, which has been used in the past but is not necessarily well-known. Familiar references are advantageous in that they provide a mechanism to resolve ambiguity and complexity introduced by pure ad-hoc references, or ease *a-priori* set-up required by well-known references. Most systems will start from an ad-hoc or well-known foundation and construct familiar references using techniques such as bookmaking or machine learning as ways to mitigate the associated disadvantages.

Well-Known: Referring to something by a specific name, such as "fred.smith@mail.com," represents a *well-known* reference in that it works unambiguously and consistently in all contexts. The disadvantage of well-known references is that they require *a-priori* setup when they are first used and can be difficult to invoke in dynamic ad-hoc situations; in contrast to ad-hoc systems, well-known references require *explicit* discovery of new resources. Well-known references are commonly used with email and chat-client programs to send messages between people; also, referring to a specific remote network file-share by machine name or IP address would constitute a well-known reference.

Similar to Casca, Platform Composition itself does not inherently rely on any given reference mechanism, although the dynamics of ad-hoc mobile collaboration fundamentally starts with ad-hoc references. A challenge with both these systems is how to limit the scope of ad-hoc discovery to the most interesting or relevant devices: effectively relying on familiar references. Mobile systems can leverage the physical location or proximity of devices [19] in order to form associations, i.e., the "physical familiarity" of a device. Emerging mobile devices are starting to possess new input capabilities, such as accelerometers, cameras, and Near Field Communication (NFC), that may be useful in aiding mobile device composition. For example, DACS [7], bumping objects together [10], Relate [9], Gesture Connect [24], and Elope [25] are all sensor-based approaches to joining devices together, which could directly mitigate some of the problems associated with ad-hoc or well-known references. The trade-off for referencing is between the ease of accessing a new or unfamiliar object using ad-hoc referencing with a more reliable accessibility for common objects using well-known referencing.

PLATFORM COMPOSITION

In contrast to the other composition systems mentioned above, Platform Composition enables users to interact by easily connecting their existing platforms' services together. For example, as highlighted in the motivating scenario and Figure 2, a file share from one device can be made available to another user's device; alternately, a user could remotely access the display of another user's device, allowing them to jointly view and edit content. This section covers the high-level concept of Platform Composition, covering the motivation and services that are applicable to pervasive collaboration applications.

Motivation

The concept of using Platform Composition to support mobile ad-hoc collaboration is motivated by four main factors: technology advances, end-user benefits, standard services, and available applications.

First, current improvements in mobile device processing, storage, and communication capabilities have created advanced mobile devices that can host significant applications. Mobile laptop computers are now the mainstay of many corporate environments, and the processing in some hand-held devices is capable of supporting high-quality touch-screen interaction; solid-state drives (SSD) and other high-capacity storage technologies provide ample capacity to store extensive caches of digital media; while advanced wireless standards such as ultra-wide band (UWB) and IEEE 802.11n provide the means for high-bandwidth inter-device communication. These trends present an opportunity in bringing these advances to cooperative work. Projects such as The Personal Server [36] and Dynamic Composable Computing [37] have explored how these same trends impact the mobile platforms themselves.

Next, composing systems from several mobile devices has the potential to improve several aspects of the collaborative user experience, addressing some of the fundamental limitations imposed by their small screens and limited I/O capabilities. Increasing the total available screen real-estate has been shown to improve group productivity by using multiple screens together [8], and also affords new opportunities for media consumption [30]. Not only do user's individual devices provide access to their personal resources, but they provide a readily available platform from which to access resources on other devices. The theme of orchestrating multiple devices to support collaboration is common in the research literature, having been addressed by systems such as Casca [6], iRoom [13], Pebbles [21], and The Display Mirror [22], among others.

From an application standpoint, there exists a wide variety of single-user applications that can be utilized for group collaboration, a notion known as *collaboration transparency* [17]. Platform Composition can directly leverage these existing applications, instead of relying upon special-purpose collaborative applications. Many previous systems have focused on collaboration transparency in the context of a single application, e.g. *intelligent collaboration transparency* [16] enables distributed crossplatform text-editing without application modification. In contrast, Platform Composition supports transparency at the *system* level, since it enables users to employ a variety of standard applications to accomplish their task.

Finally, there are several computer industry standards which provide functional abstractions for basic platform services, such as the file system, bit-mapped display, and socket-based networking. These standards provide a consistent programming model across diverse hardware platforms and utilize standard communication protocols, such as TCP/IP, enabling them to operate in distributed environments. Such distributed systems have been a cornerstone of fixed collaborative setups, which allow people to work collectively on distributed resources like desktop computers through shared file systems and remote desktop protocols. Platform Composition capitalizes on this capability by using these system abstractions as the primary building block supporting dynamic collaboration on wireless mobile devices.

Platform Services

Figure 4 shows how platform services are orchestrated by Platform Composition to enable existing applications to work with distributed resources. All these services are common platform components, with well-defined behaviors and control mechanisms: virtually every application can access these resources in a standard and consistent manner. There are three primary system resources supported by the concept:

Clipboard: By sharing the system's clipboard, users can easily transfer isolated pieces of information between systems. For example, a user can easily share the URL of a web page they are looking at by copying it to the clipboard – similarly, they can easily identify and share a specific image. As a standard system resource, the clipboard is closely integrated with virtually every application available on standard windowing systems (although it is not currently pervasive on all small mobile devices). The Remote Clip [20] project explored the impact of clipboard sharing between a personal mobile device and a desktop PC.

Storage: Groups of files, such as an image collection or current working documents, can be made available remotely using standard network file-share techniques. For example, if one user is working on a slideshow, as described in the introductory scenario, they can share their working directory to provide other users access to the files. This allows them to directly work with the content; similarly, a supporting user could make their local image collection available to the main document author enabling them to easily incorporate the images. Once a storage composition is formed, files are accessed through applications using their commonly available file open/save mechanisms. The Sharing Palette [35] and Coda [28] have explored the impact of storage sharing in distributed collaborative environments.

Display: Visual resources can be easily shared by replicating the user's visible display surface: either by remoting, pulling, or extending their display. Remoting their display to an external screen allows others to easily see (and potentially interact) with the primary user's data. Pulling a remote display to the local device enables a remote-control interaction with the other display. While this is fundamentally the same as pushing a display connection it changes who initiates the connection. Extending a system's display to utilize a supporting device enables applications run-

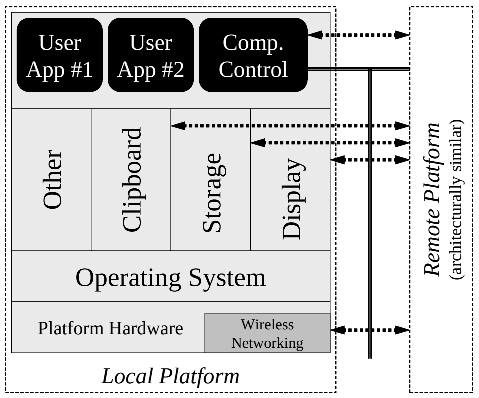

Figure 4: Platform Composition architectural overview. Each symmetric platform is represented as a standard set of services implemented on top of the operating system and platform hardware. From the applications' perspective, the underlying system resources appear the same if they are local or remote.

ning on the host system to transparently utilize additional screen real-estate, much in the same way as would be accomplished by physically attaching a second monitor. Several projects such as The Display Mirror [22], MobiUS, [30], and "The 22 Megapixel Laptop" [31] (among others) have explored the impact of display sharing in a number of different environments.

Additional services, such as the ability to share individual windows provided by Impromptu [3], arguably fit into this model as long as they provide a generic capability that would be available to any application. Sharing input devices, such as keyboards and mice, has also been incorporated into the system by sharing USB devices over the network based on commercially available USB-over-IP solutions. Likewise, other system services, such as dynamic networking, distributed sensing, and processor sharing also fall under the umbrella of Platform Composition, and represent areas for future work.

Supporting Platform Collaboration

In contrast to the examples cited under individual services above, Platform Composition provides an integrated framework from within which to access all these services. Additionally, since these services represent standard system capabilities, they are available to most applications without modification. These services present the user with very familiar mechanisms for sharing data, and they are compatible with virtually every available application, allowing Platform Composition to easily support pervasive collaborative activities. The key aspect of the underlying concept is to share coarse-grain resources in order to support familiar applications, instead of redesigning applications or trying to adapt applications into a highly constrained design space.

In essence, Platform Composition is creating a distributed logical platform that replaces the underlying individual pieces of pervasive computing technology. A modern laptop computer is made up of tightly-coupled storage, display, processing, and I/O

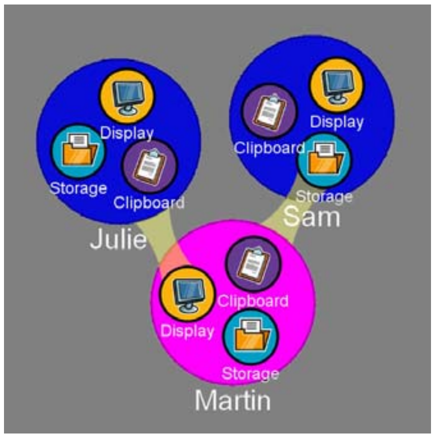

Figure 5: Join-the-dots graphical user interface (GUI) used to manually create compositions by sharing resources with a simple line drawing metaphor. This interface has been designed to work well with touch-screen or pen interfaces, and does not rely heavily on detailed manipulation or textual input.

capabilities, while a distributed composed platform provides these same basic resources, except sourced from a diverse set of devices. The potential downside, of course, is that the distribution process will either become overly confusing or complicated for users, or decrease system effectiveness due to increased communication latencies.

COMPOSITION FRAMEWORK

The *Composition Framework* prototype presents a concrete implementation of the Platform Composition concept in order to better to understand its relationship with higher-level applications, underlying resource sharing services, and the overall user experience. At its core, the Composition Framework is a distributed message passing framework that has modules for user interfaces, device/service discovery, and service integration. The framework exists as a thin middleware layer that is used to orchestrate service connections among devices; however, once services are made available through the operating system, the composition interface does not play a role in using the applications themselves, which are layered directly on the exposed services.

The primary user interface employs a graphical join-the-dots metaphor, depicted in Figure 5. To effect a connection in the system, the user simply draws a line from the core service (small circles) to the target device (larger enclosing circles); likewise, in order to provide the user with feedback on the state of the system active, service connections are represented by directed links between the source service and device. Services can be disconnected by dragging a line across the service (metaphorically "cutting" the connection). Users are also able to invoke various configuration and diagnostic operations through specialized gestures. A system-tray icon is also available, primarily for use on laptop and desktop systems. The graphical interface is similar to others that have supported composition for both objects [5][23] and events [1][15]. Addiontally, the interface provides the ability save and restore predefined compositions, as well as automatically suggesting composition candidates.

The Composition Framework architecture consists of four major components: framework core, user interface, network discovery, and service modules. DBUS, a standard message passing infrastructure, is used to facilitate communication between the modules. The core components are implemented in Java, with various specific components utilizing a number of other languages and interfaces, as required. The system supports both Linux and Windows operating systems, and has been used on a variety of standard computing platforms such as Ultra-Mobile PCs (UMPCs), laptops, desktop systems, and projection displays.

Each individual service for sharing a resource is specified by an XML service descriptor file, which encodes basic properties of the service (name, icon, etc.), and provides details on how to probe, invoke, monitor, and disconnect the service. Some services, such as storage sharing, are implemented using asynchronous operating-system calls, while others, such as display sharing, are implemented by invoking a standalone client process. Services are handled using an explicit client/server model based on commonly available standard systems:

Clipboard sharing is realized using the *synergy* [29] system, which enables clipboard and mouse sharing among a group of systems. This system seamlessly integrates with the system clipboard, providing a virtually transparent mechanism for users to share information.

Storage sharing is implemented using standard SMB-based storage capability built-in to Windows and Linux platforms. Automation of standard command-line interfaces are used to access storage sharing, and the resulting client is presented to the system as an integrated drive or folder mount point.

Display sharing is built on the standard VNC protocol, using the standard VNC protocol [26], supporting multiple client and server implementations. This implementation allows easy access of the display between systems. Furthermore, on Windows platforms, MaxiVista [11] is used to enable extending a display surface across multiple devices.

Peripheral sharing utilizes USB Redirector [12] to share USB devices between devices. For example, speakers attached to one device can be used to play music sourced from another, or a tablet input device wired into a tabletop can be used to augment traditional laptop input modalities.

These implementations each represent specific examples of how existing technologies can be used to implement the associated service, and could be easily replaced by alternate implementations if/when they become available.

Evaluation

The Composition Framework has been tested in both a laboratory based experiment and by the core research team over a period of several months. The laboratory experiment included eighteen participants recruited by convenience sampling from our corporation as well as personal friends. The participants were evenly distributed across gender, and 65% were between 20 and 29 years of age and 35% between 30 and 39. The experiment tested a well-structured sequence of tasks involving compositions between three systems. The extended core usage centered around five researchers using the system in a typical conference room environment. In both cases, the devices used consisted of a combination of laptops, ultra-mobile PCs (UMPCs), and desktop systems (attached to either a large-screen display or projector). These experiences have provided a wide range of users' reactions to the basic concept and allowed insights into the end-user benefits.

Overall, it is clear that users can relate to and understand the basic concept of Platform Composition, and find the Composition Framework user interface intuitive and easy to use. Participants from the user study were able to make use of the unified mechanisms provided by the Composition Framework to manipulate the state of the services involved in a composition. One user commented that the system was "*really easy to understand, learn and use,*" while another said that the GUI was "a *good visualization of what's going on: makes user more comfortable with sharing to know exactly what's shared and if it shared successfully.*"

Routine interactions with groups of up to five people have further revealed insights into the effectiveness of the system for collaborative tasks. The basic ad-hoc discovery mechanism and user interface have shown to be useful in lowering the barrier for initiating multi-device collaborative tasks. Based on this ease of sharing, users were able to export documents for collaboration in group settings, instead of requiring peers to gather around a single monitor or simply "talking to" a relevant document. The available alternative was generally using a standard VGA cable for projection, or emailing the document to all parties involved. In essence, users were able to share and engage their coworkers without as much interference from the underlying technology. One interesting use practice emerged as a result of being able to easily share *two* resources from the unified interface, discussed more fully in the next section.

Several conceptual difficulties with service sharing were observed relating to display and storage sharing. The most pronounced confusion was experienced with display sharing, where users would quickly become confused about which display was shown where – since many computer desktops looks very similar, and the physical separation of screens was no longer effective at differentiation. Similarly, users were somewhat confused at the difference between an underlying shared directory and the on-screen window that represented the share: the mistaken assumption was that closing the window prevented access to the underlying data. Accordingly, users commented "*All the connection lines can cause confusion,*" underscoring some of the difficulty with comprehending resource sharing.

Another deficiency apparent with shared services as they stand is the inability for a user to control a remote client, when they push a local service remotely: e.g., if they share their display, making it available on another computer. The basic problem is that client/server programs typically assume interaction from the client side, and generally don't support server-side control of the client. Specifically, the case of remote client control for VNC was an issue, since a user could push their screen to a remote display, but could not position, scroll, or maximize/minimize the client window. There are a number of immediate ways around this problem, specifically using another VNC connection to control the remote display or remoting a mouse or other input device over USB, but these solutions only offer limited relief to the underlying problem. This becomes more relevant in a collaborative environment because users may need to control the client to mediate limited client resources: e.g., switching between shared displays or tiling clients appropriately.

Observational Contributions

Based on experiences with the Composition Framework, described above, Platform Composition contributes a number of powerful properties to collaborative environments, even though it is constructed using a number of standard services and protocols. First, since users can easily create new connections to facilitate sharing, it becomes feasible for them to create multiple connections while previously they might only have created one due to the perceived overhead of interaction. Second, the interaction of standard system services with operating environments and applications allows users to fluidly adjust their composition space to meet immediate group goals. Essentially, these contributions stem from the lower bar for composition operations which otherwise are too onerous for users to consider in the midst of dynamic group interaction – a primary design goal of the system.

As mentioned above, empowering users with a unified and easy-to-use sharing interface crystallized new usage patterns that were not previously present. For example, when showing a presentation to a small group the most straightforward way to realize a composition system is to enter presentation mode on the local device and then create a composition with the projector (assuming the existence of a dedicated projection server). However, it was discovered that utilizing storage sharing to push a document to a remote display and then pull the remote display locally allowed for increased functionality. This shift allowed a user to effectively present a document to a group of people without unnecessarily exposing their system to the audience, e.g., their local screen was still private, and only the intended document was shared, although under their control. Furthermore, this sharing approach enabled other users to directly access the presentation locally on personal devices and could flip ahead or review pages, a usage not possible with a simple projection or display-service export. Another advantage was that the presentation program (PowerPoint) executed locally and therefore did not incur any display artifacts for the audience resulting from display sharing, typically present for animations and embedded videos. While this example utilizes infrastructure support, the shift in usage highlights the power of Platform Composition.

Since they are built on common existing standards, platform services can be fluidly configured in the system without significantly interfering with individual applications or the user’s overall operating environment. Within the wealth of available platform services, sometimes it will be better to share individual files using the clipboard, other times they might want to access a complete file share, and sometimes they would want to interact using display sharing. Since Platform Composition utilizes resource sharing that is intimately integrated with standard system interfaces, users can quickly switch between modalities using familiar techniques like iconifying a window, pressing a keyboard shortcut, or drag-and-drop between windows. Once the initial connections are set-up, the composition mechanism itself does not play an active part in interactions. This fluidity of accessing sharing models transcends individual applications and allows users to share their resources in a manner that matches the sub-task of the overall group.

Satisfying another design goal, the Composition Framework was highly successful in supporting different applications. Photograph sharing was easily accomplished using built-in photo viewers and by sharing photographs through display sharing. Existing audio and video playback applications, such as Real Player or Windows Media Player, could easily operate on data exported through shared drives. Furthermore, Power Point easily ran in the environment and was used for shared presentations, while accompanying storage sharing was able to share the underlying presentation itself. However, the underlying services were not able to support video over a display-sharing channel, and interactions with more advanced applications such as Photoshop suffered from network latencies, both for the underlying storage and display sharing.

FUTURE WORK

Based on the concepts outlined in the previous sections, a number of avenues for future work become apparent. First, a more comprehensive evaluation of the various platform sharing techniques and how they interact would provide valuable additional insight on how sharing is realized using Platform Composition.

In addition to the individual underlying platform services to enable resource sharing, there is additional state present in the system that could be used to facilitate interactions among users. For example, data from the process table or files held open by programs, such as an image editor, could be used to provide a remote user with a more precise window into the first system's exported storage share (by opening up a window directly to a sub-folder in the larger hierarchy, instead of simply to the root of the storage share). Similarly, if one user is browsing a specific web page and would like to share it with their colleague, state sharing would allow another user to easily access the same page (currently, this can be accomplished by sharing the clipboard and using it to transfer the URL, but this introduces another step).

Currently, the security model employed by the Composition Framework relegates the security policy to individual services. Integrating authentication and access control with the core system has the advantage of presenting a unified front to the user as well as, similar to above, enabling more seamless switching between services. For example, successful authentication with one service might imply implicit authentication to another, reducing the number of unnecessary steps towards the final composition. Privacy falls into a similar situation where it is left up to social conventions or the pre-existing system configuration to manage users' privacy, an arrangement that could possibly benefit from a more managed approach.

As mentioned previously, an interesting aspect of resolving ad-hoc references is utilizing physical interfaces to invoke compositions – towards that end, realizing a multi-modal interface, combining elements of an on-screen, physical, and speech interface, offers several compelling properties. One complaint with the GUI interface was *"When my hand is not free, I cannot use GUI. I have to rely on mouse a lot to use GUI, which is inconvenient in many environment,"* highlighting another problem with traditional on-screen interaction. Physical interfaces such as Near Field Communication (NFC) [24,23] can partially address the security problem between devices enabling physical access-based security policies. Furthermore, a speech interface would be applicable for small devices since it would have a reduced dependence on the screen. Overall, both physical and speech interfaces are attractive in a social environment because they provide transparency: one user can easily see or hear what another is doing.

Along these lines, recommendation systems can be used to help the user better decide which services to compose together. Such a module could look at the available devices and service, currently running applications, people involved or nearby, and other sources of context to suggest a specific sharing configuration. For example, if two colleagues often and typically share their storage and display when they are together, it would be natural for the system to recommend or automate the process.

CONCLUSIONS

The Composition Framework prototype has demonstrated the effectiveness of the Platform Composition concept, exploring how it fits into the framing of the Pervasive Collaboration application domain. Unlike many other collaborative systems, the necessary enabling concepts manifest themselves in a very lightweight manner, removing many potential barriers-to-adoption for platform-level composition. The basic principle of service composition, combined with an understanding of the spectrum of application sharing modes, makes Platform Composition well suited to support collaborative work on emerging mobile devices, and further integrating them with existing infrastructure to build effective systems that support work practice.

They key contributions outlined in this paper are a deeper understanding of collaborative systems by first providing a design space within which to place such systems, and then identifying several non-obvious benefits of the Platform Composition approach. These contributions highlight how emerging pervasive computing environments based on common mobile devices are capable of supporting highly dynamic group usage models. Furthermore, based on these experiences, a number of observed challenges highlight ways that existing pervasive systems can be evolved to better support collaborative usage models.

REFERENCES

1. Ballagas, R., Szybalski, A., and Fox, A. 2004. Patch Panel: Enabling Control-Flow Interoperability in Ubicomp Environments. In Proceedings of the 2nd int. Conf. on Pervasive Computing and Communications
2. Bardram, J. 1998. Designing for the dynamics of cooperative work activities. In Proceedings of the 1998 ACM Conf. on Computer Supported Cooperative Work
3. Biehl, J. T., Baker, W. T., Bailey, B. P., Tan, D. S., Inkpen, K. M., and Czerwinski, M. 2008. Impromptu: a new interaction framework for supporting collaboration in multiple display environments and its field evaluation for co-located software development. In Proc. of the 26th Conference on Human Factors in Computing Systems
4. Clawson, J., Voida, A., Patel, N. and Lyons, K. Mobiphos: A collocated-synchronous mobile photo sharing application. In Proc of MobileHCI 2008.

5. Ducheneaut, N., Smith, T. F., Begole, J. "., Newman, M. W., and Beckmann, C. 2006. The orbital browser: composing ubicomp services using only rotation and selection. In CHI '06 Extended Abstracts

6. Edwards, W. K., Newman, M. W., Sedivy, J. Z., Smith, T. F., Balfanz, D., Smetters, D. K., Wong, H. C., and Izadi, S. 2002. Using speakeasy for ad hoc peer-to-peer collaboration. In Proc. of the 2002 CSCW

7. Egi, H., Ohsuga, N., Nakada, A., Shigeno, H., and Okada, K. 2004. DACS: Distance Aware Collaboration System for Face-to-Face Meetings. In *Proc. of the 2004 Symposium on Applications and the internet-Workshops*

8. Forlines, C., Shen, C., Wigdor, D., and Balakrishnan, R. 2006. Exploring the effects of group size and display configuration on visual search. In Proceedings of the 2006 20th Conference on CSCW

9. Hazas, M., Kray, C., Gellersen, H., Agbota, H., Kortuem, G., and Krohn, A. 2005. A relative positioning system for co-located mobile devices. In *Proceedings of the 3rd international Conference on Mobile Systems, Applications, and Services*

10. Hinckley, K. 2003. Distributed and local sensing techniques for face-to-face collaboration. In proc. of the 5th international Conference on Multimodal interfaces

11. http://www.maxivista.com/

12. http://www.incentivespro.com/usb-redirector.html

13. Johanson, B. and Fox, A. 2004. Extending tuplespaces for coordination in interactive workspaces. Journal. System. Software.

14. Johanson, B. Ponnekanti, S., Sengupta, C., Fox, A.: Multibrowsing: Moving Web Content across Multiple Displays. In Proc. Ubicomp '01. Springer, 2001.

15. Johanson, B., Hutchins, G., Winograd, T., and Stone, M. (2000) PointRight: Experience with Flexible Input Redirection in Interactive Workspaces. Proc. of UIST

16. Juwon Ahn, Jeffrey S. , Pierce, Jeffrey S. SEREFE: Serendipitous File Exchange Between Users and Devices; Proceedings of the 7th international conference on Human

17. Lauwers, J. C. and Lantz, K. A. 1990. Collaboration awareness in support of collaboration transparency: requirements for the next generation of shared window systems. In Proc. of the SIGCHI Conference on Human Factors in Computing Systems: Empowering People

18. Li, D. and Li, R. 2002. Transparent sharing and interoperation of heterogeneous single-user applications. In Proceedings of the 2002 ACM CSCW

19. Lyons, Kent; Want, Roy; Munday, David; He, Jiasheng; Sud, Shivani; Rosario, Barbara; Pering, Trevor: "Context–Aware Composition", HotMobile 2009.

20. Miller, R. C. and Myers, B. A. 1999. Synchronizing clipboards of multiple computers. In *Proceedings of the 12th Annual ACM Symposium on User interface Software and Technology*

21. Myers, B. A., Stiel, H., and Gargiulo, R. 1998. Collaboration using multiple PDAs connected to a PC. In *Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work*

22. Newman, M. W., Ducheneaut, N., Edwards, W. K., Sedivy, J. Z., and Smith, T. F. 2007. Supporting the unremarkable: experiences with the obje Display Mirror. Personal Ubiquitous Computing

23. Newman, M., Elliott, A., and Smith, T.. Providing an Integrated User Experience of Networked Media, Devices, and Services Through End-User Composition. In proc. of the int. conf. on Pervasive Computing, 2008

24. Pering, T., Anokwa, Y., and Want, R. 2007. Gesture connect: facilitating tangible interaction with a flick of the wrist. In Proceedings of the 1st international Conference on Tangible and Embedded interaction

25. Pering, T., Ballagas, R., and Want, R. 2005. Spontaneous marriages of mobile devices and interactive spaces. Commun. ACM

26. Richardson, T., Stafford-Fraser, Q., Wood, K. R., and Hopper, A. 1998. Virtual Network Computing. *IEEE Internet Computing*

27. Roseman, M. and Greenberg, S. 1996. TeamRooms: network places for collaboration. In Proceedings of the 1996 ACM Conference on CSCW

28. Satyanarayanan, M. 2002. The evolution of Coda. *ACM Trans. Comput. Syst.* 20, 2 (May. 2002)

29. Schoeneman, C. 2003. Control everything from one place with Synergy. Linux J. 2003, 108 (Apr. 2003).

30. Shen, G., Li, Y., and Zhang, Y. 2007. MobiUS: enable together-viewing video experience across two mobile devices. In proc. of the 5th international Conference on Mobile Systems, Applications and Services

31. Stødle, D., Bjørndalen, J. M., and Anshus, O. J. 2007. The 22 megapixel laptop. In Proceedings of the 2007 Workshop on Emerging Displays Technologies: Images and Beyond: the Future of Displays and interacton

32. Streitz, N. , Geißler J., Holmer, T., Müller-Tomfelde, C., Reischl W., Rexroth P., Seitz P., and Steinmetz R. i-LAND: an interactive landscape for creativity and innovation. In Proceedings of the SIGCHI conference on Human factors in computing systems, pages 120–127. ACM New York, NY, USA, 1999.

33. Szentgyorgyi, C., Terry, M., and Lank, E. 2008. Renegade gaming: practices surrounding social use of the Nintendo DS handheld gaming system. In Proceeding of CHI 2008

34. Teege, G. 1999. Users as Composers: Parts and Features as a Basis for Tailorability in CSCW Systems. Computer Supported Cooperative Work (Oct. 1999).

35. Voida, S., Edwards, W., Newman, M. W., Grinter, R. E., and Ducheneaut, N. 2006. Share and share alike: exploring the user interface affordances of file sharing. In Proceedings of the 2006 CHI

36. Want, R., Pering, T., Danneels, G., Kumar, M., Sundar, M., and Light, J. 2002. The Personal Server: Changing the Way We Think about Ubiquitous Computing. In proc. of the 4th int. conf. on Ubiquitous Computing

37. Want, R., Pering, T.. Sud, S., Rosario, B., 2008. Dynamic Composable Computing, In proc. of HotMobile 2008.