Considering Mobile Devices, Context Awareness, and Mobile Users

by

Jing Su

A thesis submitted in conformity with the requirements
for the degree of Ph.D.
Graduate Department of Computer Science
University of Toronto

Abstract

Considering Mobile Devices, Context Awareness, and Mobile Users

Jing Su

Ph.D.

Graduate Department of Computer Science

University of Toronto

2010

Recent years have seen rapid growth and adoption of powerful mobile devices such as smartphones, equipped with sophisticated input and display systems, and multiple communication technologies. This trend has coincided with the rapid deployment and adoption of high-speed Internet services and web-based applications. While this rapid development of mobile technology has provided great opportunities, it also presents significant new challenges compared to traditional desktop computing. Specifically, unlike the traditional desktop computing experience where users are stationary and physically isolated, users in mobile and social settings can be faced with real time demands for their attention.

This thesis examines the relationship between mobile devices, context awareness, and mobile users. We propose the use of physical proximity context to adapt and improve system behaviour, and enable mobile users to more effectively access and share content in non-desktop settings. This work identifies three distinct challenges in mobile software, and addresses these challenges using physical proximity context awareness. First we address improvements to mobile node network utilization by using proximity awareness to automatically manage local radio resources. Next we address improvements to mobile web-backed applications and services by enabling social proximity awareness. Finally, we enable greater mobility and physical awareness for visually impaired users on mobile devices by providing an interface which enables exploration of spatial geometric layouts.

Dedication

To Mom, Dad, Nancy, Jenny, and Juliana for your love and support throughout graduate school, and amazing patience in waiting for me to pinch this one out.

Debugging is twice as hard as writing the code in the first place.

Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it.

– Brian W. Kernighan

The ability to reduce everything to simple fundamental laws does not imply the ability to start from those laws and reconstruct the universe.

– Philip W. Anderson

Contents

List of Figures

Chapter 1

Introduction

Recent years have seen rapid growth and adoption of powerful mobile devices such as smartphones, equipped with sophisticated input and display systems, and multiple communication technologies. This trend has coincided with the rapid deployment and adoption of high-speed Internet services and web-based applications. It can be argued that these two trends have been driven by a synergistic and symbiotic push from providers seeking distinguishing features and market advantage, and consumers seeking ever-greater means of accessing, creating, and sharing content.

While this rapid development of mobile technology has provided great opportunities, it also presents significant new challenges compared to traditional desktop computing. Specifically, unlike the traditional desktop computing experience where users are stationary and physically isolated, users in mobile and social settings can be faced with real time demands for their attention. An important factor for addressing this challenge is the consideration of context and context awareness. Current devices and mobile applications behave no different regardless of the users' social context. Devices rely on manual user intervention to toggle between different radio communication systems, and applications require users to manually select their peers, coordinate content of interest, and figure out how to set sharing permissions. This makes it difficult to spontaneously access and collaborate on content as part of a social exchange, since

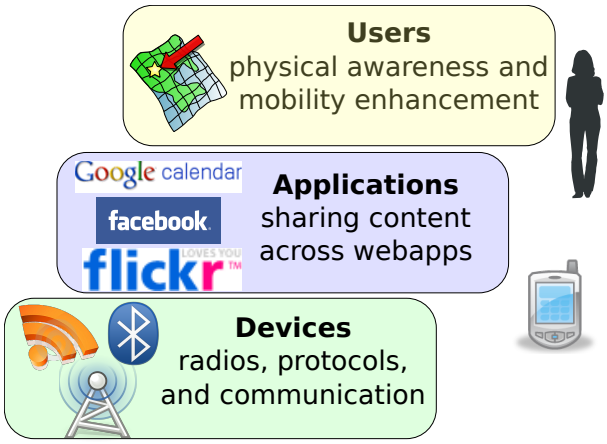


Figure 1.1: *Three layers of concern in mobile systems: (1) radio resources, protocols, and connection models in mobile settings; (2) web-based applications and services, and sharing and collaborating on content in social settings; and (3) user awareness of their physical and mobile environment.*

these chores require users to divert their attention to the mobile device.

This thesis hypothesizes that physical proximity context can help improve access to data as well as improve awareness of physical environments. To validate this hypothesis, we identify three basic layers of concerns in mobile systems, as pictured in Figure 1.1, and target specific challenges in each of these layers.

First, at the device level, mobile devices have a wide array of communication capabilities, each with varying connection models, capabilities, and user front-ends. These currently must be manually managed by the user, making devices distracting and cumbersome to use in a social setting for sharing data. Second, at the application layer, web-based applications and services provide useful facilities for sharing and collaborating on content when users are communicating remotely. However, in a social setting, these applications impose distracting chores such as searching for relevant content, identifying peers, and setting correct permissions. Third, at the user layer, a significant feature of mobile devices is their ability to enhance user mobility and awareness of their environment. Unfortunately, there currently exists a significant gap between what devices are capable of and what is accessible to visually impaired

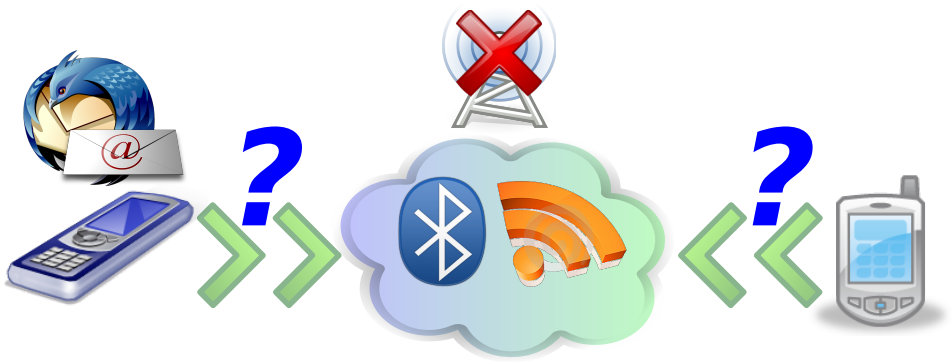


Figure 1.2: *Devices are equipped with a multitude of radio systems for different mobile situations. Unfortunately, these features require significant manual user management. Even after successful pairing, user attempts to send data can fail for non-obvious reasons.*

users.

In the remainder of this chapter, we highlight each of these layers, the challenges within each layer, and how physical proximity context can be used to address these challenges. Our contributions provide an enhanced mobile landscape with smooth data access and collaboration in social settings, and improved user awareness and mobility in physical environments.

1.1 Device Networking

At the device networking level, users currently face a complex array of wireless networking options available to them. Each of these radio communication technologies, ranging from short-range Bluetooth, medium range WiFi, to long-range cellular, all have inherent trade-offs in strengths, weaknesses, and operating modes. In an effort to cover a greater gamut of mobile situations, many devices are equipped with more than one complementary radio technologies.

Unfortunately, because these different radio technologies have different operating modes and expectations, users are left with disjoint applications designed for specific networking technologies, useful in different mobile settings. For example, infrastructure-based applications, such as email shown in Figure 1.2, assume the existence of infrastructure services such

as DNS for resolving name information and service locations. In a peer-to-peer connection where such services may be unavailable or incomplete, email applications will fail to deliver messages even if the recipient's device is reachable in the peer network. This kind of non-obvious system failure is especially frustrating for users collaborating in physical proximity. Such limitations are further complicated by differences in assumed operating modes across the underlying radio technologies. For example, WiFi P2P links assume availability of TCP connections, whereas such an assumption does not immediately hold over Bluetooth links.

These limitations force users to manually manage the applications and radio technologies to use across different mobile contexts, and coordinate and synchronize their data across these disjoint applications. This deficiency in systems support for integrating the available hardware resources leads to an inconvenient and frustrating user experience where the reasons for incompatibilities and failures may not be apparent.

The approach to addressing this challenge is to provide networking support which is aware of the current mobile context, and enable applications to function under dynamic networking conditions. The insight to our approach is to provide a data-centric late-binding framework for forwarding algorithms, end-point naming, and interface and transport mechanisms. This approach enables the networking layer to asynchronously determine best networking options independent of applications.

Haggle (Chapter 3) is a framework that provides common facilities for identifying and reaching specific peers, especially peers in physical proximity, independent of the underlying radio model. Users are free to utilize mobile communication applications agnostic of the underlying device radio or networking model. For example, if two devices are connected via Bluetooth, email messages between the two devices will still be deliverable under Haggle despite lack of infrastructure DNS.

In developing the Haggle architecture we made the following contributions:

1. We present the implementation of the Haggle architecture which utilizes physical proximity context to enable more effective use of device radio resources, and provide contin-

ued communication capabilities to mobile applications. Our implementation provides a full synthesis of Haggle's concepts as well as defining its internal and external interfaces.

2. We describe and approach for supporting legacy applications within Haggle, by providing proxy service modules that bridge infrastructure-dependent operations.

3. Finally, we present an evaluation of the Haggle architecture and new capabilities it enables in legacy applications.

1.2 Mobile Applications and Services

At the application level, users are increasingly relying on web-based services and applications for accessing, storing, and sharing content. This transition has been fuelled by the availability of high-speed Internet access which enabled users to rely on web services which are highly available, dependable, and accessible from multiple locations and devices.

While these applications work well when users are accessing content in isolation, they impose a highly distracting barrier when users access web services in a social setting. During face-to-face interactions, any delays or distractions are detrimental to the smoothness of the social exchange. Current mobile applications still rely on a desktop-like usage model where users are assumed to be working in isolation. This causes jarring breaks in the social interaction as users focus on their mobile application to access or share content. For example, social networking applications such as Facebook and web applications such as GoogleCalendar are popular for sharing and collaborating. Yet these tools are oblivious to the peers around the user in a social setting, and as a result offer application behavior that is no different than if the user is sitting in isolation at a desktop. Users must currently withdraw from the social exchange and focus on their mobile device to find the online identity of their peer, locate relevant content, and request or set sharing permissions. These chores may be acceptable when working from a well provisioned desktop or laptop in an office, but it is highly disruptive in a social setting.

The key to tackling this problem is the observation that users already have a natural and familiar paradigm in the real world for sharing and referring to content of interest – by the visual identification of the peer in proximity. Our approach is to provide an intuitive locus for referring to and sharing digital content based on real world identities. To realize this approach, it is necessary for mobile devices to determine the people that are in physical proximity. We observe that most mobile devices today are equipped with at least one long and one short range wireless networking technology. By utilizing the short range wireless radio as a sensor, it is possible to effectively determine the devices in physical proximity, and therefore, by association, the people that are in physical proximity.

We contribute a novel framework which enables people to be treated as first class entities, bridging physical social presence with online identities and resources. We achieve this using a two level map from devices to owners, and from owners to online resources. This enables sharing and permissions to be established by referencing individuals rather than machine-based addresses or service-specific identifiers. Furthermore, this mapping also enables web-based content, spanning different services and users, to be referenced, sorted, and filtered based on individuals. Copernicus (Chapter 4), utilizes social proximity context to custom tailor web services and applications at the source, using programmable interfaces exported by these Web2.0 services. Possible adaptations range from identifying the web-identities of those in social proximity, sorting and filtering content based on these identities, and automatically setting permissions and access control lists for shared content. Now, for example, when two or more users try to share data as part of their conversation, their peer and peer content will automatically be filtered to the top of their content lists and their calendars will automatically be overlaid, even if they're on different social networking or calendaring applications. The proximity context aware system enables mobile devices and applications to become supplemental aids that anticipate and enhance social interactions seamlessly.

In developing Copernicus, we made the following contributions:

1. We developed a sharing model that enables individuals in proximity of the user to be

treated as addressable objects that can be used as a locus for content exchange.

2. We show that by customizing web content/services based on proximity context we greatly simplify the task of exchanging information in face-to-face interaction.

3. We present an architecture that leverages the multiple radios available on modern mobile devices for sharing. Specifically, we use short range radio to sense proximity and long range radio for communication.

1.3 User Context

A significant and popular function of smart mobile devices is the ability to provide users with awareness of their physical environment. This appears in many forms on current smart devices, ranging from map navigation to suggestion and rating systems for nearby venues and events. In essence, greater user awareness of the physical context around them improves their mobile experience.

Unfortunately, there currently exists a significant gap in systems support for providing environmental awareness for visually impaired users. The most commonly used electronic navigational system is voice-assisted global positioning satellite (GPS) systems for helping visually impaired users navigate city streets via spoken turn-by-turn directions. These systems provide limited capability for querying nearby destinations, often presented as long spoken lists. This list format is poor at providing relative positioning context of destinations and their relation to other points of interest and landmarks. GPS systems are also limited in their ability to provide guidance around walking hazards such as detours around sidewalk construction. These hazards are extremely difficult for visually impaired users to navigate since they have very little context with regards to the length, direction, or even boundaries of the detour if it is demarcated by cones and tape. Furthermore, the scope of GPS is limited to outdoor streets. There is no pervasively available indoor localization and layout presentation and guidance system.

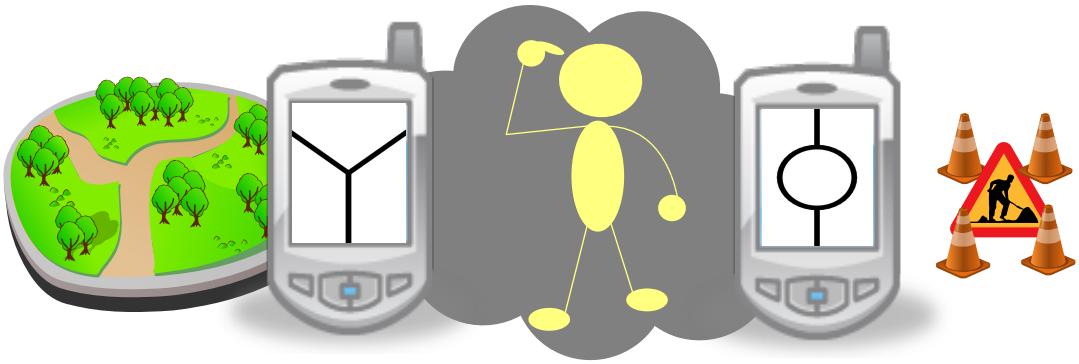


Figure 1.3: *User awareness of their physical context is critical for enabling mobility. This is particularly challenging for visually impaired users who currently have limited electronic aids for exploring paths and navigating hazards.*

We are motivated to investigate this problem from the perspective of mobile devices for three reasons. First, because smart devices have become so prevalent, they are generally affordable to most individuals, and the use of these devices are within common social norms. Second, a mobile platform can provide visually impaired users with greater freedom to alter and explore different locations without having to return to a home computer in order to access information. Finally, smart mobile devices can acquire physical proximity context and present this information to visually impaired users so they may build a greater cognitive awareness of their physical environment, as illustrated in Figure 1.3. Sources of this context can range from a combination of GPS location, cellular and WiFi fingerprints, compass direction, nearby Bluetooth devices and tags, and potentially even images from built-in cameras.

To tackle this large problem space, we focus on a more manageable first step; providing awareness of physical indoor layouts on mobile touch-screen devices for visually impaired users.

The intuition for addressing this problem is that the physical motion of finger tracing on a touch screen can provide precise geometry and spacial positioning information to the user. To investigate this approach, we identify three specific goals. First, the sonification (non-verbal auditory cues) must be effective in guiding the user's finger. Second, the tracing motion

must be sufficiently to accurately and precisely convey non-trivial geometry. Finally, user comprehension of geometry must be transferrable to building a cognitive understanding of a map layout.

We present Timbremap (Chapter 5), an interface for the visually impaired that uses sonification to convey two-dimensional geometry to the visually impaired. Timbremap uses audio techniques to effectively convey fine-grained path information via guided finger tracing on a touch-screen mobile device.

Our investigation of Timbremap makes the following contributions:

1. We show that visually impaired users are able to recognize 2D geometry by tracing their finger over a touch screen interface that provides sonification feedback.

2. We show that Timbremap's technique scales to complex geometries and enables visually impaired individuals to explore complex indoor floor-plans.

3. We present two different sonification interface implementations and show that they both achieve similar good performance. We conclude from this that the technique in itself is robust (it does not depend on the vagaries of the specific implementation), and that the choice of specific implementation is more an issue of user preference.

1.3.1 Document structure

The remainder of this thesis is organized as follows. Chapter 2 provides a general background of this thesis, from wireless networking to proximity context and feedback. Chapter 3 then presents the Haggle system which enables dynamic switching of multiple wireless networking systems based on mobile and environmental context. Chapter 4 presents the Copernicus system which enables dynamic modification of web applications and services based on users' social proximity context. Chapter 5 presents Timbremap, a system which enables visually impaired users to explore map layouts on touch surface mobile devices. Finally, this thesis provides concluding remarks in Chapter 6.

Chapter 2

Background

This chapter provides an overview of the broad spectrum of background related to the topics covered in this thesis. This chapter begins at the device level, covering the basics of wireless networking on mobile devices. We then cover works which investigate network interface selection and peer identification. Following that, this chapter discusses efforts in content management, and the use of context in improving mobile software systems. Finally, this chapter transitions to user interface concerns in audio-based feedback and notification, followed by works investigating navigation and exploration.

2.1 Mobile Devices and Networks

The area of mobile and ubiquitous computing covers many forms of mobile devices, ranging from user devices such as laptops and smartphones, to devices in the environment such as mobile nodes and sensors. In this work we focus on interactive and network capable personal mobile devices carried by users. In general, we assume that devices have at least one form of wireless radio communication, and that this interface can be activated or connected independent of the user.

Traditionally, this category has been generally filled by two classes of mobile devices: devices which have severely constrained interface and computation capabilities but are always

network-ready, such as smartphones; and devices such as laptops which approach desktop-level computing when stationary, but are suspended and disconnected when mobile. However, as smartphones become more powerful and laptops and ultra-mobiles feature hybrid modes [73], the distinction between these two classes of devices will blur.

To facilitate communication, mobile devices now typically have two or more wireless radios. Since each radio design has its own characteristic strengths and weaknesses, having multiple radios helps to cover the majority of users' mobile communication needs. As of this writing, the most common deployed technologies are Bluetooth, 802.11 WiFi, and cellular.

Bluetooth is a short range radio operating on the 2.4 GHz unregulated spectrum space. Typical operating ranges are approximately 15 feet (just under 5 meters) radius, depending on the environment. Conceptually, Bluetooth radios are designed for ad hoc networking, mostly used for personal area networks or impromptu data transfers between devices. The Bluetooth radio and stack have been designed to be relatively power-efficient, typically consuming an order of magnitude less power than WiFi.

Also operating in the free 2.4 GHz spectrum is 802.11b/g WiFi. Though WiFi also supports an ad hoc operating mode, it is most often used in infrastructure mode to connect to an access point. This operating mode provides great benefits for users in terms of high-speed connections at typically low monetary cost. However, the restricted range of WiFi signals typically bounds users to within indoor environments where access points have been placed. Even within WiFi covered areas, noise and propagation characteristics can have significant impacts in link quality [13].

Finally, cellular networks typically have the longest range and most wide-spread coverage. Cellular radios only operate in infrastructure modes and on regulated spectrum. A trade-off for this increase in availability and coverage is increased monetary and energy cost for the end-user and reduced transfer speeds. Because cellular networks operate on licensed spectrum's and require heavy investments to establish and maintain cellular towers, users pay for access to the network. The cost of these tolls can be substantial if the user roams to other networks

not operated by their home provider.

In the remainder of this section we cover the different types of network topologies typical of mobile devices and the works which have attempted to address some of the challenges in these areas. We cover three major classes of networks for mobile devices, ranging from most stable and connected to least stable with many disconnects: infrastructure covered networks, mobile ad hoc networks, and delay-tolerant networks.

2.1.1 Infrastructure Wireless Networks

Infrastructure wireless networks are two-tier systems where there are infrastructure access points which provide and coordinate connectivity to many mobile clients. Mobile clients always communicate through the access points – never directly with other clients. Unlike a traditional wired network where end-point nodes do not move, wireless infrastructure networks must manage nodes as they traverse out of the coverage area of one access point and transition into another. The main challenge in supporting a mobile client is managing its hand-off to the next access point, and maintaining consistency in the client's network address. As a mobile client moves towards the edge of an access point's coverage area, its connection to the network must be handed-over to the next adjacent access point that can continue to service the client. However, in switching to the new access point, the mobile device may be issued a new IP address. Because IP addresses contain implicit hierarchical routing information as well as identify the end-point client, the challenge is to ensure the user's existing network connections stay live and properly route to the mobile device.

One solution for this problem is MobileIP [63, 65]. In MobileIP, mobile devices are assigned a permanent unchanging *home* address which is always reachable and maintained by the device's home network. When roaming to a different networking domain, the mobile device may receive a new and different IP address. The mobile device notifies their home network of their new address, which is referred to as the *care-of* address. While roaming, responses from remote services are delivered to the well-known and fixed home address of the mobile device.

The home network then tunnels the packets to the mobile device at the care-of address.

In today's Internet, most services are accessed via asynchronous request/response messages over reliable TCP links. This transition has mostly obviated the need for MobileIP solutions since user and device identification are performed at higher application-level protocols. Within the same cellular provider, cellular networks utilize their own handover and roaming mechanisms, which allow them to provide streaming television and video services without MobileIP.

2.1.2 Mobile Ad Hoc Networks

Unlike the other infrastructure systems, mobile ad hoc networks (MANETs) have no a priori hierarchy (though some protocols allow ad hoc formation of hierarchy) distinguishing between clients and access points. Since all nodes in a MANET are considered mobile, nodes must be able to connect to new neighbours and peers and help forward packets for other nodes in order to maintain end-to-end routing paths between communicating nodes.

The two most popular basic routing algorithms are DSR [10]. and AODV [64]. Both algorithms establish routes on-demand, helping to reduce radio use when there is no traffic to send. As a source-routing protocol, DSR first establishes the full path from sender to receiver and tags this path to the packet before it is transmitted by the sender. In AODV, each node maintains a next-hop routing table for each destination. Packets are forwarded according to this next-hop table until they reach the addressed destination.

Because MANETs require cooperative intermediate nodes, and cooperative nodes must expend valuable energy and transmission time to forward other nodes' traffic, considerable research has investigated how to establish fairness and cooperation [21, 28, 31, 61].

2.1.3 Delay/Disruption-Tolerant Networks

Unlike a MANET where disconnects and partitions are infrequent, nodes in delay-tolerant networks [38] (DTNs) operate mostly in a disconnected state where there is rarely a contem-

poraneous multi-hop end-to-end path from source to destination. To overcome this limitation, DTNs rely on some pattern or predictability in the mobility and interaction of nodes to make progress in forwarding packets toward their destinations. For example, existing works have proposed using regular services such as postal mail delivery to provide a delay-tolerant network [87]. Other works have proposed the use of DTNs for challenged environments [26], such as in rural settings [70].

Many research efforts have explored techniques to provide reasonable routing performance in DTNs without resorting to epidemic flooding, which can overwhelm and collapse the network as well as waste significant amounts of energy and storage. One technique is to use limited replication strategies, such as in Spray and Wait [74], where source nodes "spray" replicas of the message to a subset of their known neighbouring peers. The source node then waits for these neighbours to either contact the destination directly or forward the message to subsequent intermediary nodes who may reach the destination. Other research efforts have studied the use of non-random human and vehicular mobility and contact patterns to form DTN routing paths [16, 34]. In our previous research efforts, we have suggested the use of gossipping protocols to disseminate the contact patterns and potential forwarding paths through the DTN [76].

2.2 Network Interface Selection and Naming

For mobile devices with potentially more than one wireless communication radio, a significant challenge arises in deciding which communication mechanism to use and how best to utilize it. Due to the mobile nature of these devices, the best choice at any one time may change depending on the location of the device, its mobility, and available resources.

At the basic level, multiple efforts have attempted to encapsulate and unify the presentation of network interfaces [55, 60, 79]. These abstraction layers attempt to mask differences in vendor and network features from the networking stacks and applications which utilize the

interfaces. Most approaches abstract interfaces into packet ingress and egress points. However, this abstraction does not address when to use these interfaces, if there is more than one, and how best to utilize their capabilities.

One approach for managing multiple network interfaces is to use policy-based mechanisms. The basic approach is to utilize interfaces in a way which maximizes their strengths and minimizes the impacts of their weaknesses. Existing efforts [86] have proposed systems which perform wireless communication system selection and stream hand-off based on user-defined policies. In addition to policies which select the single best wireless communication interface to use, other efforts, such as Horde [62], describe systems for also striping traffic across multiple varying wireless networking links. The Horde middleware system provides a facility for specifying network striping policies separately from striping mechanisms. This enables applications to indicate how content streams can be separated and striped, while allowing the middleware to use policy based rules for selecting network interfaces and striping data to maintain quality-of-service preferences.

The challenges for wireless networks go beyond interface selection. In mobile environments in which topologies and memberships are dynamic, mobile devices must be able to find the appropriate service or content, and name particular end-clients. One approach is to utilize late-binding name systems to decouple naming from physical addressing.

Late-binding name systems allow applications and services to rendezvous based on descriptive names over a self-organizing overlay network. Decoupling naming from the physical addressing provides a clean abstraction for supporting dynamic and mobile nodes in the network topology as well as routing based on new metrics such as location and domain specific contexts. The i3 [75] system hashes the name identifier space into a DHT overlay network, allowing applications and services to rendezvous at the same overlay node, independent of node mobility. INS [1] allows applications to specify names as trees of key-value pairs expressing the desired service or device. Each node participating in the routing overlay network can perform matching functions to determine where best to forward the request in order to find a

matching destination.

2.3 Content Management

Within the context of this thesis, managing content on mobile devices involves three general challenges: caching and hoarding; identification, sharing, and access; and adaptation and transformation.

Existing works have examined the hoarding and caching of content for the purposes of maintaining mobile access to content. Courier [41] provides a hoarding / caching system on the mobile device which pre-allocates documents and works which are relevant to the user. These hoarded documents can then be shared with others. Courier provides a user interface and content management model where only the owner of a document can share it with others, and only the owner may revoke sharing rights with others (for example when the meeting ends). Social presence is detected via Bluetooth and provides users with the ability to set up a meeting. Content sharing is done via a mediator, the meeting PC. The other recipients obtain their copy of the shared file via the meeting PC.

In addition to hoarding user-owned content, existing efforts have also examined mechanisms for sharing content between mobile devices. Cobalt [83] provides a mechanism for peer-to-peer based media sharing, with a focus on digital rights management. The primary focus of the Cobalt [83] system is to enable the sharing and building of dynamic collections of media based on physical proximity while protecting the content rights of the shared media. For example, a dynamic play-list of music files can be created based on the proximity of nearby individuals, with the play-list contents automatically updated and authorized based on the nearby participants.

As the abundance of available content and methods for generating content grows, many works have developed methods for providing context tags to better facilitate the identification and sharing of content. A common insight in this area of work is the observation that mobility

in geographic location and social groups provides context which can be used to tag and sort data. INCA [78] and ContextCam [58] utilize this observation to automatically tag captured data from mobile devices. Mobile Media Metadata [68] provided a framework in which photographs taken using the mobile phone are tagged with signatures detected by the device, and sent to a server which performs image recognition to further tag and group the photograph. The processed result is given back to the device for user acceptance and or correction.

Due to the constraints on display, input methods, power, and wireless communication cost, many works have examined mechanisms for adapting content to minimize the effects of these constraints. For example, there has been significant efforts made in techniques for optimizing the utilization of wireless links by adapting the desired contents, particularly for infrastructure systems which provide wide-availability and stability but at an increase in monetary cost and energy consumption. The Puppeteer [46] project takes advantage of the exported component control interfaces to provide efficient adaptation of data transport. The Urica [54] project presents a system which utilizes a proxy to serve images from web pages at low fidelity to save bandwidth. Users are then able to improve the quality of images, slice at a time, depending on their needs. However, instead of treating each individual separately, Urica utilizes a clustering algorithm to identify like-preference individuals in context with web pages, to provide good initial predictions of desired quality levels. The PageTailor [9] project provides mobile web browsers with the ability to graphically tailor the DOM tree created by web pages. Not only can this improve the layout of content for certain classes of mobile devices, but it can reduce the amount of content downloaded by allowing users to remove unwanted sections. The insight in this technique is that popular web pages maintain the same layout styles for long periods of time. Thus a one-time customization done by the user can be applied for many months.

Many research efforts have attempted to use context in order to automatically tailor the display of web content on mobile devices. This has been particularly important as mobile devices gain in computing power, but remain physically constrained in their form factor and input modalities by nature of their portable design. Chameleon [54] utilizes viewer adaptation

history in order to customize the presentation of images. CMo [12] utilizes machine learning on a proxy server to further digest and break down the content from a web page in order to serve it more succinctly and clearly. Many efforts have also attempted to improve the user interface on the mobile device, to better suit navigation, search, and selection of content returned from online services [6, 42].

2.4 Proximity Context

In the mobile environment, proximity to other devices or other markers can provide a rich source of context in addition to geographic location context. In many cases proximity context may be easier to detect and serve as a better determinant of social setting compared to geographic location.

2.4.1 Social and Device Presence

Many research efforts have explored the utility and use of social presence [20] for initiating the sharing of content, selecting the subset of applicable content, and determining the access rights and lifetimes of the share [41, 45, 67, 83]. The Cobalt [83] presented a system in which devices can automatically grant or revoke access rights based proximity. Push!Photo [67] presented a similar application of photo sharing based on social presence for content tagging and sharing. While this work shares a common motivating application with these other works, a significant distinction is that this work does not transfer content peer-to-peer between devices. The objective of this work is to enable proximity context filtering and searching for Internet based services in general. While the use of social presence [45] and collaborative filtering [32] has been previously suggested for improving services such as email, we believe this work contributes a novel study of an implementation for enabling context aware searching and filtering for Internet services.

Some works have studied the utilization of assumed close proximity of a mobile device to

its owner for establishing and inferring location context [72]. Patel et al [59] found that the assumed frequent closeness of users and their devices may not be as high when taking into account the whole day. This assumption and current results warrant continued investigation as users grow accustomed to mobile device use inside and outside of the home. For the purposes of this work, we are mostly interested in social settings, where we believe it is safe to assume users will have their mobile devices nearby.

2.4.2 Security and Privacy

There exist a wide range of works in the areas of security and privacy, with respect to mobile networks and environments. Two major concerns for mobile users and their devices is the ability to control their virtual presence to other devices in the environment, and ensuring that any communication with a locally nearby peer is secure.

SmokeScreen [20] uses clique identifiers which are identifiable by trusted users of desired cliques to identify the mobile device. In this way, unlike a system such as Bluetooth inquiries, the identity of the device is not given away to unwanted snoopers.

Amigo [82] presents a system where devices in close proximity take advantage of highly dynamic fluctuating local radio signals to establish proof of their privacy and allow them to create secure communication links and be able to detect man-in-the-middle attacks.

2.5 Feedback and Notification

In this section, we shift focus slightly to examine the basic background surrounding eyes-free feedback, specifically for visually impaired users. These types of feedback generally fall into two categories: sonification, which is non-speech audio, and tactile.

The basic motivation for examining various sonification techniques is analogous to visual signs and icons. It has been argued that icons transcend linguistics and culture because we mentally associate the visual icons directly with their intended metaphoric meaning, removing

a layer of abstraction that language would impose. Similarly, audio icons can portray the same level of rich and direct meaning as long as the sound taps into the mental association, even if the sound doesn't directly relate to the intended metaphoric meaning.

Auditory icons are sounds which directly represent objects, actions, functions. Earcons are abstract sound cues which indirectly represent an object, action, or function. For example an ascending tri-tone sound can convey "up". Spoken phrases, such as those produced by text-to-speech systems, are easy to understand but requires large amounts of mental resources. As a result, using TTS for notification can often be more intrusive than using other sonification techniques. Spearcons are a synthesis between speech and earcons, where speech is sped up to the point where it cannot be recognized as speech, but still identifiable. Thus it becomes a rich supplement to the earcon space.

Existing works [25, 85] have evaluated the effectiveness of earcons and spearcons, finding them highly effective in conveying the spoken meaning to the user while not imposing the cognitive load that standard speech incurs on the human listener. By using sonification which is highly compressed in duration, these techniques leave the speech processing channel of the user open for other more complex or abstract textual notifications. One caveat to remember is that speech is more difficult to localize than broad spectrum sounds. Therefore interfaces which require spatial localization must take this into account if they utilize spearcons.

The AUDIOGRAPH system [3] explored enhancements to the *earcon* concept, whereby musical sequences or relationships between musical sequences convey semantic information. Their results show that sophisticated audio sequences can be used to convey complex data, but users need to have understanding of musical concepts and be trained in the interface and musical concepts.

SemFeel [89] uses high-precision actuators to provide complex tactile feedback sensations. While this technique is effective at providing general notification feedback, it is currently not sufficient for conveying high-precision geometry. Furthermore, SemFeel requires complex add-on hardware that is not readily available as an off-the-shelf consumer product.

Finally, several efforts have investigated techniques for eyes-free menu and item selection on mobile devices [48, 90]. Slide rule [40], for example, developed a user-interface widget navigation and selection interface using multi-touch capabilities on mobile touch-input devices.

2.6 Navigation and Exploration

In this section we cover background for efforts in providing navigation and exploration information for visually impaired users. In general, exploration interfaces are most concerned in conveying geometry and spatial positioning relationships while the user is in a stationary setting. Navigation interfaces are most concerned in directing the user, using turn-by-turn directions, for traversing from a starting point to a destination point.

2.6.1 Exploration

Many projects have utilized tactile methods for exploring and conveying geometry. The BATS [57] project used force-feedback joysticks coupled to a pointer for providing tactile bumps and feedback over an interface as the cursor crossed boundaries and feature changes. Crossan et al [22] used a force-feedback 3-dimensional pen to guide the user's hand in a trajectory, outlining a geometry. These methods suffer from either requiring additional add-on hardware which is either expensive or non-portable. Furthermore, Crossan et al [22]'s results show that even with a highly precise tactile or force-feedback system, audio cues were still necessary. This suggests that audio systems are likely to be needed to complement interfaces for the visually impaired, even if highly precise deformable or tactile surfaces for mobiles are developed.

2.6.2 Navigation

Many research efforts have explored techniques for providing navigational aids for visually impaired users. While the current focus of Timbremap is not as a navigational assist, these related works are worth discussion as avenues for expansion and integration.

The SWAN [88] project explored the use of non-verbal auditory cues (i.e. sonification) to aid visually impaired users in way-finding. Unlike current GPS road-maps which typically guide users based on streets, the SWAN system can potentially guide users along way-points across non-road areas such as parks or large campuses. The approach used in the SWAN project is to use sonar-like beacons in a spacial audio system to guide users from way-point to way-point. Other efforts [35, 51] examined the use of some kind of pointer or remote which can be used to point at and query for environmental objects and information – effectively providing a "smart" enhanced cane. Marston et al [52] compared a haptic and audio based navigation systems, using in situ navigation in both a small city block as well as a park.

Subsequent follow-up work [84] examined the impact of users navigating via audio, while at the same time discriminating a secondary audio announced task (such as carrying a conversation). Their results show that contrary to initial assumptions, navigation performance degraded when presented with multiple audio demands, instead of the secondary audio task. In retrospect, this result can be seen in sighted users as well, with many people having difficulty "walking and chewing gum" at the same time, such as talking while operating a motor vehicle, resulting in degraded operational performance. This result is a more minor concern for the Timbremap interface, since we expect users to stand still and utilize it to explore or reference their location, rather than use it as a navigational aid.

Ross et al [66] investigated three different techniques for aiding visually impaired users in crossing intersections. This common task is difficult for the visually impaired since there are no physical aids or references which enable them to navigate across an intersection. As the person walks off the curb, the intersection is painted with lines which cannot be detected by the visually impaired individual. Furthermore, without any reference points, walking in a dead-reckoned straight line can be difficult. Ross et al investigate the use of earcons, speech, and tactile shoulder taps to aid individuals in this navigation task.

Many research efforts have examined audio and haptic techniques for conveying eyes-free notification and context information. Automated aids for user notification are an important line

of investigation, even if tactile notices, such as braille tags, are available. A difficulty with physically placed tags is that the user often cannot know a priori where the tags will be or how they are oriented [66], especially in unfamiliar environments.

Chapter 3

Haggle

Advances in computing technology have had a profound impact on the capabilities of portable devices such as smart-phones, notebooks, and personal digital assistants. Today these devices provide a rich computing environment and multiple communication methods based on different radio technologies such as short-range Bluetooth, medium-range 802.11 and longer-range cellular radios.

Users expect ubiquitous access to applications such as messaging and information browsing on these powerful devices. Unfortunately existing applications are often unable to take advantage of the mobility and connectivity options that may be present because they are written to a software abstraction that is deeply intertwined with the underlying networking architecture. This is illustrated by the fact that applications must currently be responsible for establishing all bindings necessary to perform communication. This requirement causes applications to assume the implicit design, conventions, and operating modes of the underlying networking. As a result, applications are difficult to adapt to new communication mechanisms. For example, email and web addresses implicitly assume a naming structure which requires the use of highly available DNS servers.

We begin with two motivating examples for the problem with current networking state and why it is lacking. Alice and Bob are in a train heading towards the city. Alice wishes to forward

Bob a discussion thread containing a document for review. However, the email may be difficult to send due to absence of any Internet connectivity, or slow and expensive to send as the mail is sent over a cellular link to an email server and then retrieved from the server by Bob's device.

For users Alice and Bob it is not intuitively obvious why it is so difficult or slow to send the email and attachment. Ideally the contents should be sent over a fast mutually supported peer-to-peer technology such as Bluetooth or 802.11 in ad hoc mode. However, even if Alice and Bob mutually configured an ad hoc network between their devices, their email programs would still not be able to communicate across this ad hoc network since email protocols assume the presence of infrastructure services.

Our second motivating example considers Charlie who wants to read some news to pass time in the train, but is either outside of the cellular coverage area or subject to high roaming charges. Currently, Charlie would not even try to use his web browser to read news since he knows there is no connection. However, since reading news is a popular activity in the train, it is highly likely that other people around him will have some reasonably matching cached content available. Unfortunately this information is not available to him.

We observe that the current networking framework is not flexible enough to support applications and mobile users in an intuitive and simple way. The problem is that we need a smart method for selecting the connectivity method, protocols and name bindings appropriate for the connectivity method, and a mechanism for managing the device's communication resources across the various applications on a device. For example, in the case of Alice and Bob, the non-intuitive reason why their email programs won't communicate over an ad hoc link with each other is because email clients assume the availability of DNS services to look up MX records for the domain portion of email addresses, and expect to contact the mail server found in the MX record – both of which are not available in a local ad hoc network. Our goal in Haggle is to provide a networking framework for applications and users that enables these usage models and provides an intuitive mechanism for specifying policies and preferences.

We believe that the user experience should be that of applications adapting to changing

network conditions with devices responsive to different available connectivity options, protocols, and communication environments. For instance, email should be sent peer-to-peer if the sender and recipient are in close proximity, browsers should be able to search neighbours for possible matching content if the Internet is not reachable, and devices should be able to utilize the connectivity of peers willing to provide a bridge to the Internet. Currently, it is non-trivial to add the decision logic into applications to handle these different usage models, and furthermore, if each application makes the decisions individually, the overall system may perform poorly [8, 17].

This chapter presents the design and implementation of the Haggle architecture, which builds upon previous work examining the feasibility of using mobility to form opportunistic networks [36, 76], designing a layerless mobile network architecture, and proposing data objects [69, 77]. We contribute an implementation and evaluation of this architecture, including interoperability support for existing applications.

3.1 Related work

As described in the Background (Chapter 2), many works have independently attempted to address network utilization, ad hoc routing, name binding, and content management. We propose a project aimed at developing a new networking framework for mobile devices with support for operation in any networking environment and maintain backward compatibility with existing applications.

The closest work to Haggle is OCMP [70], which has similar goals in building a node architecture for supporting applications. OCMP is an architecture focused on managing mobile node connections to enable seamless transitions from connected and opportunistic networks. OCMP's architecture incorporates application proxy services on the Internet to maintain application compatibility while the mobile node transitions between different networking environments.

Haggle's motivation is to enable mobile nodes to not only operate across different networking environments, but also enable applications to interoperably function between users in proximity, even under challenged networking conditions. Haggle's architecture internally supports application data introspection and proxy actors at the forwarding, protocol, and data management components. This enables the Haggle architecture to facilitate communication between applications on devices in proximity, even when no or limited infrastructure services are available.

There are also many functional similarities between efforts such as Haggle and OCMP with DTNs [24, 38]. The primary motivation of DTN efforts is on routing, whereas Haggle is focused on providing application interoperability in dynamic networking environments.

3.1.1 Prior work

In this section we outline the basic concepts at the foundation of Haggle, which have been described in previous work [69]. The motivation presented in this previous work is the notion of mobile nodes operating in a "pocket switched networking" (PSN) environment. In this environment, user mobile devices can be in one or a combination of three different networking contexts – able to connect to infrastructure (Internet) services; in a small "island" of connectivity where only local peers are reachable; or completely disconnected but being transported by the user. Mobile devices may transition between these different contexts rapidly.

This dynamic environment presents significant challenges for the current networking stack and networking interface presented to applications. As described in Chapter 2.1, different radio models designed for specific networking contexts may have significantly different operating modes and application requirements. This limitation poses tremendous complexities to an application that intends to operate in PSN environments, let alone supporting more than one application at a time.

The key insight is that applications should not have to concern themselves with the mechanisms of transporting data to the right place. Instead, this should be left to the networking

architecture. Providing this separation of concerns not only simplifies the application logic, but allows them to automatically adapt to new mobile environments and technologies. Haggle proposed the use of a layerless networking architecture where user data, message forwarding, naming, and resource management are exposed to and manageable by the networking layer. Instead of stream-based sockets, applications interact with the network using a data-centric networking model [2].

User data

A core concept of the Haggle layer is the ability of applications to access network resources asynchronously – enabling applications to specify networking operations for the future, or optional depending on circumstance and contention. Because the opportune moment to perform the requested networking operation may happen outside the lifetime of the application process, it is necessary for Haggle to consider the structure and context of the application data to be transported.

Because applications are knowledgeable of what data they need and what data they produce, Haggle exports a data-centric [2] interface for managing persistent data and metadata explicitly. The data interface is designed around the need to make data structured and search-able outside of the application context. This explicit visibility of data and meta-data context enables the Haggle networking architecture to cooperate with applications to deliver, acquire, and manage user and application data.

User data objects comprise of many attributes, each of which is a pair consisting of a type and value. Types and values are typically strings, though some values may also be binary packed representations.

Many have observed that as the volume and interconnected relationships between data grows, the need for search capabilities becomes critical. In addition to the necessity of search to navigate the Internet, desktop search products have also changed the way that many users file and access their data [23]. Haggle further extends this concept by enabling cross-application

and cross-node searching of data by exposing data and meta-data relationships and attributes.

Message forwarding

In order to dynamically react to changing networking contexts in PSN environments, it is necessary to provide Haggle with the ability to choose which interfaces and protocols are best for forwarding user data. Message forwarding may be to send an email message over an infrastructure link to the destination mail server, or it may be over an ad hoc protocol to a next-hop peer who can in turn forward the message to its destination. The decision as to what low-level interface or address to send to can be done "just-in-time", late-bound at the moment of opportunity.

Naming

Current networking architectures require early-binding of names as nested headers found in the front of physical-layer packets. Current dynamic name resolution systems such as DNS still require eager resolution of human-readable names to routable addresses, which then must be bound to the physical address of the transmission interface. Unfortunately this paradigm does not work well in mobile environments where infrastructure services such as DNS might not be available at the moment of the application's request, or the connectivity interface has different resolution semantics such as Bluetooth discovery.

Instead of machine end-point identifiers, Haggle proposes that applications address networking operations using familiar application and user-level names. This provides greater flexibility for applications to specify the destination for a user message, and for the networking layer to determine how best for forward the message. Valid names acceptable by Haggle are any for which Haggle has a protocol that is able to transport a user data message (now or at a later time) using an available and appropriate physical network interface.

Resource management

Network interfaces are shared mediums which consume device and user resources, in terms of time, energy, and cost. To manage and schedule multiple network interfaces, requests for network use from components and applications are delegated to the resource manager as tasks. The resource manager considers the set of outstanding tasks and determines which tasks are allowed to execute by evaluating whether it is beneficial and cost effective when taking into account the user's preferences and policies. The centralized resource management design enables Haggle to schedule network resources asynchronously in a manner coherent with the user's policies and behave in a manner understandable by the end user.

Roadmap

In the remainder of this chapter, we will motivate the problem of enabling networking in dynamic mobile contexts, particularly in social contexts where current applications fail to operate without infrastructure services. We will then highlight refinements above these base concepts in the implementation and evaluation.

3.2 Approach

The rest of this chapter contributes four refinements beyond the original Haggle concepts. First, in addition to exposing application meta-data as key-value attributes in user data objects, we define an interface enabling data objects to be linked as graphs, expressing relationships, dependencies, and organizations. Second, we extend the graph expression concept for data objects to names, enabling Haggle and applications to interact with a richer naming schema. Third, we extend late-binding concepts to resource tasks, naming, protocols, and forwarding algorithms. Finally, we observe that Haggle can extend support to many popular legacy applications by providing a local proxy to bridge the stream-socket interfaces expected by these applications to Haggle's data-centric model.

3.2.1 Late-binding architecture

Hardware manufacturers of mobile devices have attempted to address the dynamic and complex environments that mobile users find themselves in by providing more than one networking technologies on devices. Each of these networking technologies have varying strengths and weaknesses, with different trade-offs between range, power consumption, and monetary cost. The combination of these networking technologies attempts to cover the gamut of mobile networking situations that the user may encounter.

However, current application designs are unable to coordinate and take advantage of these hardware resources dynamically in response to changing mobile networking conditions. Part of this problem is that the current networking stack architecture forces applications to early-bind the communication paths and protocols. While such an approach works well for wired systems and networks, it fails to accommodate needs of mobile devices and environments. Without proper dynamic support from the networking architecture, designing applications for dynamic mobile environments is highly complex and error-prone.

Haggle enables the simultaneous and dynamic use of all communication channels by late-binding (or just-in-time binding) network interfaces, communication protocols, forwarding algorithms, and end-point names independent of applications. This late-binding capability allows the Haggle node architecture to manage the dynamic use of the different networking interfaces, ordering of networking operations based on priority, and balancing the power and monetary costs of the operations with the user and application needs. For applications, freedom from the details of managing dynamic networking conditions allows them to focus on their application logic and serving the user's needs.

In mobile environments, depending on the interface that is selected and the connection context, the necessary protocols for performing networking operations can vary. For example, the SMTP protocol is needed for sending mail messages to a server, but a peer-to-peer protocol should be used for sending to the recipient in close proximity. Haggle allows supporting multiple routing protocols such as peer-to-peer and intentional naming [1], and late binding to these

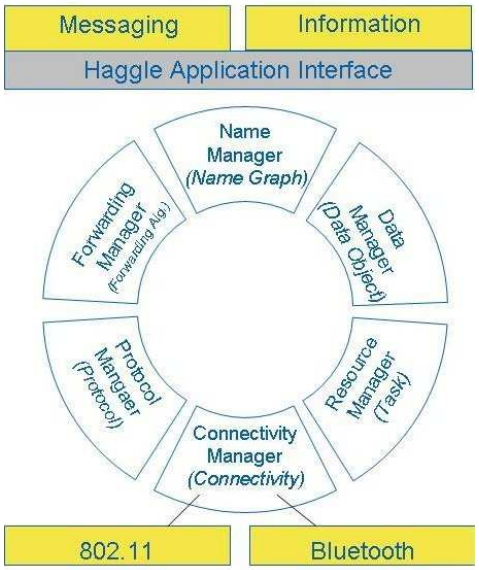


Figure 3.1: Architecture overview

protocols in different environments to automatically adapt applications.

In order to identify services, devices, or individuals, it is necessary to support a naming system that is flexible enough for the different networking environments. Specifically, it is not possible to resolve DNS names in ad hoc environments when infrastructure is absent. Furthermore, different entities along the delivery path may have different name resolution mechanisms. Thus it is necessary to have a flexible and semantically rich naming system which can support late-binding specification of services, individuals, or devices.

3.3 Prototype

Haggle is internally composed of six managers organized in a layer-less fashion (Figure 3.1). The managers are each responsible for a key modular component or data structure (shown in italics in the diagram) - this provides flexibility e.g. allowing for many protocols (e.g. SMTP and HTTP) and connectivities (e.g. 802.11 and Bluetooth) to be instantiated simultaneously. The managers and modules all have well-defined APIs, and each manager (and internal com-

ponent) may use the API of any or all of the other managers. This novel architecture provides necessary and useful flexibility over a layered architecture in which each layer may only talk to the two APIs above and below.

3.3.1 Just-In-Time Binding

Connectivity Interfaces:

Haggle aims to embrace the use of many different networking technologies at the same time. Networking technologies can differ in many aspects, including range, latency, bandwidth, cost, availability, power, and so on. It is therefore appropriate for different connectivity interfaces to be used depending on the particular type of data being sent.

For each network interface on a node, we construct a connectivity instance. For example if there were two 802.11 interfaces there would be two connectivity objects, one for each interface. This is because a connectivity is regarded as a schedulable resource which can consume network time, battery power, monetary costs, etc. As a scheduled resource, all operations that result in network activity, including operations initiated by the connectivity itself, must be delegated for scheduling.

Connectivity objects in Haggle must support a well-defined interface including functionality for neighbour discovery, opening/using/closing communications channels, and estimating the costs (in terms of money, time and energy) of performing network operations. Each connectivity must interface with the underlying driver and hardware to provide this functionality.

Neighbour discovery can take various forms, depending on the connectivity. In 802.11, any node with reception turned on can see beacons from access points which announce their existence. For Bluetooth, neighbour discovery is an active (and time-consuming) process. For GPRS, neighbour discovery is implicit in that when base station coverage is present the Internet is accessible. Delegating the initiation of such operations is an important design approach which enables Haggle to manage multiple interfaces with respect to user defined policies.

The prototype implementation of Haggle focuses on using the 802.11 connectivity because it is a widely used wireless access network and is available for a range of devices from laptops to mobile phones. It also offers both neighbourhood and infrastructure connections (through ad hoc mode and infrastructure mode respectively) which allow us to explore the range of Haggle capabilities using a single connectivity type.

As a schedulable network resource, 802.11 interfaces must provide a cost function, which we currently model in terms of time-on-network cost. For data transfers, the time-on-network cost is calculated per byte, taking into account the bandwidth and size of data. The bandwidth is currently estimated based on perceived connection strength. We used a lower bandwidth estimate for AP mode than ad hoc mode since we expect the access link to be the bottleneck in AP mode. A more ideal implementation could take into account dynamic sampling of goodput and connection history. We leave this improvement for future work. When switching to AP mode from ad hoc, there can be a delay of a number of seconds due to the latency of DHCP to provide an IP address. We model this as a 5 second switching overhead, which we have experimentally determined to be fairly typical.

Protocols and Forwarding:

Haggle encapsulates the late-binding of communication protocols and forwarding algorithms necessary for transporting data. Communication protocols specify the method for point-to-point communication, both for transmitting data as well as opening and receiving connections. For example, the HTTP protocol specifies how to connect to a web server and request objects, while the peer-to-peer protocol specifies how to connect to and receive connections from peers.

When connecting to peer or infrastructure endpoints, the most appropriate protocol is selected just-in-time to perform the necessary initializations as well as transformations and translations in order to send and receive. For protocols which must accept incoming connections, such as a peer-to-peer protocol, the protocol must provide sufficient information to the connectivity interface so that incoming connections can be redirected and properly handled by the

protocol.

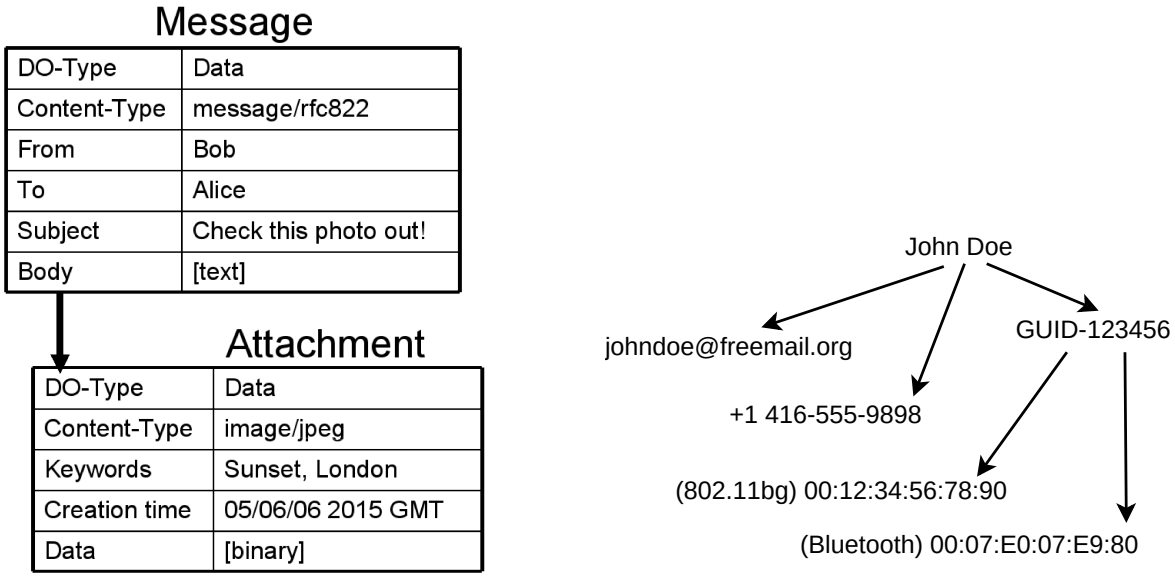
Forwarding algorithms determine the suitability of a next hop for transmission of application and user-level messages. The suitability is presented as a benefit value which enables Haggle to select the just-in-time binding for the forwarding algorithm, communication protocol and connectivity interface. Forwarding algorithms can be active entities, generating and receiving network messages required for maintenance and routing in an overlay or ad hoc network. Such messages, like all network use operations, must also be delegated for scheduling.

Haggle's flexible architecture allows many forwarding algorithms to be in use *simultaneously*. Possible algorithms can range from epidemic [80] to MANET algorithms such as geographic [53] or distance-vector [64], as well as store-and-forward [71, 91] or mobility based [36, 47, 49]. Delegating the proposed actions essentially allows the forwarding algorithms to compete for action. The most applicable algorithm for a given environment will prevail.

We implemented two forwarding algorithms so far, namely "direct" and "epidemic". The direct algorithm only proposes to deliver messages to their destinations if the destination is reachable by direct communication. As a result, the direct forwarding algorithm will always propose a delivery benefit of 100%. The epidemic algorithm proposes to send messages to all immediately reachable destinations. Because the epidemic algorithm cannot be certain if flooding will reach the destination, it will propose a lower delivery benefit value.

Names:

Haggle presents a general form of naming notation that allows late-binding of many user-level names, independent of the lower-level addressable name, as proposed in i3 [75]. We achieve this by using *name graphs*, inspired by INS [1], which are hierarchical descriptions of many known mappings from a user-level endpoint to lower-level names (which may imply particular protocols/connectivity methods). Name graphs are used as recipient identifiers for messages as well as identifying the source node and any intermediate nodes. This late-binding approach



(a) Message and Attachment

(b) Name Graph

Figure 3.2: Example Data and Name Object Graphs

contrasts with the existing network architecture were names are only meaningful at particular layers of the protocol stack.

What's in a name?

An example name graph, which illustrates the provision of many deliverable addresses for communication endpoints is shown in Figure 3.2b. The figure illustrates how one individual, John Doe, can have many different addressable identities, reachable using different connectivity methods. Name graphs span from top-level nodes such as personal names through to leaves comprising persistent methods of reaching them, such as email addresses, but not transient addressing data such as the IP address for the email server. The choice of this partition [43, 44] stems from the feature that any "name" in Haggle can also be an "address" if there exists a suitable protocol which understands that name. For example, an SMS-capable device regards a phone number name as an address, but a non-SMS capable device would not. As a message moves between nodes, different methods of mapping names to transmission methods can become available. Transient names such as looked-up IP addresses are not valid "names" since

they do not provide useful identity information for another node.

Haggle's design allows it to take advantage of any number of existing name management schemes that have been explored in previous work, such as Persistent and Personal Names [30]. In the prototype, we utilize a hierarchical name graph construction. On first startup, Haggle nodes create a GUID name to identify the node itself. Then, the MAC addresses of the node's interfaces are also created as names under the root name. New names can also be learned from applications. For example the names and emails of the sender can be captured from an outbound email and added to the name graph as part of the person's identity. This identity graph is then linked to the node graph to indicate ownership of the device by the individual.

In the prototype implementation, Haggle nodes actively request tasks to contact newly visible names which have no associated node or user identity information. If the peer responds with identity information, it is merged into node's knowledge-base of names. In this way peer nodes can learn identity information even if infrastructure is not available.

3.3.2 Data Management and Data Objects

Haggle exports an interface for applications to manage persistent data and metadata explicitly. Haggle's data format is designed around the need to be *structured* and *search-able*. In other words, relationships between application data units (for example, a webpage and its embedded images) should be representable in Haggle, and applications should be able to search both locally and remotely for data objects matching particular useful characteristics. We draw inspiration from desktop search products (e.g. Google Desktop) which have changed the way that many users file and access their data [23], allowing us to avoid having to methodically place data in a directory structure. We propose that applications can use a combination of structured data and search, with the former providing the kind of capabilities expected of a traditional file-system, and the latter allowing applications to easily find and use data that they themselves did not store.

Data Objects:

Data Objects comprise of many attributes of type/value pairs. These objects can be linked into a directed graph to either represent prerequisite dependencies or ownership information. For example, a photo album's metadata can link to the set of photos in the album, a webpage can link to its embedded objects, or an email can link to its attachments. This explicitly exposes the structured relationship between data objects more richly than directory hierarchies. Applications can also express an "ownership claim" over objects by linking its application object to the desired data objects. For example, a web browser may lay claim over cached objects, and an email reader may claim stored emails. Two examples are shown in Figure 3.2a, representing a message from Bob to Alice, and a photo of sunset. Note that we do not require users to enter more metadata about their objects than applications would require themselves; the value of exposing metadata is in the ability to search and organize data.

Since Haggle allows many applications to claim objects, it does not have a "delete" call. Instead, Haggle implements lazy garbage collection, to allow searching for unreferenced objects and delay space reclamation until it is needed.

Object Filters:

To facilitate searching, the data manager supports searching for objects using a filter object which comprises of a set of regular-expression-like queries over the attributes. For example, a query might include: $(mimetype = text/html \ \wedge \ news \in keywords \ \wedge \ timestamp >= (yesterday))$ Filters can be one-time searches, or made persistent to "watch" for new or incoming matches. Filters can also be made local or remote, effectively providing a "subscription" mechanism.

3.3.3 Scheduling and Managing Just-in-time Resources

Because certain operations are time or sequence critical, there are two types of tasks supported by Haggle: asynchronous and immediate. Asynchronous tasks can be delayed or scheduled by the resource manager at any arbitrary time. Immediate tasks are evaluated right away and a decision for whether or not to execute a task is based only on the current context.

Due to the dynamic scheduling of tasks and potentially changing mobile environment, the benefits and costs of asynchronous tasks can also vary over time. For example, an email checking task is less beneficial if email was last checked 1 minute ago, but more beneficial if over an hour has elapsed. Similarly, as the connectivity environment changes, the costs for operations can dynamically change.

Once a task is being executed, the resource manager can also be asked for an extension on the resource use if the scope of the work being done by the task needs to be increased beyond the initial cost/benefits specified. This is useful for circumstances such as email checking, where we may find a large attachment waiting for download.

The task model is in marked contrast to the traditional network stack, where networking operations proposed by applications or operating system functions are always attempted. The centralization of decision-making about what tasks are worth doing at all, and which are more important at any time, allows Haggle to have a number of advantageous features. First, Haggle can easily and intuitively manage the use of multiple connectivity interfaces. Haggle's support for late-binding protocols and names allows it to manage which subset of connectivity interfaces to use and what kinds of tasks are allowed on those interfaces. Second, Haggle can easily enable dynamic scheduling and prioritizing of tasks. For example, instead of checking email at fixed intervals, the checks can be more often when bandwidth and energy are abundant, and less often otherwise or when there are more important tasks. Similarly, applications are free to request operations of varying priorities, including speculative operations, which are often not possible or worthwhile, but automatically executed when the right opportunity arises.

The current prototype implementation only considers costs in terms of time-on-network,

which provides an estimate of energy consumption. We do not yet support costs in terms of monetary charges per byte or quota limits. Ideally the network interface card or driver would provide power consumption estimates since they have a greater knowledge of their radio characteristics and medium state. We currently assume all nodes are cooperative, and are not using policies which limit interactions with peers.

3.4 Support for Existing Applications

Based on our introductory motivating examples, we have chosen to target email and web as our prototype applications. To be clear, by "email" and "web" we mean the applications, rather than the protocols that underlie them.

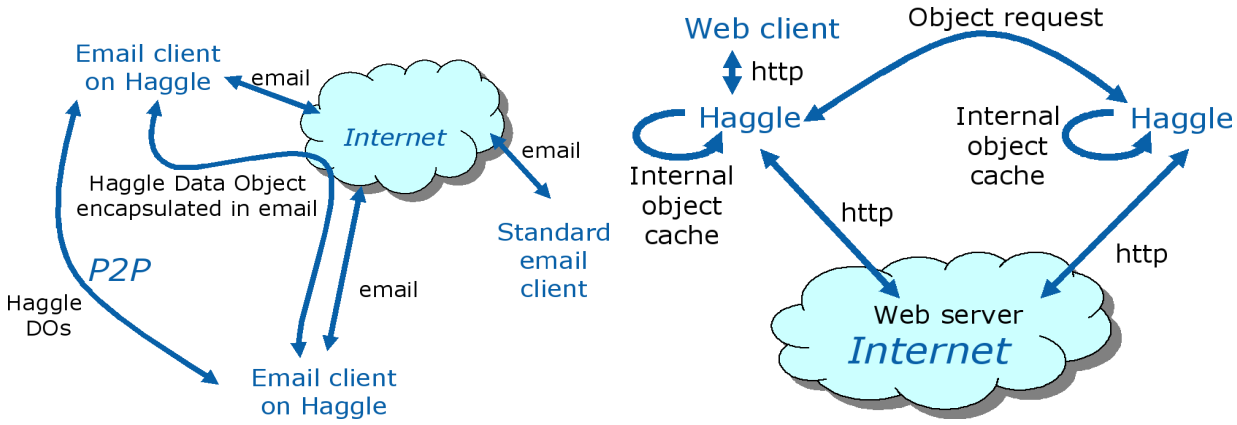
Both of these applications enjoy significant support from the pre-existing infrastructure deployment of servers and content. It is a crucial feature of Haggle that we can take advantage of this infrastructure as well as providing new functionality. This makes Haggle much more compelling to existing users of that infrastructure, and the value added by Haggle provides motivation for its deployment.

To provide legacy support for existing email and web applications, we implement localhost SMTP/POP and HTTP proxies as Haggle-native applications. This allows users to keep using the same applications they habitually use (we have tested Outlook Express, Thunderbird, Internet Explorer and Firefox) with only minimal reconfiguration. We will first describe how Haggle provides support for email, followed by the description of web support.

3.4.1 Email

Supporting email in Haggle consists of two components: an SMTP/POP proxy for interfacing with email clients, and SMTP and POP Protocols inside Haggle that communicate with email servers.

The SMTP proxy accepts emails provided by the user's email client and translates them



(a) Email Application (b) Web Application

Figure 3.3: Haggle Email and Web Applications

into linked data objects using Haggle's API. The proxy uses the recipient field of the email header to search for an appropriate *name* which describes the intended recipient, as illustrated in Figure 3.2b. The proxy then creates a forwarding request to send the mail object to the individual described by the name object. Haggle now will dynamically decide when the message will get delivered, the protocol to use, and over which network interface.

Similarly, when the user's email client checks for new mail, the POP proxy uses the data manager to search for newly arrived messages. New messages are reconstructed as email messages (including attachments) and returned to the email client.

When infrastructure connectivity to the Internet is available and the recipient is not nearby, Haggle will use the existing email infrastructure to deliver the email message. This is possible when the 802.11 connectivity reports that it has access to the Internet. The direct forwarding algorithm and SMTP protocol plugin will both report their ability to resolve the name graph to a deliverable end-point. The resource manager will then determine if there is sufficient benefit to execute the delivery. If so, the direct forwarding algorithm will use the email protocol to transform the message object and use the SMTP protocol to deliver the email using 802.11 connectivity.

If Haggle decides to use a peer-to-peer connection, whether due to lack of infrastructure

availability or to improve throughput, the two peers rendezvous to form an ad hoc network. The sending node then establishes a peer-to-peer connection to the receiver, and sends the message as a Haggle object, complete with all necessary links and meta-data.

3.4.2 Web

The Haggle web proxy operates as a normal web proxy when Internet connectivity is available. As requests are retrieved and returned to the browser, the web proxy stores the information in the Data Manager, including link and object relationships. The mechanism for resolving web addresses and retrieving web objects is similar to the description for the email application, except instead of email protocols, the web plugin is able to understand URL addresses as names and communicate HTTP protocols. If Internet connectivity is not available or too expensive to use, then the proxy creates a filter subscribing to the URL. A notice page is returned to the browser notifying the user that Haggle is attempting to service the request. This page refreshes itself occasionally so that the webpage will be displayed automatically when it arrives.

If a peer has matching cached content, the requested URL and linked embedded objects are sent back to the requester. If a peer is willing or has an incentive [4, 5] to bridge the request to the Internet, the HTTP protocol first downloads the requested URL, parses it to look for embedded content, and downloads the necessary objects. The linked object is then sent back to the requester.

3.4.3 Interoperability

Haggle's architecture enables application data and metadata (through the data object abstraction) to be explicit and introspectable by any application. Thus the document structure can be explored by a uniform interface, rather than requiring applications to code how to parse every supported data type. Conceivably, this enables Haggle to support communication across different application types. For example, two communicating users may be using different

messaging applications (e.g., one using an email client and the other using an instant messaging client), and messages sent to one another would be delivered to their application of choice. An application can also be written which provides web browsers with the ability to browse through email messages as if they were web pages. Such capabilities are possible with the Haggle framework; we leave the implementation and evaluation of such applications for future work.

3.5 Experiments and Results

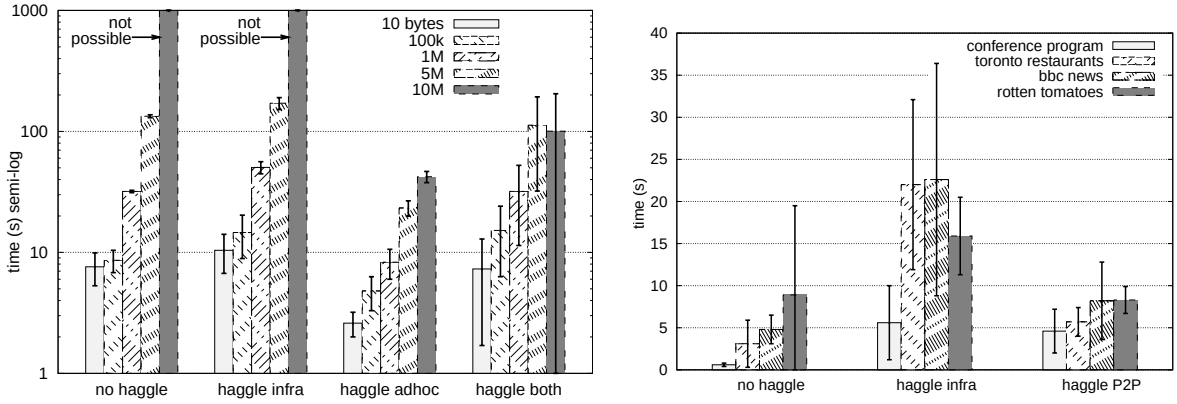
In this section we describe several experiments using the motivating applications described earlier. We provide qualitative results showing the new capabilities that Haggle enables, in addition to quantitatively demonstrating that Haggle's implementation, although not optimized, has acceptable overheads.

The Haggle implementation has been developed using Java J2ME CDC, which means it is compatible with PC and notebook platforms (e.g. Windows, Linux) as well as mobile platforms (e.g. Windows Mobile, Nokia tablets running Linux). This development has been conducted using sourceforge.net, under the GNU General Public License (GPL), available at http://sourceforge.net/projects/haggle.

We conducted the experiments on two laptop computers, which we will call *node1* and *node2*. Both are running Windows XP. Node1 is equipped with an Intel 3945 mini-PCI 802.11 interface, and node2 is equipped with an Intel 2200 802.11 interface. For infrastructure connectivity, the nodes connect via wireless 802.11g to an access point with access to the Internet.

Email:

For the email experiments we created several accounts using the Google Mail (GMail) service. GMail provides POP and SMTP services over an encrypted and authenticated SSL link. This allows us to have one configuration which works from within any network that allows Internet



(a) Mean email end-to-end delivery times. Individual bars indicate attachment size.

(b) Mean webpage retrieval times.

Figure 3.4: Email and web experiment performance. Both graphs show standard deviations as error bars. Lower values are better. Note that in 3.4a for both *no haggle* and *haggle infra* cases it was not possible to send 10M emails due to server limitations. 3.4b does not show *no haggle P2P* because it is not possible to access web pages using existing software.

access. However, there are limitations with using the GMail service. Though there is no limit for the size of email received, GMail restricts the size of outbound emails to be 10 megabytes or less.[1]

For the quantitative experiments we send emails of varying sizes, ranging from 10 bytes (no attachment) up to a 10 megabyte attachment, from node1 to node2. For each size increment seven unique emails are sent over a 3 Megabit download / 800 Kilobit upload DSL link. An automated script is used to send an email from node1 to node2, with node2 configured to check its inbox once every 5 seconds. The script ensures that for every email that node1 sends, node2 must receive it first before node1 sends the next email. This eliminates any congestion effects in the results.

Figure 3.4a shows the latencies for end-to-end delivery of various-sized emails under different network connectivity conditions, both with and without Haggle. The *no haggle* and *haggle infra* clusters provide a comparative baseline between email clients as normal versus using

[1] limit raised to 20 megabytes at time of publication.

Haggle forced to use infrastructure connectivity, respectively. The results show that Haggle imposes a low overhead.

The most important result is shown in the *haggle adhoc* cluster, which shows Haggle sending and receiving emails without infrastructure present. This operation is not possible using the email client alone and would have corresponding graphs of infinite height. The ad hoc transmission of emails, shown by the *haggle adhoc* bars, is fastest since it uses a direct transmission in ad hoc mode. The other modes of operation require use of the access point link, which includes going out the DSL line, through the Internet for both transmitting the email as well as retrieving. We note that having ad hoc transmission ability can also overcome limitations of infrastructure based services, as seen in the 10Mb attachment test. Gmail places a size limit on the mail size, which prevents large emails from being sent.

The *haggle both* bar shows Haggle performing in an environment that has infrastructure access but Haggle has the option to communicate in ad hoc mode when appropriate. Ideally, the *haggle both* performance would be close to the *haggle adhoc* performance. This is not so (though it is still comparable with *no haggle*) and there is a larger variance in the numbers. This is due to interaction between the 802.11 connectivity and the POP protocol. Because an Internet neighbour is visible in this scenario, the POP protocol is requesting tasks to check the email account. This is additional work that Haggle is not doing in the *haggle adhoc* case. Added to this is the significant overhead incurred by 802.11 in switching between ad hoc and AP modes due to lost DHCP request packets. This overhead is not inherent to Haggle, but rather a limitation in the current implementations of 802.11 which can be overcome using techniques such as MultiNet [18].

Web:

In our web experiments we focus on the retrieval of static and dynamic pages from content providers. We chose four different webpages to cover a range of complexities, sizes, and scenarios. All of these sites represent classes of content which users would like to look up and,

in the right mobile context, have a reasonable expectation that other users around might have similarly matching content.

- Conference Program: This page is for a conference's technical program, which is relatively simple consisting mostly of text and no dynamic content, with a transfer size of 64Kb. Attendees at the conference are likely to request this page frequently to see what is on next; however, wireless networks at conferences can frequently encounter connectivity failures [39].

- City Life is a popular city life and culture site with moderately complex layout. The transfer size is 500Kb, sent as 380K of gzip-enabled web traffic.

- BBC news is a relatively complex website with frequently updated content. The transfer size is 370Kb, sent as 100Kb of gzip-enabled web traffic. This page is highly viewed, so there is a reasonable likelihood of a copy being present in a group of users.

- Rotten Tomatoes is a movie review site which contains a dense layout with dynamic content. The transfer size is 834Kb, sent as 230Kb of gzip-enabled web traffic. This might be looked for at a cinema while deciding what to watch, with a reasonable expectation that others in the area already looked it up.

For each of the experiments we retrieve the contents seven times, each time clearing all caches. We measure the end-to-end time as starting from the moment of request at the browser until the browser finishes loading and rendering all content on the page, using the Firefox web browser with the FasterFox plugin since it contains a built-in page load timer (we turned off all other functionality that FasterFox offers).

Figure 3.4b shows the performance results for retrieving the above described webpages with and without Haggle. Between *no haggle* and *haggle infra* the comparison is less favourable than for the email case. We attribute this to the overheads of (a) the time taken to parsing the HTML pages to determine linked data objects, (b) the overhead in the proxy approach, since

the web client opens and closes multiple socket connections to inform Haggle of different objects it requires, (c) inefficiencies in our Data Manager implementation in that the webpages are stored data persistently before they are transmitted to the web client.

For each web object retrieved from the Internet, the web proxy attempts to reconstruct its relation with other objects it was linked from. Because web browsers make multiple simultaneous connections to the proxy and use each pipe in parallel, we must examine the headers of the objects returned in order to properly associate objects to webpages. To do this the web proxy examines the referrer tag of the HTTP response message for the retrieved object to determine from which other object the current was requested from. After finding the originating object, the web protocol creates a link from it to the newly retrieved web object. This search and link requires queries to the data manager in Haggle which adds overhead time to each web object retrieved, visible in the comparison between *no haggle* and *haggle infra*.

For the *haggle P2P* experiments, we have node1 configured to enable access to the access point as well as communicate in ad hoc mode. The four webpages described are then visited using the web client on node1 so that it has a cache of data objects representing those pages and embedded objects. At this time, the access point is turned off, modelling node1 being moved to an infrastructure-free location. Node2 is only able to communicate in ad hoc mode, and is placed near node1. We request a webpage on node2 (clearing the cache each time an experiment is run); since node2 does not have Internet connectivity, it sends a filter requesting the webpage to node1, who returns the matching webpage with embedded objects. This last experiment shows fundamentally new functionality enabled by Haggle for the web browser, in that it can now operate even when there is no infrastructure Internet connectivity..

3.6 Limitations

In this section we highlight a few limitations of the current Haggle prototype.

First, the current implementation enables Haggle to make asynchronous late-binding de-

cisions for tasks based on a cost-benefit analysis of the current mobile context. However, the ability to make this determination is dependent largely on the policies that are in place and the user's ability to specify and tailor the policies. Policy construction and management are an active area of research and not accounted for in this initial prototype.

Second, the current task organization of logically related network operations are suitable for discrete units of content such as web pages and emails. However, it is not clear what extensions can be made to further support streaming connections such as online videos and VoIP calls. At this stage we do not have clear answers to such questions and defer the investigation for future work.

Third, the current prototype supports only the WiFi interface in infrastructure and ad hoc modes. Adding support for the Bluetooth interface would be straight-forward with no foreseeable difficulties. In terms of implementation, the complexities involve working around different API designs for various mobile platforms. For example, the WindowsMobile platform only supports RS232 serial transfers for peer-to-peer data transport. Furthermore, the limited thread scheduling capabilities of the mobile platform requires Bluetooth code to be implemented in a different process, not just thread, to prevent IO blocking. Otherwise, IO-blocked Bluetooth operations would block the entire Haggle middle-ware process.

The lack of a Bluetooth implementation in the current prototype leaves a significant gap in our ability to model and demonstrate functionality with a longer-range, slower, and monetarily more expensive interface such as cellular. While we are confident that the architecture will easily enable the addition of such an interface, the significant challenge is with the proper policy rules for dictating its use.

Finally, a major premise of this current proposal and prototype is that peers are inherently non-malicious and cooperative. Dealing with non-cooperative peers, privacy preservation when peer-foraging for web content, and securely communicating with peers in an ad hoc environment are all challenges which we leave for future work.

At the hardware level, a fundamental technique for spoofing the identity of the peer device

is with hardware address spoofing. With WiFi this is easily done since MAC address spoofing is a supported feature on many WiFi interfaces. Other radio platforms are more difficult to spoof, such as Bluetooth, where the hardware address is burned into the device. However, there exists hardware platforms, either by design or by flaw, which do allow spoofing of Bluetooth addresses. We envision addressing this problem from higher-level security protocols for ensuring the identity of the peer device. A possible technique is to use radio fingerprinting to prove proximity [82] when communicating with new peers with whom no previous identity information is available. Known peers can be checked using more standard techniques such as checking against known public/private keys previously exchanged. Identity keys could also be exchanged out-of-band using alternative, more trusted links, such as SMS, where the monetary cost to the communicating parties is manageable, and more difficult for spontaneous attackers to access.

Chapter 4

Copernicus

People are increasingly making their personal content available on the Internet by hosting it on a variety of web applications. Extremely popular web applications include FlickR for photos, Google Calendar for scheduling, and Imeem for sharing music. This has resulted in a situation where users have two modes of interaction: traditional face-to-face interaction in the real world, and computer-based asynchronous interaction on the web (e.g., commenting on a friend's photographs). Our aim is to enable people to socialize simultaneously in both modes by providing seamless access to the user's web content during face-to-face interaction.

Consider the current situation. Say Bob is a supplier who is visiting a group of people at a prospective client company. Bob has a portfolio of web hosted content, including documents, videos, CAD files, and other media, that he wishes to share with his clients. Some of this content is sensitive, hosted on Bob's company's website, and requires authenticated access. Yet other less sensitive promotional content is hosted on third-party web applications, such as a media streaming service, where for example, a "list of friends" allows accessing content. Providing access to these relevant materials is not a simple matter of emailing links and attachments; logins and usernames must be determined or created, access permissions granted, and directions for where and how to access this content provided. This complex set of operations are cumbersome and disruptive during a face-to-face meeting, especially if this is the first

introduction between Bob and many of the clients. To mitigate this problem, Bob may bring in mass-storage media with the content pre-loaded. However, now the onus is on the clients to load the media and ensure they have the proper software to view it. This approach is still problematic if the clients primarily use smartphones.

Continuing the previous example, suppose that Bob decides to setup a follow-up meeting with the clients. Though this collaborative task is conceptually simple, it involves users focusing on their calendars on their mobile devices, followed by several rounds of verbal interaction in which people suggest their free times and others who respond with their busy times, and then all users updating their calendars. These examples illustrate that while sharing content and collaborating in social settings is desirable, it is cumbersome and rarely done. Instead, these tasks are done later, and may involve multiple rounds of emails for coordination.

In this chapter, we present Copernicus, a mobile system that lets people socialize at both the physical level and on the web by providing a bridge between the two worlds. Copernicus introduces a collaboration model in which users in proximity are represented as first class objects in the system. These *user objects* encapsulate the users' web applications and content. This abstraction provides users with an intuitive model for finding and sharing content and collaborating with nearby users. The result is a computing experience in which digital content is seamlessly made available to users interacting in the real world, instead of forcing users to focus on their mobile devices and retreat away from the social interaction in order to navigate the web.

Revisiting our example, Bob uses his Copernicus-enabled mobile device to access the web content that is to be shared. He then selects a Copernicus specific "share" option from the menu, and is presented with a pop-up of thumbnails of the people around him. Bob selects the clients, and confirms the sharing operation. The clients find a dynamically generated directory containing the files that Bob has shared on their own favourite documents web site (e.g., an internal web site or Google documents). Note that Bob does not require the clients' email addresses, and the clients do not have to login to Bob's web sites to access the shared documents.

In fact, they may not even be aware of where the files are actually hosted. Similarly, the participants set up the next meeting time by opening their calendar on their mobile devices and sharing all their free times with others in the room. Then, Bob sees a highlighted entry two weeks from now in his calendar, indicating that everyone is free on that date and time. Bob selects that entry, books the time, and the clients' calendars are all updated with that entry. Bob shakes hands and looks forward to the next meeting.

Copernicus makes two main contributions that enable these collaborative interactions. First, it provides support for user objects that represent users in proximity. Copernicus detects individuals in the proximity of a user by mapping low-level device identifiers such as Bluetooth MAC addresses of mobile devices to real individuals, and then mapping individuals to their web applications, login ids and their content. The user objects thus encapsulate a user's applications and content. This approach allows building new web applications that can take advantage of these objects for seamless sharing and collaboration. Second, Copernicus provides an adaptation service that retrofits existing web applications to enable sharing and collaboration with nearby users. This service takes advantage of the programmable interfaces available in modern web applications.

We have built and evaluated a prototype of the Copernicus system using smartphones. Our prototype includes support for several applications (calendar, email, photo and file sharing) from different web providers, thus showing the generality and wide applicability of our approach.

4.1 Related works

Several existing projects have explored various aspects of adding real-world proximity context information to generated content, as described in Chapter 2. For example, INCA [78] tags contextual information to content generated in a mobile environment; Puppeteer [46] tailors existing applications using their exported programmable interfaces; and OPA [6] attempts to

tailor web content to suit constrained mobile interfaces.

Copernicus presents a comprehensive framework which not only enables the functionality of these existing works, but also enables extending these customizations to third-party web applications, at their source, using exported Web2.0 interfaces. This novel framework enables users to enjoy context aware enhancements without needing to switch to a new application interface. Furthermore, by allowing content to be composed from multiple web applications, Copernicus enables new classes of applications to be built on top of existing web applications and services.

4.2 Approach

Copernicus simplifies web-based sharing and collaboration during face-to-face social interactions by providing support for user objects that represent users in proximity. User objects have a human friendly representation, as shown by the list of face-shots in Figure 4.1b.

Alice can share some of her content by simply selecting her content, clicking on the share button as shown in Figure 4.1a and choosing the face-shot of the receiving peer. Conversely, Bob can easily find content that has been shared with him, as shown by the links in Figure 4.1c. Bob can also filter content by peer by using a face-shot selection interface. Under the covers, Copernicustakes care of the intricacies of identifying users in proximity, sourcing content from the origin web applications, enforcing access control, converting between data types supported by different web applications (e.g., Picasa and Facebook), and presenting the shared content in an intuitive manner.

The rest of this section describes the two main ideas, user objects and an adaptation service, that Copernicus uses to support face-to-face web sharing. We then describe the range of sharing capabilities and applications enabled by Copernicus.

(a) Share menu option (b) Peer selection (c) Finding content

Figure 4.1: *Sharing and finding content. Alice initiates content sharing by using the share menu option or by using an embedded link tagged with the content. A peer can be selected easily by clicking on a face-shot (or by typing in a cellular number). Bob can find shared content by clicking the different links, each of which implements a different filter.*

4.2.1 User Objects

User objects represent users in proximity and they encapsulate a user's web applications and web content. Copernicus provides this abstraction by using three mechanisms that are discussed below.

Identify users in proximity Copernicus needs to detect individuals in the proximity of the user. It can leverage different techniques, such as a GPS-based location service, to inquire about users within a specific geographical area. Our existing prototype leverages short range radio (e.g., WiFi, Bluetooth) to detect the MAC addresses of other mobile devices in close proximity. Short-range wireless radios are available on most smartphones today and they provide good accuracy for determining physical proximity. Next, Copernicus maps the low-level hardware identifiers to user objects. This mapping is provided by users when they initially register with the system. The eagerness with which Copernicus updates the user's proximity depends on user preferences and applications, as discussed later. Copernicus can also keep a log of users that have come within proximity of the user for later use (e.g., to simplify sharing with someone we met earlier in the day).

Determine the user's web applications Copernicus is aware of each users' web applications. This mapping is also provided to Copernicus by users when they register with Copernicus, and it allows linking a user object with a user's applications.

Enable access to web content The last aspect of a user object is enabling access to the user's web content. Copernicus aims to provide seamless access to web content, across different web applications and with different user memberships. Meeting this goal places two requirements: access control and content interoperability.

Copernicus accesses user content using delegated credentials, a widely available feature of Web2.0 applications for enabling third-party access. Copernicus is provided with these credentials (unique tokens) when users register their web application. Copernicus can then make requests to access the application content on behalf of the user. Requests are signed with public/private key certificates and the web application checks the IP address of the request to ensure that it originates from Copernicus. We discuss the security concerns with credential delegation later in this section.

When users share content, Copernicus attempts to use the native access control settings of a web application whenever possible. However, these settings may be insufficient due to user privacy settings, insufficient granularity in the application's access control mechanism, or lack of common service membership between users (e.g., the peers do not use the same web application). In this case, Copernicus securely proxies content using the delegated credentials of the content owner, thus enabling access control at the user object granularity. For example, Alice can share a specific document with Bob, even though Bob does not have an account with the service used by Alice.

Seamless content access also requires content interoperability between different web sites. Copernicus provides a framework and an API for translating content into a per-application class intermediate format. It is important to clarify that Copernicus is not a content repository. Instead, its API enables content to be linked and shared across web applications.

4.2.2 Proximity-aware Adaptation Service

The user object abstraction described above allows building new web applications that provide seamless sharing of web content and collaboration during face-to-face interactions. However, we would also like to retrofit existing web applications to achieve the same goals. Our solution is to provide a service that utilizes the APIs provided by modern web applications to adapt these applications.

For example, the Copernicus adaptation service allows filtering, searching and sorting of web content. As the amount of user-generated and shared web content increases, these operations become burdensome in a mobile setting, even though they may be as simple as marking a piece of content to be shared, or vice versa, finding a piece of marked content.

The adaptation service adapts the organization and layout of content based on the user's current and past proximity context. We apply this approach to both the user's own content as well as shared content. In the introductory example, when Bob opens his calendar to arrange a follow-up meeting with his clients, the adaptation service inserts temporary "free" entries that match the intersection of available times of all participants.

The adaptation service provides two interfaces. It can serve as a web portal that provides access to the adapted content. This Copernicus portal presents a series of links containing the adapted content from both the user as well as others. For example, when Alice visits the portal, there is a link that shows the content she has shared with the peers in her proximity, a link that shows the content of other people in nearby proximity, and a link that shows the content that has been shared with or received by Alice. These links provide Alice with a convenient and fast way to find and see the content that Bob has shared with her. For example, Alice can access Bob's shared photos and videos hosted on Facebook and YouTube, without even knowing the web applications that Bob uses.

The Copernicus adaptation service provides a second, more intuitive interface by directly adapting the web application. In this case, users directly interact with their application rather than visiting the Copernicus portal. This approach has the added advantage that it leverages

the efforts of web application providers in creating custom clients and interfaces for mobile devices. For example, say Bob has taken several photos at a conference and shared them with Alice. When Alice visits her Picasaweb, she finds a virtual album inserted by the Copernicus adaptation service containing thumbnails of the shared photos. Opening the thumbnail photo provides Alice with a link to the full-quality photo, with permission settings automatically set up by Copernicus. Figure 4.4b provides a screen-shot of how Bob's photos on Facebook can be seen by Alice, despite Alice not having a Facebook account.

4.2.3 Sharing Policies

Copernicus enables a range of sharing models, based on the person sharing the content (*sender*) and the user which whom the content is shared (*receiver*). Applications and tasks may require explicit interaction from both the sender and receiver, or just sender or receiver, or be completely automated without requiring any user interaction.

Copernicus provides an intuitive point of contact for initiating the sharing of content. However, we note that users are able to share content with others who are not in their physical proximity, for example people they have met previously. Whether shares persist is application specific; some applications may enforce share permissions to be proximity-based, while others may allow shares to persist until revoked by users.

Explicit sender/receiver interaction

In this model, the sender explicitly marks the content to share and the individuals with whom to share (e.g., by clicking on their picture on her mobile device). Similarly, the receiver explicitly chooses who to receive content from. Many applications fit into this model: sharing of documents, photos, or music, and scheduling meetings.

To help filter friends from strangers when sharing content in a crowded space, Copernicus can utilize relationships available in social networking applications. In our example, suppose Bob wanted to share some of the photos he took with Alice. When he initiates the sharing

operation, Copernicus presents a list of individuals with whom he can share content. Eve, a stranger, is nearby but does not show up on the list because she is not in Bob's social network.

Though this explicit model requires user interaction, it is less taxing for the user than existing approaches. Instead of working with device identifiers and machine names, users are able to interact much more naturally by specifying their peers as people instead of arbitrary IDs.

Implicit sender or receiver

An example of an application that provides a mixed mode of operation, where user interaction is required from only the sender or receiver, is a business card exchange web application. Unlike existing business card formats such as vCards, which are restricted in content and format for size purposes, a business card application on Copernicus can easily exchange arbitrarily large and rich portfolios since the sharing is performed by well-provisioned servers. The client devices (in this case smart-phones) do no work.

Imagine a pink-slip party, where individuals who have recently lost their jobs can network and commiserate, and recruiters can look for potential employee candidates. In this setting, a user looking for work can set his portfolio to be available automatically to anyone interested. Conversely, a recruiter may wish to be more selective with which portfolios to accept. In this usage mode, the sender is offering content without requiring user interaction. However, the receiver chooses to selectively and interactively accept content. At the same time, potential employers may also be proactively recruiting well-qualified candidates, offering them portfolios highlighting the benefits of working at their company. The candidates happily accept all such solicitations from potential employers. In this usage mode, the sender selectively decides who to send content to, but receivers automatically accept the shared content without user interaction.

Implicit sender and receiver

On the opposite end of the spectrum is a usage model where the application implicitly shares and accepts content on behalf of users. There are many possible applications which may opt for such a usage model. For example imagine an application that enables Alice to take a tour of a city, without taking any photos of her own, and then later be able to build a photo-album from the public photos of the people she took the tour with or even random strangers who happened to be in proximity.

4.3 Architecture

Figure 4.2 shows the main components of the Copernicus service: the Copernicus client running on the users' mobile devices, the Copernicus service, and various third-party web applications, such as Facebook and Picasaweb. The example illustrates how Alice can access photos that Bob has shared with her. These photos are available to Alice, without her needing to know that Bob uses Facebook, the albums in which his photos are stored, or requiring membership to Facebook to gain access.

4.3.1 Copernicus Mobile Client

The Copernicus mobile client consists of a web browser plugin that adds a share menu to the browser and scans the local radio environment to help identify users in proximity. The current implementation scans for nearby Bluetooth addresses and collects cellular and WiFi radio fingerprints. The device sends the results to the Copernicus service via a secured connection through an infrastructure-based Internet link, e.g., WiFi or 3G. This client is necessary because acquiring this information requires device-specific code.

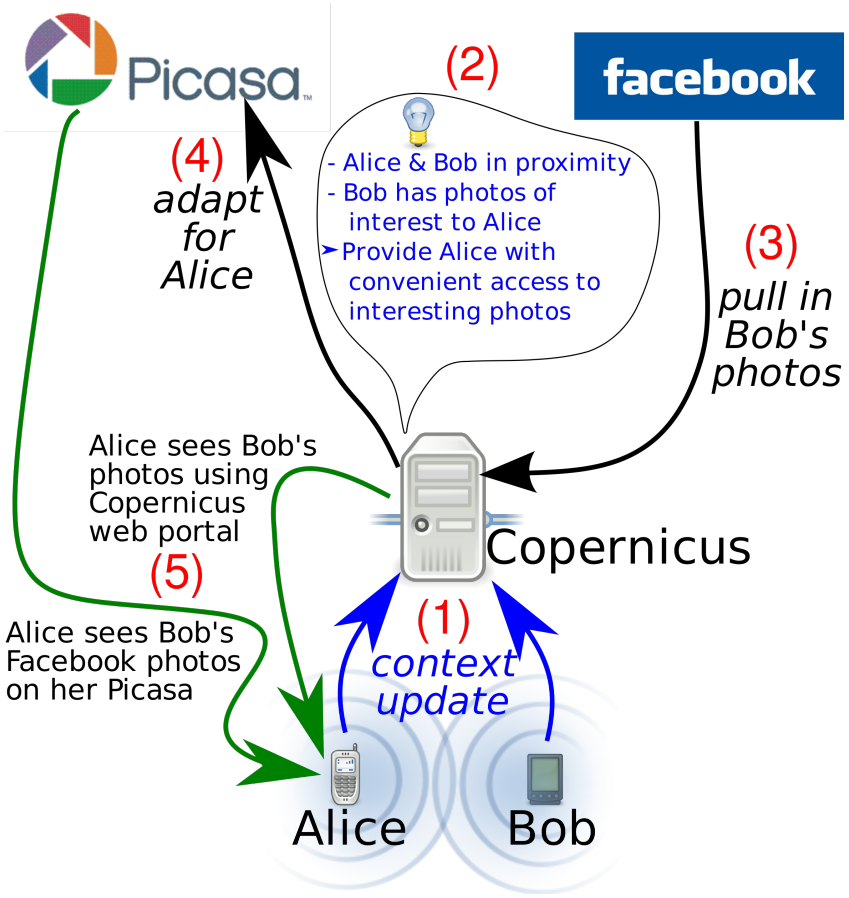


Figure 4.2: *The Copernicus Service*

Sharing

Copernicus provides three methods for initiating a sharing request. When supported by the application, users can select a *share* link embedded within the application content. Alternatively, users can install the Copernicus browser plugin and click on the *share* menu item it adds to the browser menu (Figure 4.1a). Finally, as a last resort, users can select a *share* link from Copernicus's web portal.

Once Copernicus has determined that sharing is allowed, the user's browser is redirected to a peer selection page hosted by Copernicus, shown in Figure 4.1b. Here, the user is presented with a list of individuals with whom to share content. There is also an input field for the user to manually enter a cellular number, in case the peer is not a registered Copernicus user. Once a peer has been selected, Copernicus sets up the appropriate access permission settings and redirects the user back to the originating web application.

Bootstrapping and notification

Copernicus utilizes SMS notification for two functions: explicit user identification and bootstrapping of new users. We chose to use SMS because it is a well supported feature on cellular devices and is operated by trusted carriers. Furthermore, exchanging cellular phone numbers is a socially accepted practice amongst acquainted individuals.

For a user wishing to remain "invisible" as well as users who are not registered with Copernicus, a peer wishing to initiate sharing can do so by entering the user's cellular number in the recipient selection page described in the previous section. Copernicus then creates a new anonymous user account, establishes the sharing permissions for that account, and creates a unique URL hosted by Copernicus with which to access the shared content. This URL is then sent to the recipient's cellular number via SMS. The recipient is then able to access the shared content via the URL. The recipient user can merge this anonymous account into their own Copernicus account at a later time via the Copernicus portal. Similarly, a new user can register on Copernicus and integrate this anonymous account.

4.3.2 Copernicus Server

Figure 4.3 shows the architecture of the Copernicus server. The adaptation service implements application-specific policies and interfaces with the Copernicus API. This API covers a range of popular user applications, such as calendar and photo sharing, and specifies common operations and data types for these applications. Copernicus includes driver modules that interface directly with specific web applications, and implement the code to connect with the Copernicus API. For example, there is a Facebook Photo driver and a Google Picasaweb driver, which both implement the Copernicus photo API. The Copernicus APIs define more than just data access functions. They also define operations for reorganizing content, managing dynamically created content, and useful hot-links for initiating content sharing operations. How these operations are manifested depend on the specific web application driver implementation. In addition, the Copernicus server also contains a name mapping module that maps individual users to their devices and their online identities, a proximity detector, and a credential lending and access control module. The name mapping module exposes individuals as user objects via the Copernicus API.

Adaptation Service

The adaptation service implements application policies as Copernicus-native programs. These programs specify domain specific algorithms for how content should be filtered, sorted, organized, merged, and shared. Copernicus supports two primary modes for how these content management decisions are applied: a *now* mode, and a *past* mode. The *now* mode applies policies based on the user's current proximity context, as determined by the proximity detection module. The *past* mode enables the user to apply policies based on accumulated previous proximity context. For example, suppose Alice forgets to schedule a follow-up meeting with Bob. Alice can effectively "turn back the clock" to her meeting with Bob, and see the adapted policies as if Bob were in proximity.

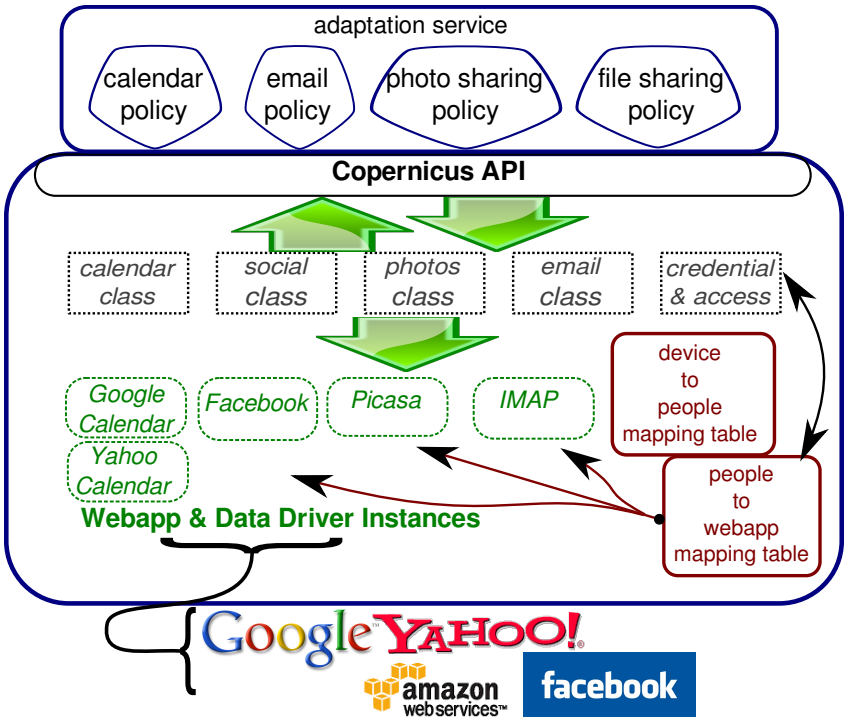


Figure 4.3: *The Copernicus architecture*

Web application classes and drivers

The web application interfacing subsystem has two main components: a library of application domain classes and intermediary data formats, and web application-specific drivers which implement these interfaces.

Copernicus uses drivers to cope with the application-specific nature of the APIs exported by the different web applications. Drivers must implement one or more application class interfaces, which basically encompass four core functions: pull content, push content, manage access control policies, and insert user interface extensions. Due to the diversity of features and capabilities across web applications, drivers are free to determine how the content organized by Copernicus is to be applied. For example, to add dynamically created calendar entries, the Google Calendar plugin may create supplementary calendars for the entries, which can then be overlaid atop the user's primary calendar. A plugin for Yahoo! Calendar, which does not support secondary calendars, would add the new entries directly into the user's primary

calendar.

To minimize the transfer of content between the web applications and Copernicus, a driver can implement an optional interface for allowing filter expressions to be pushed to the web application. This enables significant filtering and selection operations to be evaluated locally on the web application's system. Many web applications support some form of search or query interface, which can filter across different content types (e.g. FBQL [27]) or tag and location attributes (e.g. FlickR search [29]).

To support operating on data formats from many different web applications, Copernicus's application class interfaces define an intermediary format for every supported data type. Copernicus specifies intermediary formats for images, calendars, emails, social networking relationships, and meta-data such as geo-tags and radio signatures. This common set of formats enable the filtering system to work on content independent of their originating or or destination web applications.

Name mapping

Copernicus must solve two important questions: who is the individual that owns a given mobile device, and where are that individual's online identities and resources? To provide these two mappings, Copernicus's name mapping module assigns each individual a unique user account with which devices and online identities are associated. Human and socially identifiable attributes such as a name or profile photo can be directly specified by the user to Copernicus, or the information can be extracted from the user's other services such as Facebook. Copernicus's name mapping module assumes that devices, device identifiers, and online identities are owned by a single person, though a person may own many devices and many online identities.

Proximity detection

The proximity detection module detects nearby peers by processing the radio environment updates submitted by the Copernicus mobile client. The proximity module appends the received

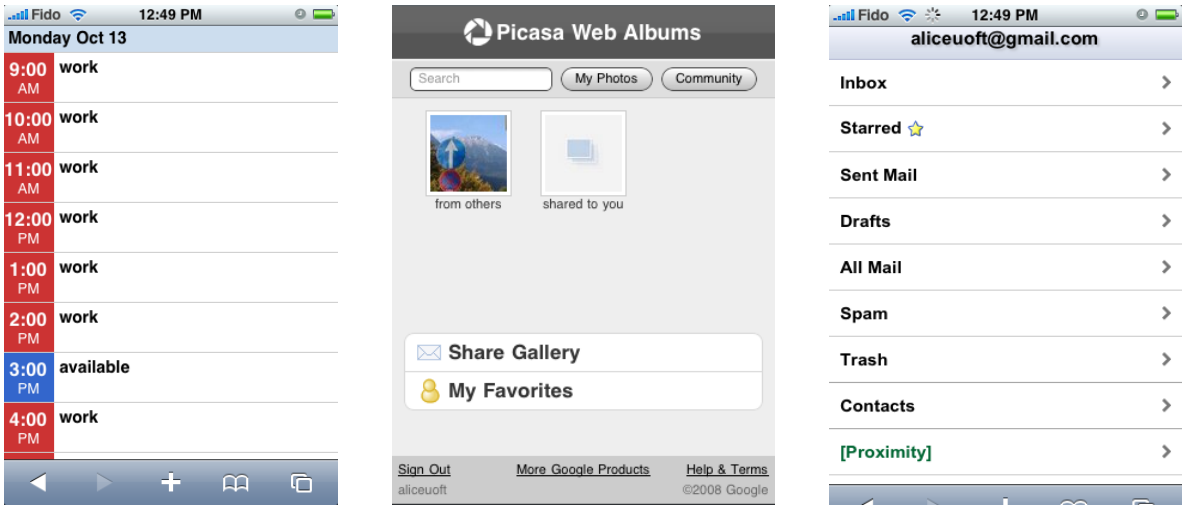
updates to a ring buffer, and supports multiple proximity detection modules, by allowing each of them to scan over the update buffer and raise notifications for proximity changes or updates. The proximity module then determines how to weigh the notifications from various modules, depending on the user's preferences. The Copernicus architecture supports proximity modules based on a variety of sensors, such as GPS readings, WiFi and cellular fingerprints [7, 19, 81], and Bluetooth scanning. The framework can also accommodate approaches, such as Amigo [82], which detect proximity based on similarities in the radio signal fluctuations perceived by nearby devices.

Credential lending and access control

Copernicus uses credential delegation, a widely available feature on Web2.0 applications, to allow third parties to make requests on behalf of a user. This credential lending module is used for three purposes: to sort and filter content on behalf of the user, to retrieve content for purposes of sharing between users, and whenever possible, setting access permissions for content hosted on webapps.

The access control module in Copernicus determines which of the filtered content can be accessed by the receiver. By default, Copernicus extracts and utilizes the settings from the user's webapps. This provides the user with an already familiar access model. The exception is when users explicitly share content with specific receivers. This fine-grained explicit sharing over-rides the default rules.

In many cases web applications are unable to grant this fine-grained explicit sharing request due to two possible limitations: insufficient access control granularity, and the lack of mutual membership on the web application between sharer and receiver. Copernicus's access control module tackles both of these problems by serving as a secure content proxy. When providing access, receivers and the receiver's web application are given uniquely generated URLs hosted by Copernicus. Every time these unique URLs are accessed, the access control module first checks the identity and permissions of the accessing device. If permission is granted,



(a) Calendar (b) Photo (c) Email

Figure 4.4: *Copernicus applications. The Calendar applications shows the union of busy times. The Photo and Email applications show a dynamically "Proximity" folder or gallery with pictures shared by people in proximity or emails from people in proximity.*

Copernicus securely retrieves the shared content, and proxies access to the receiver.

4.4 Implementation

The prototype Copernicus system is implemented using the Apache Tomcat Java Servlet framework. For user devices, we implemented support for the Windows Mobile 5 smartphone platform. In addition, an initial implementation for the Apple iPhone platform is partially complete as of this writing. In the remainder of this section we provide a brief description of the implementation details of the prototype applications. We show the generality and wide applicability of Copernicus by implementing applications from different domains and web providers.

Calendar

The calendar application aims to simplify the task of finding common free times for establishing future meetings. This task can become especially time consuming as the number of

participants increases and the complexities of schedule times and conflicts result in burdensome communication overhead.

We implement support for the Google Calendar service. The calendar application adaptation policy produces the set of intersecting free times. This set is given back to the plugin which pushes this result back into the users' Google Calendar (see Figure 4.4a). This implementation creates a temporary calendar which can be overlayed on top of the user's primary calendar. To mark these entries as Copernicus-generated, the description fields for the entries are tagged with a hash check-sum generated from the event details. This enables Copernicus to find and update these entries later when the user's social proximity changes. The description fields also contain links to common actions, such as reserve a meeting at this time slot. This is accomplished using a link to a uniquely-generated URL on Copernicus which triggers the action.

Photo Sharing

The photo sharing application aims to enable users to find and share photographs of interest between people in a social group. The Copernicus photo application automatically constructs virtual photo albums for sorting a user's own content, as well as displaying content shared to the user by others. This feature works across different web applications, enabling users to find both their own content as well as shared content from their preferred service without having to figure out where the peer's content is stored or how to access it. As the user's physical proximity changes, Copernicus automatically manages these virtual albums, adding or removing photos. An example of this can be seen in Figure 4.4b, where Bob has shared a photo from his Facebook album with Alice. Alice can see and access the photo from her Picasaweb application, without needing Facebook membership.

Copernicus provides photo sharing functions through Picasaweb using the Google APIs, and Facebook photo albums using the Facebook API. The current photo application has two policies: one which selects the set of photos Bob took when Alice was in close proximity, and

one which selects explicitly shared photos. These filter results are instantiated as virtual albums populated with thumbnails of the selected items, all managed by Copernicus. Appended to each thumbnail is an identifying hash-tag and full-image link, hosted by Copernicus which enables Alice to see the full sized original.

Email

The email application aims to make it simpler for users to quickly and easily find relevant related correspondence between the user and their peers in proximity. Existing solutions for finding email include manual sorting into folders or tags, using textual search, or sorting by date or sender. All three of these methods are less than smooth in a mobile environment. Typing a search term may be convenient on a desktop, but is clumsy on a mobile device. Finally, sorting by sender is often too coarse-grained. For example, people can belong to different project groups, all with their own threads of correspondence. Sorting only by a specific sender does not properly distinguish between different project threads, implicit in the list of included recipients.

The Copernicus email application creates a virtual folder sorted with email correspondence from the included participants in proximity, as illustrated in Figure 4.4c. Unlike simple sort-by-sender, Copernicus email can sort by all-senders-in-proximity, which is much more fine-grained and selective. As the user's physical context changes, the messages in the folder are dynamically updated.

Despite the widespread popularity of web-mail applications, there exists no Web2.0 APIs for manipulating and managing email. However, a basic set of adaptations can be performed using the IMAP protocol, which is almost universally supported by email services. In our implementation we use Google Mail service as our IMAP email back-end, though nothing in the IMAP plugin implementation's design is specific to Google Mail.

4.5 Evaluation

In this section we evaluate the performance of our Copernicus prototype. The mobile device used is a JAQ3 Windows Mobile, manufactured by HTC, which is an EDGE capable GSM phone with WiFi, color touch-screen, and hardware QWERTY keyboard. The Copernicus server is a quad-core Xeon 3.6 GHz with 4 gigabytes of RAM, running Apache Tomcat.

For each application introduced in the previous section, we profile the interaction requirements to achieve common tasks using Copernicus, and compare them with existing models for sharing content. Tasks are profiled anticipating an expert user familiar with the mobile device and applications at hand. The metrics used are from the KLM variant of the GOMS model [14, 15]. Due to the compact design of the mobile device, we only consider keystroking and mental operator components. We assume 0.20 second timings per key hit, which is the KLM measure for an average typist achieving 55 WPM, a rate anecdotally achievable by some expert BlackBerry$^{\mathrm{TM}}$ users. Mental preparation before key sequences accounts for 1.35 seconds. Because these are web-based applications, we also consider network speeds for retrieving content and rendering it on the device. We factor 1.0 seconds for system response time, which we conjecture is reasonable for wireless networks and mobile processing power.

It is noteworthy that GOMS models are good at estimating perfect expert user performance under perfect ideal conditions. Unfortunately, perfect ideal conditions practically never occur under many wireless communication situations evaluated below. For example, when attempting to send content peer-to-peer over Bluetooth, the sender must first find the receiver's device. This requires first a Bluetooth inquiry, which first looks for devices in vicinity, followed by a Bluetooth scan to retrieve the human-readable string name for the device. Since Bluetooth is designed to have no centralized control channel, the inquiry itself can typically take between 4 to 16 seconds. Connecting to and retrieving the human-readable device name can also take a variable amount of time. Finally, the size and ordering of potential receiver devices listed to the user can be variable and may require additional user input. To the best of our knowledge there is no literature which explores techniques for modelling user interfaces with such variable

factors. However, we believe that our GOMS analysis can provide a useful metric for making relative comparisons between the different use cases under perfect conditions.

Calendar

We evaluate the number of interactions and time required to schedule a meeting time which is mutually agreeable to two individuals that are in proximity. We consider two calendars which are pre-loaded with busy schedules such that 3 PM a week from today is the first mutually available free time.

Under the existing application, each days worth of calendar entries can be longer than what is presentable on a single screenful. Users may have to scroll down to search for available free slots. The Copernicus framework automatically hides the user's normal schedule items and instead highlights mutually available free times. This reduces the amount of content the user must visually process and examine.

In terms of interaction requirements, assuming a user familiar with their schedule, we compare the interaction requirements using the existing web calendar interface versus Copernicus. Using our contrived example schedule, the user must utilize three additional key clicks, due to some days with long schedules. With Copernicus this penalty is removed because normal entries are hidden and only mutually free times are highlighted, simplifying the amount of displayed content. Using the original calendar interface requires 10 clicks, consuming 17.7 seconds (MKR[next], MKKR[pgdn,next], MKKR[pgdn,next], MKKR[pgdn,next], MKR[next], MKR[next], MKK[pgdn,select]). The Copernicus interface requires 7 clicks, consuming 16.9 seconds (MKR[next]$\times 6$, MK[select]).

Obviously this analysis does not account for conversational overhead when using the original interface. In reality, especially when arranging a meeting between three or more parties, the communication time is expected to dominate over the interaction time. Fully capturing this communication overhead with real, non-contrived calendar schedules is left for future work. However, we conjecture that even with large numbers of participating parties, the Copernicus

interface will perform close to this GOMS estimate because the communication overhead will be minimized with Copernicus filtering the schedules down to good mutual free times.

Email

In this experiment, we examine a use case where Alice checks her email in order to reference a message Bob sent to her earlier. The purpose of this experiment is to show the potential for how automated sorting and filtering can be applied to a user's own content, and that such adaptations can be effectively applied using IMAP. All experiments use the GMail web interface.

To set up this experiment, we assume that Alice receives an average of 25 non-spam emails per day. The particular correspondence from Bob she is looking for took place three days ago, with the subject line containing "Fact Check". When Alice and Bob are in close proximity, Copernicus automatically sorts and creates a "Proximity" folder sorted with correspondence with those who are currently nearby. For each of the trials, timings start with the email client open at the top of the inbox, and end when the desired message is selected.

A linear search of the user's inbox to find the three-day-old email message requires 18 clicks, totalling 15.7 seconds (M4KR, M4KR, M4KR, M4KR, MK, MK). In this example, we assume our imaginary user is familiar with the contrived inbox, and is able to quickly skip 'next' four pages to the appropriate email, where a single more careful page-down is performed before spotting the desired email.

To reduce the amount of linear searching the user must perform, we consider using sender search and keyword search. At the time of this experiment, a disadvantage the existing interface suffers from is in the placement of certain widgets, which can only be found at the bottom of the presented interface, requiring multiple interactions just to reach. Searching for the sender requires 14 clicks, totalling 9.2 seconds (M4K[search], M8K['bobuoft',enter]R, MK, MK). Searching by the expected keywords results in 16 clicks, totalling 8.3 seconds (M4K[search], M11K['fact check',enter]R, MK).

This widget penalty could be ameliorated with better widget placement. We also consider

the interaction requirements if the widget placement were improved. An improved search box location would reduce both search operations by three clicks. This gives 11 clicks and 8.6 seconds for sender search, and 13 clicks and 7.7 seconds for subject search.

Finally, we consider the Copernicus framework which automatically provides proximity context sorting. In this implementation, we sort into a folder instead of directly modifying the inbox. This requires the user to switch folders and find the email one page down, totalling 4 clicks and 5.9 seconds. In this example we consider a scenario between two people. However, Copernicus can provide prioritized sorting based on the intersection of multiple present parties, versus sender sorting which may still result in a noisy selection of matched conversation threads.

These results, summarized in Table 4.1, show that Copernicus can provide significant interaction requirements, requiring significantly fewer clicks and 25% savings in time to completion.

Photo Sharing

In this experiment we compare Copernicus to two existing usage modes for sharing photos: sharing a like to the content, and sending the content itself. To compare the sharing models, interaction measurements begin when the sender's device has the desired photo loaded and ready. For Internet-related methods, this means the photo is loaded on the sender's browser, hosted by Picasaweb. For file transfer based methods, this means the photo loaded in the sender device's native photo viewer. Measurements stop when the receiving device has received and loaded the photo. Two sizes of photographs are tested: 36 kilobytes, typical of a low-end camera-phone; and 358 kilobytes, typical of a well-compressed medium-range camera-phone.

This experiment presents an averaged timed trial performed by the experimenter. Unlike the previous applications which can exhibit wide variations due to the underlying content, the precise file size and radio transfer speeds can be benchmarked in this application. The clicks and times presented are the fastest achievable by the experimenter after significant practice.

Though another individual may conceivably achieve faster interaction speeds, we expect the delay times to be dominated by radio communication speeds rather than interaction speeds.

The first sharing model we explore involves sending a URL link of the photo of interest to the intended recipient, via email and SMS. Most popular email and SMS clients today are able to detect and hyperlink URLs in the text. The sharer addresses the link to "bobuoft" as the recipient address (the address book automatically fills in the remainder @gmail.com suffix). For SMS, the sharer enters the receiver's 10 digit cellular phone number. In the email experiment, the receiver must manually force a new mail check, instead of waiting for the usual periodic mailbox check. A limitation of this sharing model is the complexity of the permission granting process, if the sharer's photo is not at a publicly accessible URL. For this test, the photo to be shared is set as publicly accessible.

The *link* rows of Table 4.1 summarize the sender and receiver interactions as well as end-to-end time for sharing the photo. Because only a link is sent, there is little difference between file sizes of the content. However, these operations still require network traffic to connect to and transfer mail messages (both between mobiles and their mail-servers as well as Internet traffic between mail servers) before the link can be received.

For email link transfers, we also provide a GOMS estimate, which assumes instantaneous mail drop-off and pick-up, to characterize a best-case user interaction speed. The sender's interactions requires 9.4 seconds (M3K, M8K, M2K, M2KR), and the receiver's interactions require 7.3 seconds (M2KR, M2K, M2KR), for a total of 16.7 seconds in interaction time. For SMS link transfers, the GOMS estimate also assumes instantaneous SMS transfer. Using SMS, GOMS estimates 8.1 seconds for the sender (M3K, M10K, M2KR), and 2.6 seconds for the receiver (MKR), for a total of 10.7 seconds.

The second sharing model involves transferring the actual photo file to the recipient device. We compare three methods of sending content: via email attachment, Bluetooth OBEX file transfer (BT), and MMS. We assume the sharer has a copy of the photo cached on her mobile device. The *attachment* rows of Table 4.1 summarize the result of this experiment. Unlike

the link sharing model above, file sizes matter in these trials since binary file content must be transferred from one device to another. In the email attachment trials, there is a notable increase in the completion time compared to the email link sending trials, which is caused by the uploading and downloading of the image attachment. The difference in interactions required on the sender is simply due to differences in menu clicks on the device to initiate the email composition. For the Bluetooth trials, a large amount of user time is spent waiting for Bluetooth device discovery, followed by the transfer time. This network time is variable depending on interfering radio noise and speed of peer response. Discovery, which requires both a Bluetooth inquiry as well as name request, typically requires approximately 16 seconds. We note that photos larger than 250K in size could not be sent via MMS due to carrier-imposed size restrictions.

4.6 Discussion

In this section we provide a discussion of the assumptions, security, privacy, and scalability considerations with the Copernicus framework.

A significant assumption underlying the Copernicus framework is the dependence on a highly available high-speed infrastructure wireless system, such as wide-spread high-speed cellular. As consumer demand for wireless services and smartphones continues to grow, the availability of high-speed infrastructure services is expected to expand. We expect that availability and use of these infrastructure services will continue to dominate the vast majority of use cases for most users.

However, as described by Haggle in Chapter 3, there may be occasions when infrastructure is unavailable or undesirable, such as when the user is roaming. We believe there is a complementary role between Copernicus and Haggle. While much content is dominantly hosted and serviced online, some web applications also support local plugins which enable off-line access to locally cached content and functionality, for example Gears [33] for GMail. We believe

Calendar (GOMS)	clicks	GOMS estimate (s)
regular calendar	10	17.7
Copernicus	7	16.9

Email (GOMS)	clicks	GOMS estimate (s)
GMail (linear scan)	18	15.7
GMail (sender search)	14	9.2
GMail (subject search)	16	8.3
Copernicus	4	5.9

Photo	interactions			GOMS
	sender	receiver	timed trial (s)	estimate (s)
link (email)	15	6	66.2	16.7
link (SMS)	15	1	48.8	10.7
attachment (large,email)	10	6	96.8	13.0
attachment (small,email)	10	6	79.3	13.0
attachment (large,BT)	3	3	58.4	7.6
attachment (small,BT)	3	3	37.1	7.6
attachment (large,MMS) †	14	3	-	-
attachment (small,MMS)	14	3	48.8	12.2
Copernicus	3	4	15.0	12.2

Table 4.1: *Results for Calendar, Email, and Photo. Timed trials are average of best three attempts. GOMS estimates interaction time only – does not take into account network lag. Large photo is 358k, small is 36k.* †*failed due to carrier's 250k limit.*

these plugins and more can be built atop Haggle and extend the functionality of web-based applications to opportunistic or disconnected mobile environments.

4.6.1 Security and Privacy

Copernicus simplifies sharing by providing a list of peers with human-identifiable attributes (such as real name and profile photo), and mapping their online identities, accounts, and web applications. This new usage model raises security and privacy concerns as well as the risk of receiving unwanted content such as spam.

The current Copernicus system relies on periodic Bluetooth inquiries for detecting user proximity, which raises spoofing and tracking risks. First, while most consumer products with Bluetooth radios are hard-wired with a Bluetooth MAC address, a malicious party may attempt to build custom hardware that transmits fake Bluetooth address responses, allowing impersonation. Such an attack would have limited effectiveness in Copernicus, since it does not allow the attacker to gain access to the content of the impersonated identity. For example, suppose Alice wants to share with Bob, but Eve impersonates Bob's Bluetooth address. Alice would be tricked into thinking that Bob is nearby. However, since Copernicus sets access permissions on the originating web application, Eve can only gain access to the content by compromising Bob or Alice's accounts on the web applications.

Second, Copernicus does not expose users to user tracking risks above the current status quo for identifying mobile devices. Existing methods have explored techniques for overcoming this social presence problem [20], which we leave for future work. In addition, users can disable Bluetooth by default, and enable it explicitly when they want to share. To mitigate the risk of Bluetooth inquiry response tracking, our prototype provides a method for direct peer specification via a cellular phone number. This enables the recipient to remain completely invisible, while being able to receive sharing requests. We described the use of SMS, as well as other possible techniques for establishing proximity without the use of active Bluetooth inquiries, in the architecture section (Section 4.3.1).

Malicious users may attempt to spam other users in their proximity by sharing unwanted data with them. Copernicus enables users to restrict their visibility to individuals with whom they have previous sharing relationships or are marked as acquaintances on a social networking application. In addition, Copernicus provides users with filtering and rejection options analogous to email spam filtering.

The use of delegated credentials is common amongst web applications, and can take many forms, ranging from "home-brew" delegated credentials to delegation through single-sign-on services. In all cases, Copernicus does not store passwords. Access to the delegation token does not grant an attacker access to the user's web application data, since web application requests must be signed from a specific IP address associated with the token. An attacker must compromise the Copernicus portal, or steal its private key and spoof its IP address to gain access to the user's data. Thus, Copernicus's security profile is similar to other web applications using credential delegation, and Copernicus does not expose additional vulnerabilities. In particular, users who wish to unregister from Copernicus can revoke the delegated credential from their web applications.

As a centralized service, there is a security risk of having Copernicus maintain delegated credentials for users across many of their different web applications. If the Copernicus server were compromised, an attacker may potentially have the ability to access and modify the contents and web applications of other Copernicus users. It should be noted that delegated credentials are a known feature of many Web2.0 applications, and Copernicus's use of delegated credentials does not pose a greater threat than the status quo. Many delegated credentials are tied to specific public IP addresses. This ensures that delegated credentials cannot be granted once and copied to many different services. Therefore, an attacker must not only steal delegated credentials from the Copernicus server, but also execute requests from the server in order to utilize the stolen tokens. If such a compromise were to occur, Copernicus and its users can revoke all delegated credentials granted to Copernicus from the originating web applications.

4.6.2 Scalability

The current prototype of the Copernicus framework is focused on the mapping of user objects and enabling the adaptation of web applications to be socially aware. As such, there are many avenues of improvement for improving the scalability of the Copernicus framework.

One limitation of the current Copernicus implementation is the centralization of content sorting and filtering. Currently, Copernicus must pull content and meta-data from web applications to sort and filter, and then push back adaptation results to the web applications. A possible solution is to push these operations to the originating web applications. Many web applications include support for advanced queries and batch operations to reduce load and latency. Copernicus can utilize these interfaces to also reduce the amount of content which must be pulled and examined. As these web application APIs mature and the number of third-party applications grow, we envision web applications to continue expanding support for sophisticated query interfaces.

As users move about their environment, the fingerprinting information sent by their mobile devices can fluctuate frequently, especially in areas with large numbers of devices and people. Since Copernicus only considers pairwise connections and peers (no transitive closures), distributing and balancing this processing load is simply a matter of implementation. Furthermore, database updates to specific users can be balanced using database sharding techniques across multiple sub-tables across multiple machines.

We envision a wide range of user preferences, ranging from users who utilize very few dynamic adaptation filters under strict preferences, to users who utilize many dynamic filters and lax privacy requirements. For this later group of users, Copernicus must make many adaptation actions, many of which might not be seen by the user. This results in much "wasted work". To mitigate this, we can take advantage of the Copernicus mobile client. Since the mobile client runs on the user's local device, it can also detect whether or not the user may be actively interacting with the device, and send this information to the Copernicus server along with its periodic fingerprints. The Copernicus service can use this as an indicator to more aggressively

apply adaptations, or delay taking action if the user is inactive.

Chapter 5

Timbremap

Mapping applications on mobile devices have gained widespread popularity as a means for enhancing user mobility and ability to explore new locations and venues. These applications can provide two features, *navigation* to a specific location and *exploration* of an area to obtain a sense of the general layout of streets and locations. For visually impaired users, current mapping applications are mostly text-to-speech (TTS) driven, whereby lists of directions or items are enumerated to the user. These applications are primarily for real-time navigation, providing corner-by-corner walking directions [50, 88].

Unfortunately, TTS is poorly matched for map exploration, as speech-based techniques are limited in their ability to succinctly describe complex shapes and positions of features. This shortcoming is apparent even in human conversation when attempting to describe a complex layout. In formative interviews that we conducted with visually impaired participants, several have indicated that a common technique for addressing this limitation is to "draw" with their finger on the other person's hand, illustrating a shape or position through tactile interaction and feedback.

A second significant limitation of existing mapping applications is that they mainly support public outdoor environments. There is no pervasively available localization system for indoor environments, such as office spaces and large malls, to aid in navigation or destination

searching. Indoor environments are particularly difficult to explore or navigate for many of our visually impaired participants. While some buildings have maps and diagrams to show the layout of the indoor space, these diagrams are rarely accessible to the visually impaired. Furthermore, even at locations with braille or tactile maps, it can be difficult for individuals to determine the correct orientation and direction of travel [11]. More commonly, visually impaired users will call accessibility hotlines or ask help from strangers to describe, in detail, locations and features.

This chapter introduces Timbremap, a novel interface for visually impaired map and floor-plan exploration. Timbremap utilizes a touch surface as an input and output mechanism to convey complex geometrical information. User's provide tactile input by using a finger to trace shapes on a touch surface, and Timbremap provides output feedback in the form of sonification (non-speech audio to convey or perceptualize data) hints to guide the user's finger along shapes, enabling the user to develop a cognitive understanding of the paths and features within a layout.

We chose to implement Timbremap on a touch-based off-the-shelf mobile phone because it enables users to explore maps and floor-plans both at home or while on location (e.g., when entering an unfamiliar hotel, office space or shopping mall). Mobile exploration helps users deal with unexpected situations, and reduces the cognitive load associated with memorizing floor-plans.

The design of Timbremap presents two challenges. First, the interface must be effective in guiding the user's finger on a smooth touch-screen surface. Second, the interface must function within the confined surface area of a mobile phone.

This chapter presents two major contributions. First, we present a novel sonification technique which conveys fine-grained path geometry and positioning on a touch-screen mobile device. We present two sonification modes – one actively guides the user's touch, based on the underlying layout information, while the other uses a graph-coloring technique to passively convey information at the touch point. Our user-study evaluation shows Timbremap is effective in conveying complex geometry information to visually impaired users. Our participants

achieve an average accuracy of 81% in shape identification, spending just over 41 seconds per shape. Second, we show that the Timbremap application is effective in conveying complex indoor floor-plans. A user is able to build a mental map of a complex indoor floor-plan, describe the points of interest, and provide directions for navigating around the indoor space.

These results are highly encouraging, thereby motivating future development of mobile and ubiquitous platforms for providing accessible exploration of indoor and outdoor map and location information. Furthermore, there is significant potential in expanding Timbremap's interface to medium and large format touch devices such as tablets and tabletops.

5.1 Related works

Existing products such as tactile and braille maps have long been available to convey geometric and spatial information to visually impaired users. Unfortunately, these physical objects are expensive or time consuming to create, difficult to carry, and often provide limited amounts of information or outdated information. Higher-end TTS GPS devices enable users to virtually navigate around street maps without physically travelling the route. Some visually impaired users are able to use this feature to gain some acquaintance with some locations and build a rough cognitive map of relative street locations. However, this interface suffers from several significant drawbacks which Timbremap addresses. First, the GPS virtual navigation interface is poor at conveying complex street and intersection orientations, and requires significant user attention to deduce relative positioning of points of interest. Second, because GPS systems do not function indoors, these systems are unable to provide exploration or guidance of indoor spaces.

A closely related work which attempts to utilize finger spatial positioning to convey geometric information is presented by Jacobson [37]. Jacobson's approach utilized a touch-pad to convey relative positioning of points of interest on a map using sonification.

The most significant difference between Jacobson's work and Timbremap is in the motivat-

ing focus of the interface design. Jacobson's interface focused on the presentation of points of interest on the touch surface. Paths are themselves points of interest (in terms of audio feedback), and users must deduce that paths exist and determine their characteristics. Timbremap's primary focus is to present a sonification interface conveying precise path geometry and positioning information, with points of interesting being secondary objects attached to the path map. Comparing Jacobson's interface to Timbremap's active sonification mode, Jacobson's interface provides sonification primarily for points of interest, where paths are themselves points of interest. Contrastly, Timbremap's interface provides sonification specifically designed to precisely convey path geometry and positioning.

5.2 Timbremap Interface

The primary objective of the Timbremap interface is to provide a clear method of conveying non-trivial geometry, spatial positioning, and feature information by guiding the user's touch exploration of the device's touch surface. To this end, we have developed an interface which can handle a wide range of complex geometries found in most roads and indoor spaces.

Through an iterative design process we present and evaluate two sonification modes for the Timbremap interface. First we present a pro-active sonification mode which attempts to guide the user's touch. Then we present a passive sonification mode which conveys spatially connected features. Finally we present an indoor floor-plan exploration application using the Timbremap interface, and its implementation on an iPhone device.

5.2.1 Shape hinting mode

The Timbremap *shape hinting* sonification mode provides pro-active feedback to help guide the user's touch within the path segments of a layout. Should the user's touch point wander away from the path segment, hinting sounds are played to guide the user toward the nearest path segment, relative to the touch point.

The design process started with an initial interface which provided audio feedback when the user's finger was over a path portion of the shape. The audio feedback faded in stereo as the user's finger wandered farther from the path. Early user feedback on this interface suggested that the audio feedback did not provide sufficient information with regards to what corrective action the user should have taken. Stereo positioning of the path sound, relative to the deviation of the user's finger to the path, was found to be too subtle.

To address this issue, we implemented an audio hinting system which indicated to the user what corrective action to take in order to return to the nearest path point. The first iteration of the hinting system used mid-tone beeps on left/right stereo channels, and up/down were indicated using high and low tones, respectively. Feedback to this found the high and low pitched tones for up down to be too abstract. To address this, we replaced the high/low tones with high and low-pitched spearcons [25, 85] (compressed speech audio icons). Furthermore, the initial iteration supported diagonal hints – for example playing a high-pitched tone in the left ear to indicate upper left. Many users found the diagonal hints to be too complex. We subsequently restricted this hinting system to the four compass directions during the experiment.

In designing the sonification for the shapes and paths, we found a trade-off had to be made between the width of the path segments and the sensitivity of the interface. Making the path segments very thin enabled users to detect subtle changes such as minor curves. However, the thin segments were also much less forgiving, resulting in "noisy" feedback, constantly hinting the user towards the thin line. Conversely, a thicker line was found to be much more relaxing and forgiving to use, but made subtle curves difficult to detect. To address this, we use a slightly modified sonification for curved segments. The curve sound does not provide information as to the direction of the curve or how curvy the segment is. It only indicated a curve segment, and the user must interact with the device to determine its characteristics. This limitation on curve sonification was designed for two reasons. First, it reduces the amount and complexity of audio feedback the user must process and listen for. Second, interviews with participants revealed that how a curve should be sonified is relative to the user's perception. For example,

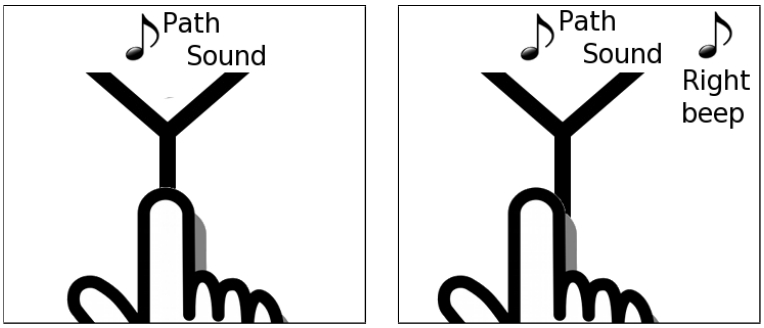


Figure 5.1: *Shape hinting example. As finger shifts to the left, sonification beeps on the right to indicate path is to the right*

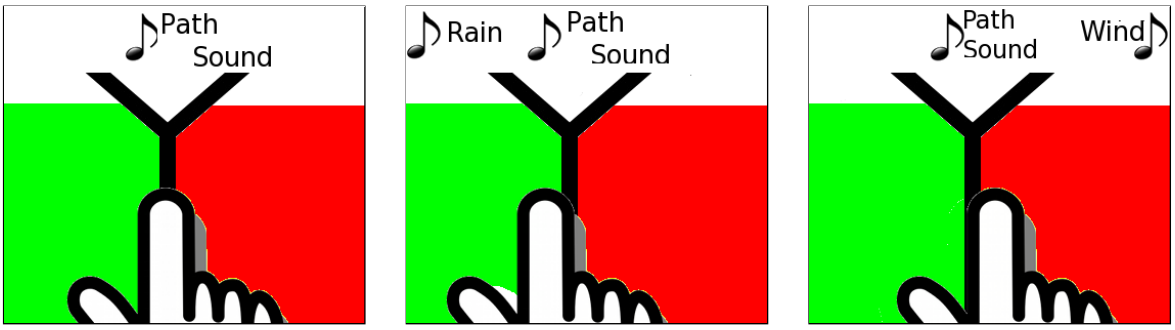
some participants perceived a curve from the center to the right of the device's screen as a curve to the right when traversing from bottom up; but perceives the same curve as a curve to the left when traversing from top down.

Finally, we added sonification markers for intersections in order to indicate more than one possible traversal path. We defined an intersection as any point where three or more segments connect. Thus, a right-angle turn is not considered an intersection, but a T-junction is. The intersection sound not only helped notify the user of an intersection, but also serves as an easy-to-find marker as a reference point.

Figure 5.1 illustrates an example of this sonification mode. First, the user hears a repeating audio pattern representing the path segment. As the user's touch shifts to the left, a repeating beep in the right ear is played to indicate that the nearest segment is to the right of the finger. This interface uses mid-tone beeps on stereo channels to indicate whether path is to the left or right of the touch point. For up and down adjustments, this interface uses high-pitched "up" and low-pitched "down" spearcons, respectively. With practice, users should be able to respond very quickly to the audio hints and follow path segments accurately.

5.2.2 Color hinting

Observation of user strategies in the iterative design process of the shape hinting mode led us to develop an alternative sonification, which we called the *color hinting* sonification mode. We



(a) finger on path

(b) finger shifted to left; sound for left side coloring is played

(c) finger shifted to right; sound for right side coloring is played

Figure 5.2: *Color mode*

found that users often swept their finger across the screen surface to pick up global characteristics such as number of segments around edges, gaps between segments, and existence of any intersections. The coloring mode had three design goals: (1) remove explicit directional sonification while providing left/right/up/down information; (2) enable users to deduce contiguous regions of empty space; and (3) improve clarity of feature detection when using a swiping strategy.

As with shape hinting, the color hinting mode uses the same sonification strategy for denoting straight and curved paths and intersections. The blank spaces surrounding the paths are flood-filled with colors, assigned using a 4-coloring heuristic. These four colors are mapped to a set of four ambient sounds: light raindrops, wind, chirping crickets, and gentle wind chimes. This sonification system enables users to detect whether they are straying from a path segment, and in what direction, by listening for and comparing changes in sound. For any given segment, the ambient sound for either side of the path will be different. An example of the audio feedback is illustrated in Figure 5.2. On the path, the user hears a repeating audio pattern. If the user's finger shifts to the left, the path sound fades out and the sound of rain fades in from the left. If the user's finger shifts to the right, the sound of wind fades in from the right. By swiping over the touch surface, users can assess the number of segments as well as the connectivity of empty spaces.

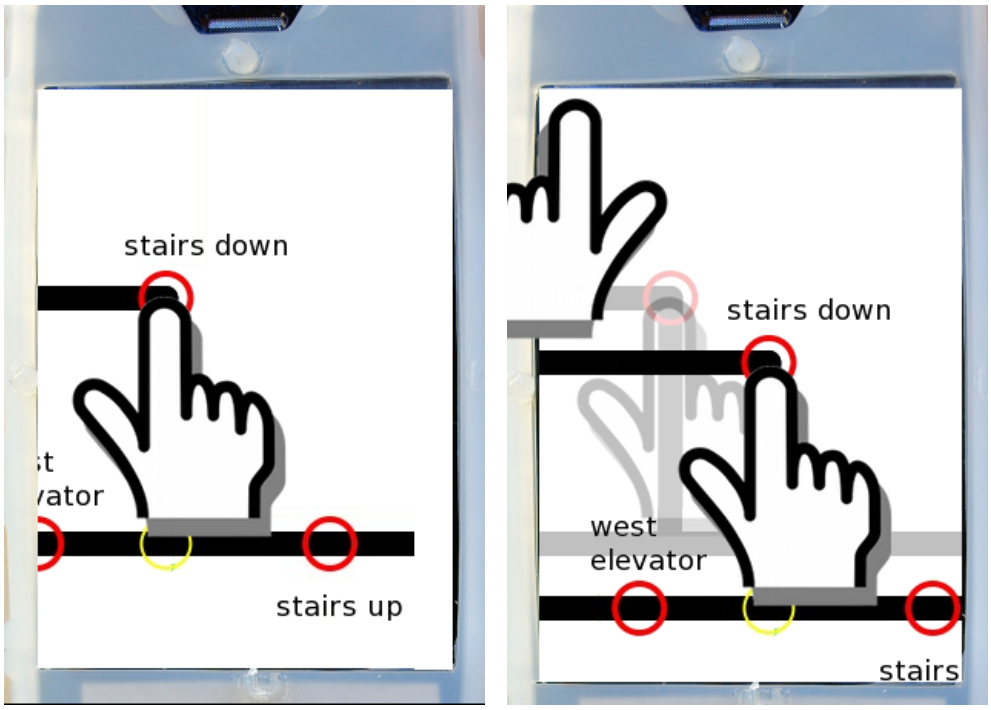
5.2.3 Indoor exploration application

Using the sonification interface described above, we implemented a basic indoor floor-plan exploration application. In addition to providing sonification for paths, spaces, and intersections, this application added a sonification for marking points of interest (POI). Identifying POI markers and map panning is supported via intuitive multi-touch interactions [40]. Specifically, to identify a POI marker, the user holds one finger on the POI marker, and double-taps anywhere on the screen with a secondary finger. This commands the system to read the marker using a TTS voice.

To pan the map, the user first positions their primary finger on any spot on the map. With a secondary finger, the user holds any of the four corners of the screen. By dragging the primary finger, the map pans by moving with the primary finger. Figure 5.3 illustrates an example map panning operation. Assuming the user wants to pan the "stairs down" point of the map. With a finger over the "stairs down" marker, the user places a second finger on one of the corners, in this case the top left corner. Then as the user pans the primary finger, the map pans with the finger. By holding and releasing the corner touch, the user is able to quickly "walk" and pan across multiple screens without losing their place.

5.2.4 Device

We chose to use the iPhone as the mobile device platform for two reasons. First, at the time, the iPhone was the only available mobile smartphone which featured a capacitive touch-screen with support for multiple simultaneous touches on the screen. We chose a capacitive screen instead of a resistive screen because we felt it would be a more ergonomic touch mode when primarily using fingers. The actual device used was an iPhone 3G, running the 3.0 firmware. The application is implemented as a native iPhone application, written in Objective-C. Storage for reading shapes and logging data are done using the SQLite3 library provided as part of the iPhone application API.



(a) positioning finger (b) map pans with finger

Figure 5.3: *Panning operation. By holding any of the corners, the user can drag the map.*

For aesthetics, the iPhone is designed with many smooth surfaces. While visually appealing, this results in a device which can be slippery to hold and lacking in tactile feedback for certain physical features. To mitigate this, we wrapped the iPhone in an after-market silicon "glove". The silicon border enabled users to feel the boundaries of the touch screen area. This was important since the top and bottom edges of the touch screen area are smooth with respect to the earpiece and home button areas. The non-slip silicon glove also enables users to use both hands to interact with the iPhone on a table-top without the device sliding around. Figure 5.4 shows the iPhone device in its silicon glove.

In this prototype implementation, we assumed users will use stereo headphones in order to take advantage of the stereo sonification techniques Timbremap employees. Since Timbremap is focused on exploration rather than navigation, we do not expect this to negatively impact visually impaired users.

(a) iPhone in silicon glove

(b) tile dimensions match device's screen dimensions

Figure 5.4: *iPhone and sample tile.*

5.3 Experimental Setup

The objective of the experiment and user study is to first determine whether the use of sound hints combined with touch interaction in the Timbremap interface is effective in conveying geometry information; second, determine whether there is a significant difference in the effectiveness of the *shape* and *color* hinting modes; and third, whether these basic skills translate to an ability to explore an indoor floor-plan.

5.3.1 Shape identification and discrimination

This first user study is designed to quantitatively determine if users are able to identify and distinguish between different shapes using the two sonification modes described in the previous

section. We begin with a description of the experimental procedure in this user study, followed by a description of the shapes and selection tiles in the following subsections. Participants will use the prototype interface implemented on the iPhone device, along with a pair of stereo headphones for the sonification.

This study is divided into three parts: a tutorial, practice, and test. In the tutorial segment, the participant is given time to be acquainted with the interface using sample shapes which do not appear in the test portion. In the practice segment, participants were given eight practice shapes. In the first four practice shapes, the participant is first presented with a physical etched tile of the shape (described later in Section 5.3.1), and then asked to look for the shape on the device's interface. In the later four practice shapes, the participant is presented with the device's interface first, and then asked to find the shape amongst a set of possible tile choices. This tutorial portion is not timed, and participants are given immediate feedback on the correctness of their selection. At the end of the tutorial session, participants are given the option to retry any of the tutorial shapes until they are comfortable with the workings of the system.

The test portion of each interface is divided into two blocks. Each block presents 24 shapes, including repeated presentations. These 24 presentations are chosen from coreshapes and distractors. We present 18 coreshapes (three repeats of six coreshapes), and 6 presentations of distractors. The goal of this arrangement is to determine if participants are able to detect general properties of the shapes, as well as differences between similar shapes. By using two test blocks, we are able to test if there is a noticeable learning effect between the first to the second block.

For each shape, the participant is first presented with the device interface, and given at most 60 seconds to interact with the interface, as illustrated in Figure 5.5. Participants are encouraged to finish before the expiration of the timer if they are confident they know what shape is being presented. The participant is then given 30 seconds to select from amongst three engraved tiles. These three choices are placed in a wooden holder, as shown in Figure 5.7b. One tile choice is an exact match to the displayed shape, with the two others being similar

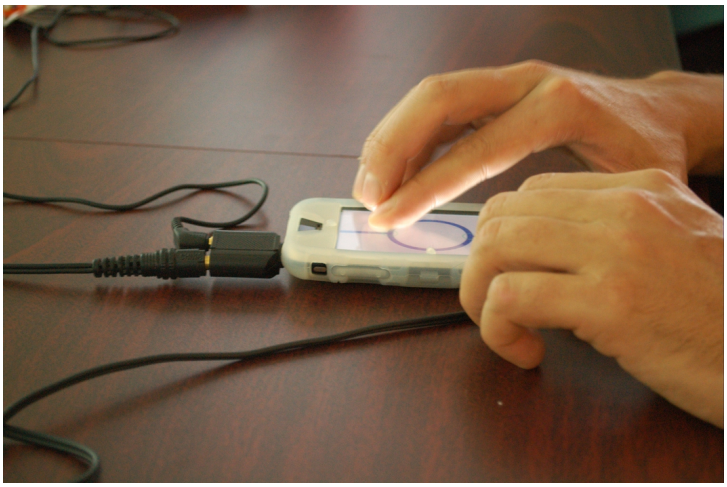
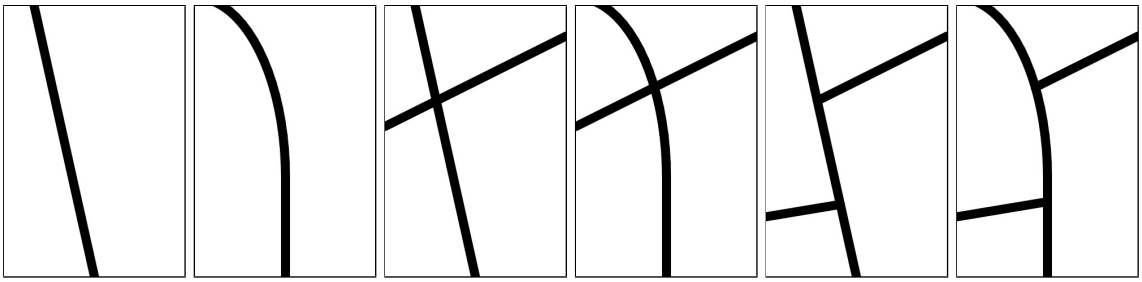


Figure 5.5: *Participant using the Timbremap device during a study. The shape is shown on the device screen for experimenter reference. This particular participant is completely blind and could not see the shape. A Y-splitter was used in the audio jack to enable the experimenters to also hear the sonification given to the participant.*

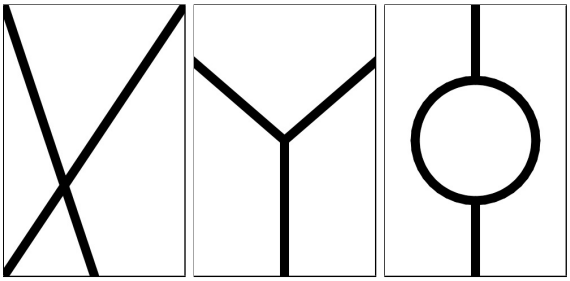
but wrong choices. We use a multiple choice answer selection format to remove ambiguity that might otherwise result if participants were asked to describe or draw the displayed shape. After each block of 24 presentations, the participants responded to a brief questionnaire and are given a short break before repeating the three-part process again with the other hinting sonification mode.

Shapes

One goal of the shape design process was to select a variety of shapes which conveyed different road or indoor path features, in incremental degrees of complexity or feature. Thus, we were not interested in overly arbitrary or abstract geometries, but rather to determine participants' ability to distinguish between realistic shapes. Figure 5.6 illustrates the final set of shapes used in the test portion of the experiment. These shapes are divided into two main categories: *coreshapes* and *distractors*. The coreshapes, as seen in Figure 5.6a are similar in concept, each designed with a subtle variation and increasing in complexity. The distractor shapes



(a) coreshapes numbered 1 to 6

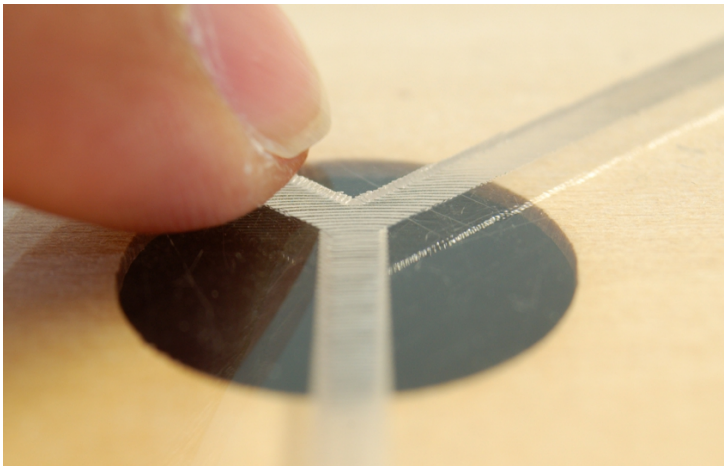


(b) distractors numbered 1 to 3

Figure 5.6: *Shapes used in experiment. Coreshapes are incremental in complexity. Distractors used to prevent educated guessing*

(Figure 5.6b) are designed to be a departure from the coreshapes, to provide variations in the features that participants must check for, yet bear relation to shapes a user may commonly encounter on maps and paths. In formative trials of the user study, participants quickly learned the common characteristics amongst the coreshapes and "over-fit" their interactions to look for very specific features such as a specific intersection or line segment. These chosen distractors were found to be effective in preventing this overfitting behaviour.

To provide participants with a familiar and consistent reference point for exploring the shapes, we decided that all shapes should have a common absolute point in the tile area. This reference point would provide a known location to start at or return to should the participant become lost or confused. We decided to place this common reference point at the middle of the bottom edge of the rectangular area.



(a) groove dimensions relative to index finger

(b) tile holder with three sample tiles

Figure 5.7: *Close-up of tile groove and multiple-choice wooden tile holder.*

Tiles

To provide a real-life tactile representation of these shapes for the visually impaired participants, we chose to engrave them onto acrylic tiles. These tiles are cut to the same dimensions as the iPhone device used as the mobile platform. The widths of the engravings match the widths as presented on the device's interface. In designing the engraved tiles, we decided to engrave the path and leave the "blank" areas smooth, as opposed to raising the path above the surface. This was decided for two reasons. First, the grooved etch along the path provides a rougher, stronger tactile sensation. This positive feedback reinforced the presence of the shape,

while the smoothness of the blank areas conveyed a sense of "nothing". Furthermore, the etching produced a groove which helped guide the participants' finger along the path. A close-up picture of an index finger along a tile's groove is illustrated in Figure 5.7a.

For participants to feel and compare multiple tiles, we used a wooden frame which can hold three interchangeable tiles. This frame, illustrated in Figure 5.7b with three sample tiles, provides a convenient method for participants to find and compare three given choices.

5.3.2 Indoor exploration application

In a second user study, we qualitatively examine whether the shape identification and discrimination skills utilized in the first user study translates toward an ability to pan a large indoor floor-plan and building a cognitive map of the space and its features.

Two maps were supplied, one more complex than the other. Each map is larger than can be displayed in one screenful, and contains several POI markers. An example indoor floor-plan is shown in Figure 5.8. This study imposed no time limits; participants are given as much time as they like to explore the map until they are comfortable and confident they understand the layout. We then ask several POI identification and walking direction questions, such as navigating from the attorney's office to the break room.

5.3.3 Participant pool

Our first user study consisted of 6 individual participants; 2 female and 4 male, as shown in Table 5.1. Three participants reported their age in the 30's; one participant in their 40's; one participant in the 50's; and one participant in their 60's. One participant is congenitally blind. All have been completely or nearly blind for at least 10 years, save for basic light sensitivity. Four participants relied primarily on a guide-dog for navigation, and two relied primarily on a cane. The second user study used one returning participant from the first study.

All of the participants had at least some braille reading ability. On a five-point scale (with

participant	sex	age	braille score	shape	color
1	M	50s	2	Y	Y
2	M	40s	1	Y	N
3	F	60s	5	N	Y
4	M	30s	1	Y	Y
5	F	30s	4	Y	Y
6	M	30s	1	Y	Y

Table 5.1: User study participants

five being excellent), two participants scored themselves as 4 and 5, with the remainder scoring themselves as either 1 or 2. All participants reported having extensive experience or familiarity with mobile phones and computers; only two participants have technical occupations relating to computers and software. None of the participants were affiliated with the university or research lab.

Participants were recruited via a mailing list of local and diverse visually impaired users interested in research participation. Participants were scheduled for a 3-hour block of time, and granted a $60 honorarium for their time.

5.4 Results

The following two sections describe the results of our two user studies.

5.4.1 Shape identification and discrimination

We carried out the experiment using 5 trials of the color interface and 5 trials of the shape interface. Four out of the six participants were able to try both the shape and color hinting sonification modes. We counter-balanced the presentation of the two sonification modes to

eliminate learning bias.

We first examine the correctness results of participants' answers, detailed in Table 5.2. A summary of the mean accuracy with 95% confidence intervals between the distractors, core shapes, and all shapes combined, is shown in Figure 5.9. Across both modes, participants were very successful at determining the shape within the time constraint, averaging over 80% accuracy.

We examined whether there is an accuracy difference between the two modes, and whether there is an improvement due to learning from block 1 to block 2. We found no statistically significant difference in the means and deviations between the sound and color hinting sonification modes ($T = 0.097 < T_{crit}^{\alpha=0.05} = 2.440$). This suggests that both modes are effective. Based on participant feedback, the choice of which to use should be left to user preference. Future work may examine whether users have specific preferences for sonification modes under different environmental conditions (e.g. noisy rooms or outdoors).

Our calculations also do not indicate statistically significant improvements in accuracy between blocks 1 and 2 for the shape and color hinting sonification modes ($T = 0.458 < T_{crit} = 2.44$). We suspect two contributing factors in this result. First, we believe that more time is needed for users to improve their skill with the interface. Some users commented that they believed more time and practice would improve their ability to use the interface. Second, from our observations, we speculate that some participants were increasingly mentally fatigued by the second block. Some of the participants were more senior in age, and some performed the experiment after work hours. These factors could impact the participants' concentration and accuracy. Future experiments could split the study over two or more days to reduce the impact of fatigue and test for possible improvements due to learning.

Next, we examined the amount of time that participants needed to determine a correct answer. On average, participants were able to determine the correct shape after approximately 41 seconds. The full findings are detailed in Table 5.3. A summary of the distribution of mean completion times is summarized in Figure 5.10. This is a positive result showing par-

	block1	block2	combined
distractor 1	90.0	83.3	85.0
distractor 2	90.0	66.7	85.0
distractor 3	70.0	66.7	65.0
coreshape 1	100.0	100.0	100.0
coreshape 2	93.3	100.0	96.7
coreshape 3	86.7	88.9	86.7
coreshape 4	70.0	77.8	70.0
coreshape 5	66.7	77.8	63.3
coreshape 6	70.0	88.9	70.0

(a) shape mode

	block1	block2	combined
distractor 1	83.3	66.7	75.0
distractor 2	83.3	83.3	83.3
distractor 3	50.0	83.3	66.7
coreshape 1	88.9	100.0	94.4
coreshape 2	100.0	88.9	94.4
coreshape 3	100.0	77.8	88.9
coreshape 4	66.7	100.0	83.3
coreshape 5	88.9	55.6	72.2
coreshape 6	77.8	55.6	65.6

(b) color mode

Table 5.2: *Percentage answered correctly*

ticipants are able to determine complex shapes, without a priori knowledge of what to expect, in relatively short periods of time. Our analysis found no statistically significant difference

	block1	block2	combined
distractor 1	45.8	40.5	45.3
distractor 2	46.4	38.0	43.8
distractor 3	46.0	51.5	50.3
coreshape 1	36.8	20.3	28.2
coreshape 2	37.1	34.8	36.6
coreshape 3	43.6	33.7	43.8
coreshape 4	49.2	43.8	46.2
coreshape 5	45.2	38.5	45.0
coreshape 6	45.5	47.5	48.3

(a) shape mode

	block1	block2	combined
distractor 1	51.5	48.5	50.8
distractor 2	50.0	49.7	51.5
distractor 3	46.0	40.8	42.8
coreshape 1	38.2	35.0	37.8
coreshape 2	45.3	45.0	46.5
coreshape 3	46.0	46.0	47.8
coreshape 4	54.2	50.0	53.5
coreshape 5	53.5	52.5	50.0
coreshape 6	54.5	55.5	55.7

(b) color mode

Table 5.3: *Median completion time distribution (on-time and correct answers).*

between the time-to-completions for the two hinting modes ($T = 0.761 < T_{crit}^{\alpha=0.05} = 2.440$), which corresponds with earlier results.

Finally, as described in Section 5.3.1, we designed distractor shapes to prevent "overfitting" behaviour while not presenting geometries significantly more challenging than the core set of shapes. Our results show no statistically significant difference between correctness results for the core shapes versus distractors on the shape hinting ($T = 0.187 < T_{crit}^{\alpha=0.05} = 2.170$), and color hinting ($T = 0.435 < T_{crit}^{\alpha=0.05} = 2.440$) sonification modes. The median completion times for the distractors versus core shapes were also not significantly different for the shape ($T = 0.586 < T_{crit}^{\alpha=0.05} = 2.200$) and color ($T = 1.693 < T_{crit}^{\alpha=0.05} = 2.440$) hinting modes.

5.4.2 Indoor exploration application

In the first map, the participant interacted with the device for approximately 14 minutes before indicating he was confident about the layout of the space. The participant found all but one of the POI markers (the washroom), which he was able to locate after an additional 1:20 (one minute and twenty seconds) of interaction.

In the second, more complex map, the participant interacted with the device for approximately 10 minutes before indicating he was confident about the layout of the space. We attribute this increase in speed despite the more complex map to improved familiarity with the interface. The participant found all except the west-side break room and conference room. The participant took an additional 1:10 to find the conference room, and an additional 3:10 to find the break room. When asked for directions from the attorney's office to the break room, the participant correctly provided turn-by-turn directions and named the POI markers for the route via the "stairs down" path. The participant indicated that he suspected there could be a second path via passing the conference room, but was not confident.

Qualitatively, this experiment suggests that users are able to piece together multiple screenfuls of content to perceive a much larger map. The participant noted that the larger map was intuitive to piece together because it utilized a similar skill to what is currently available in GPS navigation systems.

In post-experiment questioning, the participant was positive in his reaction to the applica-

tion, stating that "if I were sitting in an airplane and had the time to look at the hotel lobby before I got there, I'd definitely do that." The participant went on to detail "this one particular hotel in Anaheim that I wish I could have had [this]. There was [sic] four different banks of elevators, and they were all in the middle of the lobby and the lobby was *huge*. It was all on angles; even sighted people had trouble."

With regards to the interface, the participant indicated that the largest difficulty he had was with the panning operation. Interestingly, the participant preferred to have the primary sensing finger stationary, and to use a secondary finger to slide the map under the primary finger. We suspect this preference may be because it is similar to the panning mode utilized by GPS navigation devices available today, where the map pans under a centrally anchored cursor.

5.5 Discussion

When comparing the two hinting modes, most participants remarked that the color hinting mode was much more aesthetically pleasing. Unfortunately, we believe this hurt the performance results for this interface due to the setting of our experiment. Our experimental test setting was "exam-like" in many ways, including timed progression and multiple-choice questions. Several users remarked that as they mentally fatigued, they often took time to enjoy the pleasing background sound effects in the coloring interface, despite the delay being detrimental to the task at hand.

Some participants noted that the coloring sounds were too different. We then tried changing them to be more subtly close (wind, beach waves, rain drops, hum of an air-conditioner). Other participants who tried the more similar coloring sounds remarked they would like the sounds to be more distinctly different. We suspect that the best sound for the ambient coloring will depend on user context as well as individual user preferences.

Despite the shape hinting mode being less aesthetically pleasing, many users preferred it over color hinting because the sonification provided explicit feedback for what to do. Some

participants remarked that this explicit hinting helped them stay focused. Removal of the diagonal hints initially used made the interface simpler and easier to learn. However, we observed that this directional restriction resulted in confusion in certain situations. For example, if the closest direction to a curve is diagonally to the lower left, the system may hint "down". Due to the downward curve, this resulted in a large travel distance to find the path again. We suspect this produced an inconsistency in the participant-perceived granularity in feedback sensitivity, resulting in confusion.

We do not have sufficient data to conclude whether or not there is a correlation between braille reading ability and performance. However, based on anecdotal observation of the participants, there appears to be no correlation between braille reading ability and performance on the interface. We speculate that spatial positioning skills are likely to be a more important factor in determining whether a user performs well with the Timbremap interface.

There is little doubt that regardless of natural ability, the Timbremap interface is one that must be learned and practiced. Despite not seeing an appreciable improvement in interaction speed between the first and second blocks of each hinting mode, many participants speculated that additional training would likely improve their confidence, accuracy, and speed with the interface. We leave this extended study of improvement over training and time to future work.

5.5.1 Limitations

In this section we discuss various limitations of the Timbremap interface and its experimental evaluation.

Distractors and core shapes

As presented in previous sections, the distractor shapes were designed to discourage educated guessing of the limited number of test shapes. While our analysis did not reveal significant difference between core shape and distractor results, the confidence intervals for distractor shapes appear to display wider variability than the core shapes, as seen in Figure 5.9b. There is insuf-

ficient data in our study to determine why the distractors appear to exhibit greater variability, or whether it is a specific minority of the distractors that participants have greater difficulty with. This is a limitation of the current experimental design. A greater comprehensive study of the relative difficulty of various shapes and shape features, and how visually impaired participants perceive and evaluate different shapes is left for future work.

Practical deployment

One assumption made underlying Timbremap's motivation is that the physical world can be sufficiently tagged with wireless information to provide localized context for the interface. For example, we assume that major entrances, elevators, walkways, and common areas will be tagged with short-range identifiers, allowing Timbremap to accurately localize the user, especially in areas where GPS localization is inadequate.

Such a deployed system must consider the roles of the environmental tags and tags on mobile devices. Active environmental tags with passive mobile device tags can be a power effective solution, since mobile devices would not be burdened by power-hungry periodic scans. Alternatively, active mobile tags would be able to detect ad hoc environmental tags, such as tagged hazard cones marking a detour.

In either approach (or combination of the two), privacy concerns will have to be addressed. Such a deployment can potentially leave the user open to tracking, either through the tags themselves or through the device's query pattern. For example, suppose the system relies on a third-party service, such as Google or Bing, to map tag identifiers to map locations and layout information. By tracking the tags queried, these third-party services may be able to track the user's movement.

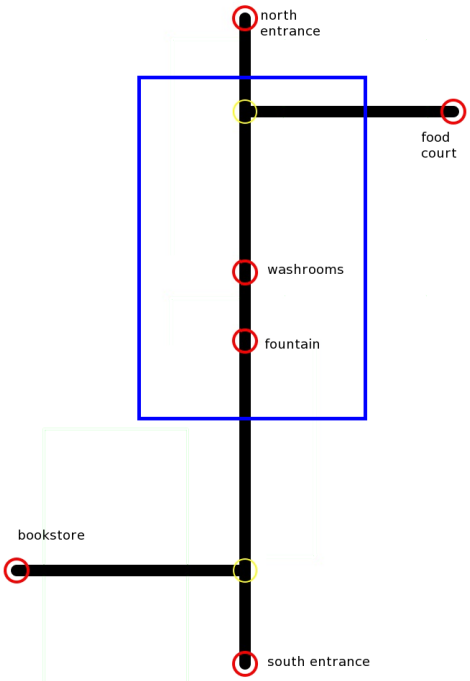
Exploration and navigation

In this study, we have focused on exploration tasks versus navigation. This is a practical consideration since Timbremap's interface currently requires significant user attention to audio

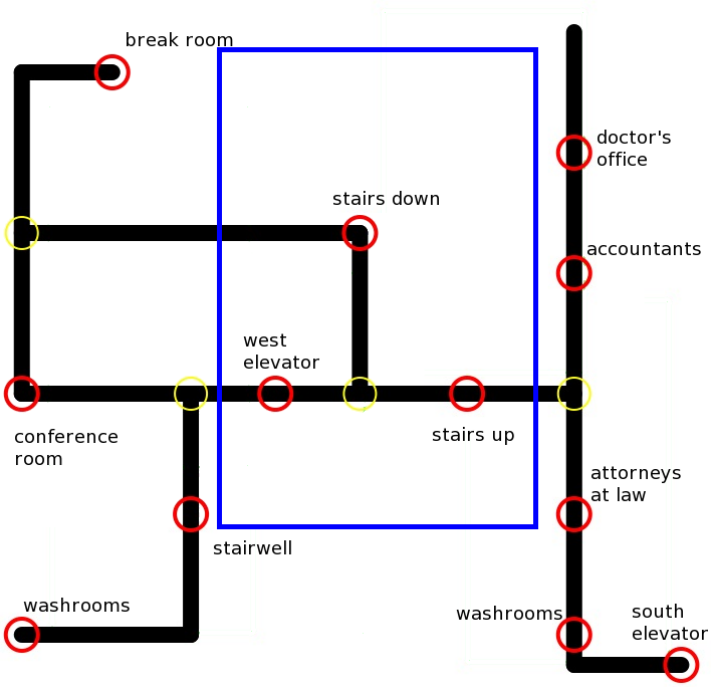
feedback and device interaction; both of which can be dangerously distracting if used while actively navigating.

Existing technologies exist for possibly addressing the safety concerns of utilizing headphones while mobile. For example, personal directional speakers or air-tube headphones can minimize the attenuation of ambient environmental sounds. Existing research efforts have suggested that air-tube headphones are less intrusive when compared to standard headphones [52]. More studies comparing navigation and exploration tasks in situ using different audio equipment will be an important piece of future work.

There are opportunities for integrating exploration systems, such as Timbremap, with navigation systems. Environmental tags to provide localization context for exploration systems such as Timbremap can also provide localization context for navigation systems where GPS localization is insufficient. Conversely, environmental tags can warn navigating users of approaching hazards, enabling the user to stop, examine the state of the hazard ahead, and plan accordingly.

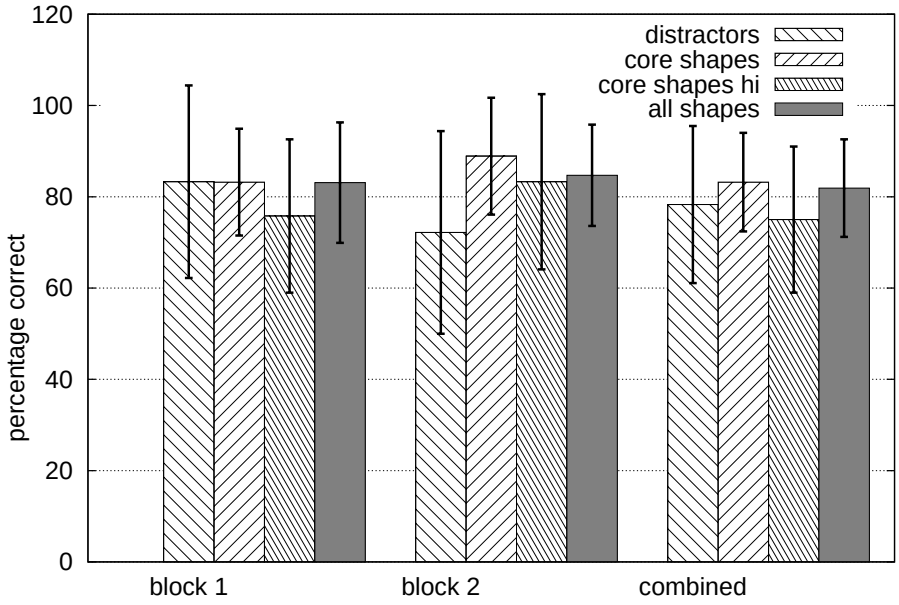


(a) map 1

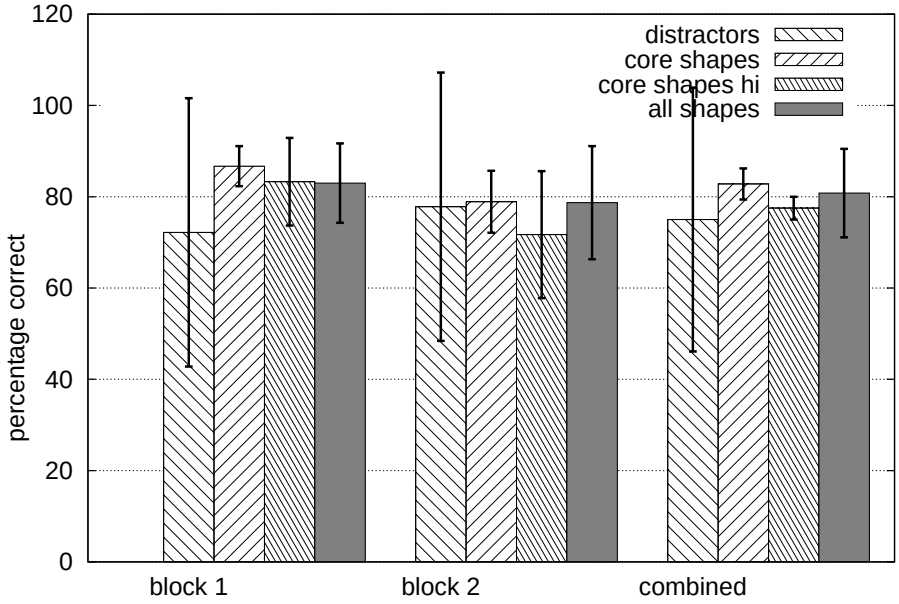


(b) map 2

Figure 5.8: Indoor floor-plans. POI and intersection markers indicated by labelled and unlabelled circles, respectively. Size of device screen coverage shown by blue box.

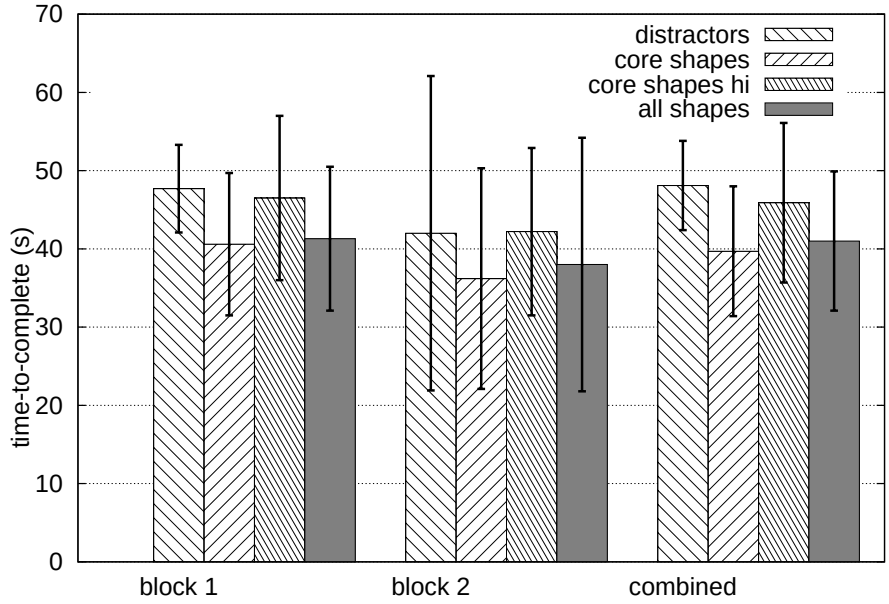


(a) shape hinting sonification

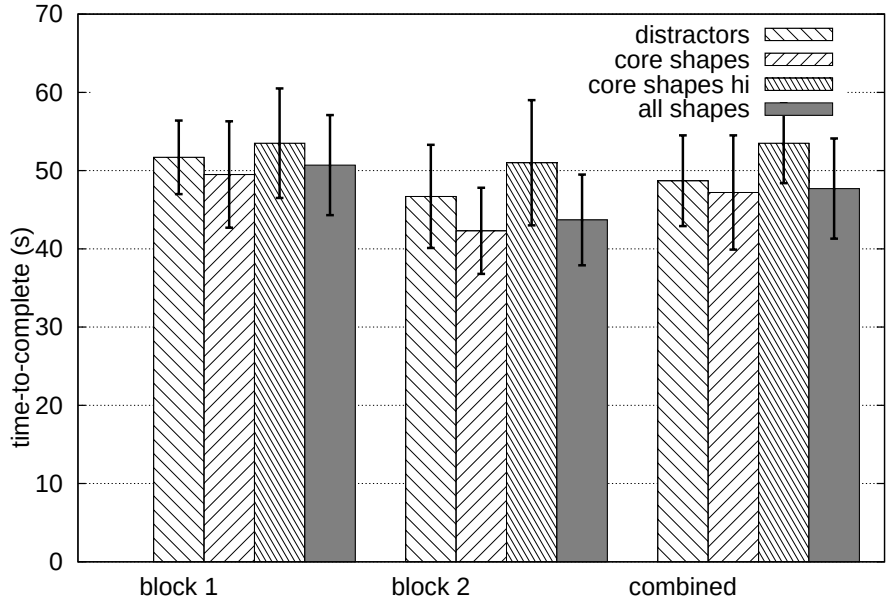


(b) color hinting sonification

Figure 5.9: *Summary of percentage answered correctly. Bars are means with 95% confidence interval*



(a) shape mode



(b) color mode

Figure 5.10: *Summary of time to answer correctly. Bars are means of median timing with 95% confidence interval*

Chapter 6

Conclusions

In this thesis, we have highlighted the rapid growth and adoption of powerful mobile devices which are equipped with multiple radio communication antennas, capable of running sophisticated software, and have dynamic input mechanisms. This has contributed to a significant increase in users utilizing mobile technology to communicate and access content while mobile and social. Unfortunately, in this dynamic computing environment, device and application behavior have not changed much from the desktop usage model, where users access their devices and applications in isolation.

Mobile users currently face many obstacles which must be manually managed in order to access and collaborate on content in a social setting. These range from manually toggling radio communication devices and their associated applications, to toggling through online identities and setting proper sharing relationships in web-based applications and services. In a social setting, these chores distract and pull the user away from the social exchange, causing an undesirable disruption.

This thesis hypothesizes that physical proximity context can help improve access to data as well as improve awareness of physical environments. We identify three layers of concerns in mobile systems – first at the device level, then applications, and finally user awareness. Within each of these layers, we identify specific challenges which impose hurdles to a smooth mobile

experience, and show how physical proximity context is used to tackle these challenges.

First, at the device layer, we identify a significant gap in systems support for the dynamic management of available on-device radio resources and their associated applications in social settings. The result is a system where users must perform significant amounts of manual management, and suffer incompatibilities in application data or operational functionality for reasons which are not always obvious. For example, if two people manage to pair their smartphone WiFi radios in ad-hoc mode, it is not clear why their emails still fail to reach one another. We address this challenge by contributing a novel device middle-ware layer which is able to utilize physical proximity context to enable the identification and routing of peers. This separation of concerns enables users to utilize familiar applications for sending and sharing data with their peers agnostic of the underlying device radio or networking model.

Second, at the application layer, we identify a significant shortcoming in the usage model of web-based applications and services in social settings. Despite the growing popularity of web-based applications and services for providing highly available access to content and convenient sharing capabilities, these web applications still burden users with the chore of identifying their peers, finding shared content, and manually setting sharing permissions. We address this challenge by contributing a novel mapping and service framework which enables identifiers and content to be referenced based on real-life identities. This mapping enables Copernicus to identify peers in social proximity, and adapt web-based applications and services at the source via their exported Web2.0 APIs. By effectively making any web-based application or service proximity aware, users are able to use their devices as a supplemental aid to smoothly find and collaborate on content with their peers in a social exchange.

Finally, at the user layer, we highlight how mobile devices can increase user awareness of their physical environment and improve their mobility and mobile experience. For example, mobile maps allow users to not only find directions but also explore new paths and neighbourhoods, and integrated recommendation applications can provide suggestions for new events or venues to experience. Unfortunately, a significant gap exists in the ability of current mobile

systems to provide physical context information to visually impaired users. There currently exists no available application for enabling visually impaired users to explore geometry and spatial relationships of features on maps. This is an even greater problem in indoor environments where no available localization system exists. We contribute a novel sonification touch interface focused on conveying precise path geometry and positioning to visually impaired users. Our user studies suggest Timbremap is effective in its motivation, and can enable visually impaired users to explore the spatial relationships of floor layouts on mobile touch-screen devices. This work provides an important first step in enabling visually impaired users to gain a greater cognitive understanding of their physical environment.

In conclusion, this thesis highlighted several unique challenges and opportunities which mobile devices enable. The mobile environment poses a usage model distinct from the familiar desktop environment. Mobile systems must consider the social and physical settings of the user, and we show how physical proximity context can be utilized to improve device and application behavior and improve user awareness of their physical context.

6.1 Future Work

In this section, we highlight several directions of future work for extending the contributions presented in this thesis.

The Copernicus framework enables opportunities to build many applications which are currently not possible. For example, a calendar application can provide journal-like features, annotating the user's calendar with people met throughout the workday, or collecting business cards for new contacts. Copernicus enables these types of application extension even if the content that's being mashed up is hosted by different providers across different domains and federations.

To perform these filtering and adaptation operations, the Copernicus service currently must pull the content or content meta-data into itself in order to analyze and filter for the desired

adaptations. An improvement would be to enable pushing of filtering and selection operations to the web applications and services themselves. This saves the third-party services and Copernicus bandwidth as well as reducing latencies in making adaptations. Many Web2.0 APIs already enable basic batching and filtering options. As these APIs mature and third-party add-ons grow in popularity, we envision growing support for sophisticated query-like interfaces.

For security, privacy, or legal reasons, various web applications may wish to run their own Copernicus-like service. A non-centralized design can mitigate the damage a user suffers if a particular service were compromised. We envision extending the use of delegation credentials to enable interoperability between Copernicus-like services. A cooperative API standard, similar to the OpenSocial effort [56], would enable accessing and searching data across trusted services.

The Timbremap prototype presented in this thesis contributed a novel first step in providing visually impaired users with a means to gain a cognitive understanding of their physical environment. From this foundation, we envision several directions of potential future work. First, one of the original design goals of Timbremap was to ensure that the technique works on size-constrained mobile devices, due to their pervasive availability and portability. However, as medium format touch interfaces enter the consumer market, such as tablets, it is worth exploring the effectiveness of Timbremap on these other form factors. Specifically, it is necessary to determine whether larger spatial surfaces impact the speed and accuracy of users with this sonification interface.

Second, the current prototype of the indoor floor-plan exploration application only maintains a single orientation – that the top of the device points north. However, as previous works have mentioned, some individuals have difficulty translating maps into a cognitive maps that they can relate to the physical space [11]. To aid these individuals, devices can automatically place the user's position and orientation on the virtual map. Even without pervasive indoor localization, Timbremap could take advantage of the built-in magnetometers to auto-rotate the map to match the user's facing direction. In the absence of accurate magnetometer readings,

Timbermap could utilize positions of known environmental tags, combined with dead reckoning, to estimate the user's current facing direction. Exploring techniques for notifying and conveying map rotation to the user, so that they may more easily orient themselves, would be a significant area of future study and work.

While there is currently no pervasively available indoor localization system, we conjecture that a coarse-grained location marker system using short-range wireless signals may be sufficient. For example, major exits and doorways in an indoor space may be marked with unique wireless tags using RFID or Bluetooth beacons. These would be sufficient to not only identify where the user is, generally, but also what floor they are on in a high-rise (a capability difficult to achieve with current GPS systems, even if the signals penetrated indoors). The user's device can then download the appropriate map information from a well-provisioned web service using wireless infrastructure links, and position the user as well as orient them using the device's built-in magnetometer. Such a system would enable visually impaired users to explore or refresh their cognitive map of a new area to plan their path – for example, figuring out how to navigate and exit an unfamiliar subway station in order to arrive at the proper street intersection.

Finally, a more ambitious future work is to utilize the built in cameras available on many mobile devices to provide basic obstacle recognition for visually impaired users. This is a particularly important challenge with regards to road hazards such as sidewalk construction, where detour boundaries may be difficult to follow and detour paths themselves fraught with many unexpected obstacles. Vision systems utilizing the mobile camera may be augmented by short-range tags that may be included in construction markers such as hazard cones, enabling the device to convey the dimensions and shapes of the detour. The user would be able to plain their traversal of the detour, or explore the larger street map to plan their own detour around the hazard altogether.

Bibliography

[1] William Adjie-Winoto, Elliot Schwartz, Hari Balakrishnan, and Jeremy Lilley. The design and implementation of an intentional naming system. *SIGOPS Oper. Syst. Rev.*, 33(5):186–201, 1999.

[2] Marcos K. Aguilera, Robert E. Strom, Daniel C. Sturman, Mark Astley, and Tushar D. Chandra. Matching events in a content-based subscription system. In *Proceedings of PODC 1999*, 1999.

[3] James L. Alty and Dimitrios I. Rigas. Communicating graphical information to blind users using music: the role of context. In *CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 574–581, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co.

[4] Ganesh Ananthanarayanan, Venkata Padmanabhan, Chandramohan Thekkath, and Lenin Ravindranath. Collaborative downloading for multi-homed wireless devices. In *HotMobile 2007*, 2007.

[5] Ganesh Ananthanarayanan, Venkata N. Padmanabhan, Lenin Ravindranath, and Chandramohan A. Thekkath. Combine: leveraging the power of wireless peers through collaborative downloading. In *MobiSys '07: Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 286–298, New York, NY, USA, 2007. ACM.

[6] Yuki Arase, Takahiro Hara, Toshiaki Uemukai, and Shojiro Nishio. Opa browser: a web browser for cellular phone users. In *UIST '07: Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 71–80, New York, NY, USA, 2007. ACM.

[7] P. Bahl and V. N. Padmanabhan. Radar: an in-building rf-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 775–784 vol.2, 2000.

[8] Hari Balakrishnan, Hariharan S. Rahul, and Srinivasan Seshan. An integrated congestion management architecture for internet hosts. *SIGCOMM Comput. Commun. Rev.*, 29(4):175–187, 1999.

[9] Nilton Bila, Troy Ronda, Iqbal Mohomed, Khai N. Truong, and Eyal de Lara. Pagetailor: reusable end-user customization for the mobile web. In *MobiSys '07: Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 16–29, New York, NY, USA, 2007. ACM.

[10] David B.Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996. edited by Tom Imielinski and Hank Korth.

[11] B. Blasch, W. Wiener, and R. Welsh. Foundations of orientation and mobility, second edition. In *American Foundation for the Blind*, pages 291–293, 1997.

[12] Yevgen Borodin, Jalal Mahmud, and I.V. Ramakrishnan. Context browsing with mobiles - when less is more. In *MobiSys '07: Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 3–15, New York, NY, USA, 2007. ACM.

[13] Simon Byers and Dave Kormann. 802.11b access point mapping. *Communications of the ACM*, 46(5):41–46, 2003.

[14] Stuart K. Card, Thomas P. Moran, and Allen Newell. The keystroke-level model for user performance time with interactive systems. *Commun. ACM*, 23(7):396–410, 1980.

[15] Stuart K. Card, Thomas P. Moran, and Allen Newell. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1983.

[16] Augustin Chaintreau, Pan Hui, Christophe Diot, Richard Gass, and James Scott. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Transactions on Mobile Computing*, 6(6):606–620, 2007. Fellow-Jon Crowcroft.

[17] Rajiv Chakravorty, Andrew Clark, and Ian Pratt. Gprsweb: optimizing the web for gprs links. In *MobiSys '03: Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 317–330, New York, NY, USA, 2003. ACM.

[18] Ranveer Chandra, Paramvir Bahl, and Pradeep Bahl. MultiNet: Connecting to Multiple IEEE 802.11 Networks Using a Single Wireless Card . In *Proceedings of IEEE Infocomm 2004*, 2004.

[19] Mike Y. Chen, Timothy Sohn, Dmitri Chmelev, Dirk Haehnel, Jeffrey Hightower, Jeff Hughes, Anthony LaMarca, Fred Potter, Ian Smith, and Alex Varshavsky. Practical metropolitan-scale positioning for gsm phones. In *Proceedings of the Eighth International Conference on Ubiquitous Computing (Ubicomp 2006)*, Lecture Notes in Computer Science, pages 225–242. Springer-Verlag, September 2006.

[20] Landon P. Cox, Angela Dalton, and Varun Marupadi. Smokescreen: flexible privacy controls for presence-sharing. In *MobiSys '07: Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 233–245, New York, NY, USA, 2007. ACM.

[21] Landon P. Cox and Brian D. Noble. Samsara: Honor among thieves in peer-to-peer storage. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, pages 120–132, New York, New York, USA, 2003. ACM Press.

[22] Andrew Crossan and Stephen Brewster. Multimodal trajectory playback for teaching shape information and trajectories to visually impaired computer users. *ACM Trans. Access. Comput.*, 1(2):1–34, 2008.

[23] Edward Cutrell, Susan T. Dumais, and Jaime Teevan. Searching to eliminate personal information management. *Commun. ACM*, 49(1), 2006.

[24] M. Demmer, E. Brewer, K. Fall, S. Jain, M. Ho, and R. Patra. Implementing delay tolerant networking. Technical Report IRB-TR-04-020, Intel Research, Berkeley, 2004.

[25] Tilman Dingler, Jeffrey Lindsay, and Bruce N. Walker. Learnabiltiy of sound cues for environmental features: Auditory icons, earcons, spearcons, and speech. In *Proceedings of the 14th International Conference on Auditory Display*, Paris, France, 2008.

[26] Kevin Fall. A delay-tolerant network architecture for challenged internets. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–34, New York, NY, USA, 2003. ACM.

[27] `http://wiki.developers.facebook.com/index.php/FQL`.

[28] Michal Feldman, Christos Papadimitriou, John Chuang, and Ion Stoica. Free-riding and whitewashing in peer-to-peer systems. In *Proceedings of the ACM SIGCOMM Workshop on Practice and Theory of Incentives in Networked Systems*, pages 228–236, New York, New York, USA, 2004. ACM Press.

[29] `http://www.flickr.com/services/api/flickr.photos.search.html`.

[30] Bryan Ford, Jacob Strauss, Chris Lesniewski-Laas, Sean Rhea, Frans Kaashoek, and Robert Morris. Persistent personal names for globally connected mobile devices. In *Proceedings of OSDI*, 2006.

[31] Michel Goemans, Li Erran Li, Vahab S. Mirrokni, and Marina Thottan. Market sharing games applied to content distribution in ad-hoc networks. In *Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 55–66, Roppongi Hills, Tokyo, Japan, 2004. ACM Press.

[32] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, 1992.

[33] `http://gears.google.com/`.

[34] Shimin Guo and Srinivasan Keshav. Fair and efficient scheduling in data ferrying networks. In *CoNEXT '07: Proceedings of the 2007 ACM CoNEXT conference*, pages 1–12, New York, NY, USA, 2007. ACM.

[35] Andreas Hub, Joachim Diepstraten, and Thomas Ertl. Design and development of an indoor navigation and object identification system for the blind. In *Assets '04: Proceedings of the 6th international ACM SIGACCESS conference on Computers and accessibility*, pages 147–152, New York, NY, USA, 2004. ACM.

[36] Pan Hui, Augustin Chaintreau, James Scott, Richard Gass, Jon Crowcroft, and Christophe Diot. Pocket Switched Networks and human mobility in conference environments. In *Proceedings of WDTN 2005*, 2005.

[37] Dan Jacobson. Navigating maps with little or no sight: A novel audio-tactile approach. In *Proceedings of Content Visualization and Intermedia Representations*, 1998.

[38] Sushant Jain, Kevin Fall, and Rabin Patra. Routing in a delay tolerant network. In *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 145–158. ACM Press, 2004.

[39] Amit P. Jardosh, Krishna N. Ramachandran, Kevin C. Almeroth, and Elizabeth M. Belding-Royer. Understanding Congestion in IEEE 802.11b Wireless NetworksRevised. In *Proceedings of IMC 2005*, 2005.

[40] Shaun K. Kane, Jeffrey P. Bigham, and Jacob O. Wobbrock. Slide rule: making mobile touch screens accessible to blind people using multi-touch interaction techniques. In *Assets '08: Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility*, pages 73–80, New York, NY, USA, 2008. ACM.

[41] Amy Karlson, Greg Smith, Brian Meyers, George Robertson, and Mary Czerwinski. Courier: A collaborative phone-based file exchange system. Technical Report MSR-TR-2008-05, Microsoft Research, 2008.

[42] Amy K. Karlson, George G. Robertson, Daniel C. Robbins, Mary P. Czerwinski, and Greg R. Smith. Fathumb: a facet-based interface for mobile search. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 711–720, New York, NY, USA, 2006. ACM.

[43] Martin Karsten, S. Keshav, and Sanjiva Prasad. An axiomatic basis for communication. In *Proceedings of HotNets 2006*, 2006.

[44] Martin Karsten, S. Keshav, Sanjiva Prasad, and Mirza Beg. An axiomatic basis for communication. In *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 217–228, New York, NY, USA, 2007. ACM.

[45] John Krumm and Ken Hinckley. The nearme wireless proximity server. In *UbiComp 2004: Proceedings of the 6th International Conference on Ubiquitous Computing*, pages 283–300, 2004.

[46] Eyal De Lara, Dan S. Wallach, and Willy Zwaenepoel. Puppeteer: Component-based adaptation for mobile computing. In *USITS'01: Proceedings of the 3rd conference on USENIX Symposium on Internet Technologies and Systems*, pages 14–14, Berkeley, CA, USA, 2001. USENIX Association.

[47] Jérémie Leguay, Timur Friedman, and Vania Conan. Dtn routing in a mobility pattern space. In *Proceedings of WDTN 2005*. ACM Press, 2005.

[48] Frank Chun Yat Li, David Dearman, and Khai N. Truong. Virtual shelves: interactions with orientation aware devices. In *UIST '09: Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 125–128, New York, NY, USA, 2009. ACM.

[49] Anders Lindgren, Avri Doria, and Olov Schelén. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(3):19–20, 2003.

[50] `http://www.loadstone-gps.com/`.

[51] J. M. Loomis, R. G. Golledge, R. L. Klatzky, J. M. Speigle, and J. Tietz. Personal guidance system for the visually impaired. In *Assets '94: Proceedings of the first annual ACM conference on Assistive technologies*, pages 85–91, New York, NY, USA, 1994. ACM.

[52] James R. Marston, Jack M. Loomis, Roberta L. Klatzky, Reginald G. Golledge, and Ethan L. Smith. Evaluation of spatial displays for navigation without sight. *ACM Trans. Appl. Percept.*, 3(2):110–124, 2006.

[53] M. Mauve, A. Widmer, and H. Hartenstein. A survey on position-based routing in mobile ad hoc networks. *Network*, 15(6), Nov 2001.

[54] Iqbal Mohomed, Jim Chengming Cai, Sina Chavoshi, and Eyal de Lara. Context-aware interactive content adaptation. In *MobiSys '06: Proceedings of the 4th international conference on Mobile systems, applications and services*, pages 42–55, New York, NY, USA, 2006. ACM.

[55] `http://www.ndis.com/`.

[56] `http://code.google.com/apis/opensocial/`.

[57] Peter Parente and Gary Bishop. BATS: The Blind Audio Tactile Mapping System. In *Proceedings of ACM South Eastern Conference*, 2003.

[58] Shwetak N. Patel and Gregory D. Abowd. The contextcam: Automated point of capture video annotation. In *UbiComp 2004: Proceedings of the 6th International Conference on Ubiquitous Computing*, pages 301–318, 2004.

[59] Shwetak N. Patel, Julie A. Kientz, Gillian R. Hayes, Sooraj Bhat, and Gregory D. Abowd. Farther than you may think: An empirical investigation of the proximity of users to their mobile phones. In *UbiComp 2006: Proceedings of the 8th International Conference on Ubiquitous Computing*, pages 123–140, 2006.

[60] `http://www.crynwr.com/packet_driver.html`.

[61] Ying Qiu and Peter Marbach. Bandwidth allocation in ad hoc networks: a price-based approach. In *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 797–807 vol.2. IEEE, 2003.

[62] Asfandyar Qureshi and John Guttag. Horde: separating network striping policy from mechanism. In *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 121–134, New York, NY, USA, 2005. ACM.

[63] Ip mobility support for ipv4. RFC 3344, IETF, 2002.

[64] Ad hoc on-demand distance vector (aodv) routing. RFC 3561, IETF, 2003.

[65] Mobility support in ipv6. RFC 3775, IETF, 2004.

[66] David A. Ross and Bruce B. Blasch. Wearable interfaces for orientation and wayfinding. In *Assets '00: Proceedings of the fourth international ACM conference on Assistive technologies*, pages 193–200, New York, NY, USA, 2000. ACM.

[67] Mattias Rost, Mattias Jacobsson, and Lars Erik Holmquist. Push!photo: Informal photo sharing in ad-hoc networks. In *UbiComp 2006: Proceedings of the 8th International Conference on Ubiquitous Computing*, 2006.

[68] Risto Sarvas, Erick Herrarte, Anita Wilhelm, and Marc Davis. Metadata creation system for mobile images. In *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 36–48, New York, NY, USA, 2004. ACM.

[69] James Scott, Pan Hui, Jon Crowcroft, and Christophe Diot. Haggle: a networking architecture designed around mobile users. *Proceedings of the Third Annual Conference on Wireless On demand Network Systems and Services*, 2006. Invited paper.

[70] A. Seth, D. Kroeker, M. Zaharia, S. Guo, and S. Keshav. Low-cost communication for rural internet kiosks using mechanical backhaul. In *MobiCom '06: Proceedings of the 12th annual international conference on Mobile computing and networking*, pages 334–345, New York, NY, USA, 2006. ACM.

[71] Rahul C. Shah, Sumit Roy, Sushant Jain, and Waylon Brunette. Datamules: Modelling a three tiered architecture for sparse sensor networks. In *IEEE SNPA 2003*, 2003.

[72] Timothy Sohn, Alex Varshavsky, Anthony LaMarca, Mike Y. Chen, Tanzeem Choudhury, Ian E. Smith, Sunny Consolvo, Jeffrey Hightower, William G. Griswold, and Eyal

de Lara. Mobility detection using everyday gsm traces. In *UbiComp 2006: Proceedings of the 8th International Conference on Ubiquitous Computing*, pages 212–224, 2006.

[73] Jacob Sorber, Nilanjan Banerjee, Mark D. Corner, and Sami Rollins. Turducken: hierarchical power management for mobile devices. In *Proceedings of MobiSys 2005*, 2005.

[74] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 252–259, New York, NY, USA, 2005. ACM.

[75] Ion Stoica, Daniel Adkins, Shelley Zhuang, Scott Shenker, and Sonesh Surana. Internet indirection infrastructure. *SIGCOMM Comput. Commun. Rev.*, 32(4):73–86, 2002.

[76] Jing Su, Alvin Chin, Anna Popivanova, Ashvin Goel, and Eyal de Lara. User mobility for opportunistic ad-hoc networking. In *WMCSA '04: Proceedings of the Sixth IEEE Workshop on Mobile Computing Systems and Applications*, pages 41–50, Washington, DC, USA, 2004. IEEE Computer Society.

[77] Jing Su, James Scott, Pan Hui, Jon Crowcroft, Eyal De Lara, Christophe Diot, Ashvin Goel, Meng How Lim, and Eben Upton. Haggle: seamless networking for mobile applications. In *UbiComp'07: Proceedings of the 9th international conference on Ubiquitous computing*, pages 391–408, Berlin, Heidelberg, 2007. Springer-Verlag.

[78] Khai N. Truong and Gregory D. Abowd. Inca: A software infrastructure to facilitate the construction and evolution of ubiquitous capture & access applications. In *Pervasive 2004: Proceedings of the 2nd International Conference on Pervasive Computing*, pages 140–157, 2004.

[79] `http://www.projectudi.org/`.

[80] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical report, Duke University, 2000. CS-200006.

[81] Alex Varshavsky, Eyal de Lara, Anthony LaMarca, Jeffrey Hightower, and Veljo Otsason. Gsm indoor localization. *Pervasive and Mobile Computing Journal*, 3(6):698–720, 2007.

[82] Alex Varshavsky, Adin Scannell, Anthony LaMarca, and Eyal de Lara. Amigo: Proximity-based authentication of mobile devices. In *UbiComp 2007: Proceedings of the 9th International Conference on Ubiquitous Computing*, pages 253–270, 2007.

[83] Kaushik Veeraraghavan, Andrew Myrick, and Jason Flinn. Cobalt: separating content distribution from authorization in distributed file systems. In *FAST'07: Proceedings of the 5th conference on USENIX Conference on File and Storage Technologies*, pages 29–29, Berkeley, CA, USA, 2007. USENIX Association.

[84] B. N. Walker and J. Lindsay. The effect of a speech discrimination task on navigation in a virtual environment. In *Proceedings of the Annual Meeting of the Human Factors and Ergonomics Society (HFES2006)*, pages 1538–1541, San Francisco, CA, 2006.

[85] B. N. Walker, A. Nance, and J. Lindsay. Spearcons: speech-based earcons improve navigation performance in auditory menus. pages 63–68, London, UK, 2006. Department of Computer Science, Queen Mary, University of London, UK, Department of Computer Science, Queen Mary, University of London, UK.

[86] H. J. Wang, R. H. Katz, and J. Giese. Policy-enabled handoffs across heterogeneous wireless networks. In *WMCSA '99: Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*, page 51, Washington, DC, USA, 1999. IEEE Computer Society.

[87] Randolph Y. Wang, Sumeet Sobti, Nitin Garg, Elisha Ziskind, Junwen Lai, and Arvind Krishnamurthy. Turning the postal system into a generic digital communication mecha-

nism. In *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 159–166. ACM Press, 2004.

[88] J. Wilson, B.N. Walker, J. Lindsay, C. Cambias, and F. Dellaert. Swan: System for wearable audio navigation. In *Wearable Computers, 2007 11th IEEE International Symposium on*, pages 91–98, Oct. 2007.

[89] Koji Yatani and Khai Nhut Truong. Semfeel: a user interface with semantic tactile feedback for mobile touch-screen devices. In *UIST '09: Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 111–120, New York, NY, USA, 2009. ACM.

[90] Shengdong Zhao, Pierre Dragicevic, Mark Chignell, Ravin Balakrishnan, and Patrick Baudisch. Earpod: eyes-free menu selection using touch input and reactive audio feedback. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1395–1404, New York, NY, USA, 2007. ACM.

[91] W. Zhao, M. Ammar, and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *Proceedings of MobiCom 2004*, 2004.