DomeView: Community-Based Digital Bulletin Boards and Mobile Phone Interaction

by

Harel M. Williams

Submitted to the Department of Electrical Engineering and Computer Science

in Partial fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May, 2007

©2007 Massachusetts Institute of Technology. All rights reserved.

Author___

Electrical Engineering and Computer Science
May 25, 2007

Certified by_______

P

Larry Rudolph
.I Laboratory
s Supervisor

Accepted by______________

:hur C. Smith
Professor of Electrical Engineering
Chairman, Department Committee of Graduate Theses

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
OCT 0 3 2007
LIBRARIES

BARKER

DomeView: Community-Based Digital Bulletin Boards and Mobile Phone Interaction

by

Harel M. Williams

Submitted to the
Department of Electrical Engineering and Computer Science
May 25, 2007
In Partial fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Our thesis is that a networked public display/kiosk system, that provides information for a local community, functions best when it is decentralized and interactive. We deployed such a system at MIT that has two aspects, DomeView for distributed decentralized display and content distribution, and PhoneView for enhanced user consumption of that content. PhoneView is an implementation that we propose to solve a number of issues with current interactive public kiosk deployments, as well as enables scenarios of enhanced interactions. By using the Hands-Free Bluetooth profile as the basis for the communication between a mobile phone and a kiosk, we provide an enhanced personalized interaction for all passersby with Bluetooth-enabled mobile phones, without requiring the installation of custom software. Some examples include the ability to remotely control a kiosk, exchange calendar and contact data with the kiosk, and play games on a kiosk with other users via one's mobile phone. By removing the software installation barrier and providing new mechanisms of public interaction, this implementation is ripe for wide-spread and immediate adoption across multiple public kiosk platforms.

Thesis Supervisor: Larry Rudolph
Title: Principal Research Scientist at the MIT Computer Science and Artificial Intelligence Laboratory

ACKNOWLEDGEMENTS

First, I would like to thank Larry Rudolph, who is probably the best thesis advisor I could have possibly asked for and is the one who came up with the idea for the amazing hack that makes PhoneView possible. Additionally, there was a very high likelihood that I wouldn't have stayed to pursue my masters if I hadn't met him last April. He allowed me to pursue a thesis topic that I was deeply interested in, and inspired me to look at my project in a whole new light. It was always a pleasure swinging by his office to discuss some new ideas and talking about the future of public kiosks and pervasive computing.

Second, I would like to thank my friends from the Undergraduate Association, most notably John Velasco, as well as the administrators who were involved in the DomeView project, which served as the spring board for my Masters research as well as a source of accomplishment during my graduate career. Also, a special thanks to the Microsoft-MIT iCampus Alliance for providing the funding for the DomeView project, which resulted in a much needed resource for the MIT community.

Last, but not least, I would like to thank my family for supporting me financially as well as emotionally through all 6 years of my MIT career: Mom, Dad, Marc, and Aunt Judy. Also, a very special thanks to my wonderful girlfriend Tammy, who made sure I was cheery as well as timely in the completion of my thesis.

TABLE OF CONTENTS

LIST OF FIGURES

1 INTRODUCTION

This thesis introduces a new paradigm of interaction between public kiosks and users via mobile phones, which provides significant benefits over currently available modes of interaction. We have deployed our implementation on public kiosks throughout the Massachusetts Institute of Technology (MIT) college campus, and explore further enhancements and extensions with our implementation as a foundation.

1.1 DIGITAL SIGNAGE AND PUBLIC KIOSKS

As the world population becomes more technically savvy and the cost of computational equipment and digital displays continue to drop, we have seen a proliferation of the deployment of digital displays and kiosks in many varying and unique locations. Some of the displays and kiosks are provided as a service to individuals (ATMs, campus maps, public transportation assistance), while others act as glorified advertising space (Times Square in New York City). However, most of these deployments do not strive to enhance the interactions and knowledge of individuals around their communities.

In this thesis, we introduce DomeView, a digital bulletin board system tailored specifically to the MIT community, which could be easily adapted as a service on other college campuses. The system is unique in that every member of the MIT community has the opportunity to use the system to advertise his/her groups' events and announcements, essentially having a sense of ownership over the system. However, up until now, there was limited interaction available at the sites of the DomeView kiosks, which served more as a passive medium for viewing information. The PhoneView interaction mechanism that we created uses mobile phones as its vehicle, and is at the core of this thesis.

1.2 MOBILE PHONES AND DEVICES

At the current time, mobile phones have become a commodity in most first world countries, and almost everyone has one on his/her person at all times. Additionally, mobile phones are not only used for their telephonic functions anymore, but many come equipped with additional features including personal data assistant capabilities, as well as media capture mechanisms. In part because of these advancements, mobile phones have become highly-personalized devices. People spend many hours and dollars adding their own touches to their mobile phones by modifying the cases, creating/downloading ringtones, as well as developing, purchasing, and installing mobile applications.

Although not strictly associated with mobile phones, Bluetooth has played an important role in enhancing these devices to be more convenient for general and advanced use. Most mobile phones on sale today come standard with Bluetooth capability, generally enabling the use of Bluetooth headsets, as well as other functionality. Although Bluetooth will not be the short-range communication protocol of choice forever, we believe it is not going to be replaced for some time, and therefore serves as a good platform upon which to base our interaction with public kiosks.

1.3 Mobile Phones as an Interaction Vehicle

We believe that with the current developments in and the massive proliferation and adoption of mobile phones and public kiosks, these two technologies can be used to leverage one and other on a broad scale. We have discovered a method whereby virtually any Bluetooth-enabled phone can interact with any Bluetooth-configured kiosk. Up until now, there have been a couple of deployments that create interactions between mobile phones and kiosks, but they have either been extremely limited or required specialized software for different mobile platforms. We hope our implementation helps others explore the possibilities of this type of interaction in further depth, and possibly encourage speedier adoption of a clearer and possibly more feature-rich implementation that enables the usage scenarios we have discussed in this thesis.

2 Related Work

There has been a tremendous amount of research concerning public kiosks, Bluetooth, and mobile phones as individual fields of study, but there is surprisingly little directly related work, as far as mobile devices interacting with public kiosks via Bluetooth is concerned. We think this is primarily because all previous solutions required custom-made software for individual mobile phones and therefore could not be widely deployed. We believe our implementation could be leveraged by most of these projects in order to accomplish their goals in a broader setting. We also pulled research concerning interaction with public kiosks, but not necessarily via Bluetooth, in order to illustrate related concepts and to learn from the successes and pitfalls of other deployments.

2.1 Public Kiosks With Standard Interaction

As mentioned previously, there is little directly related work to our implementation, but there are a couple of public kiosk deployments that helped us identify our implementation as a unique and beneficial solution to a number of interaction problems. Additionally, a number of ideas expressed in previous kiosk research were utilized in forming the principles at the basis of the DomeView system.

2.1.1 IBM Blueboards

Recognizing that the deployment of large format displays as public kiosks would become increasingly popular as the cost of displays continue to drop, a couple of researchers at IBM decided to investigate by deploying a system called IBM Blueboards. The displays do not have a keyboard or mouse attached to them, but rather users identify themselves via their corporate RFID badges, and interact via touch-screen. The concept behind the displays is to provide very quick interactions, such as allowing a user to quickly check some personal data, such as calendar items, or encourage spontaneous collaborative situations, such as multiple users sharing notes relevant to a specific project. The Blueboards are connected to the network, where it pulls all of the relevant information. The quick interaction concept is an idea that we wanted to adopt in our implementation, if possible. The ability to uniquely identify users is another feature that we hoped to employ. However, this system is limited to viewing and manipulating the data that is already located on the network, and there is no way for a user to take the data with him/her, but rather is required to log on to the network somewhere else and look up the information again. (1)

Figure 1: IBM Blueboards

2.1.2 Notification Collage

The Notification Collage is a system that acts as a bulletin board, where the bulletin board appears to be a collage of information. Users post items, such as web pages, notes, and video feeds, to the bulletin board through various client programs. When the board becomes filled up with items, the newer items are placed on top of older ones. The researchers directing the project deployed a couple large format screens that displayed the bulletin board, as well as allowed individual users to see a reflection of the bulletin board on their personal computers. The users were able to adjust the layout of the items on their bulletin board, as well as hide certain elements. Although DomeView is a bulletin board system, the display of information is much more structured, and user submissions are managed. Although the collage is much more focused on promoting awareness among users in multiple locations, some of the concepts they investigated are relevant to the implementation of DomeView. (2)

Figure 2: Notification Collage

2.2 Public Kiosks with Mobile Device Interaction

In this section we discuss some of the more directly relevant projects to our implementation. All of the research described include some sort of interaction with kiosks via mobile phones or Personal Digital Assistants (PDA), but that interaction is often severely limited. We hope to extend upon some of the ideas presented in these projects quite significantly with our implementation.

2.2.1 Kimono

Kimono was a public kiosk system with mobile phone interaction capabilities that was deployed on MIT's campus in 2005. The idea behind Kimono was to enhance the interaction with public kiosks by transferring data between a user's mobile phone and the kiosk. The architecture of Kimono was highly geared towards the transfer of information between kiosk and mobile phone based on the time and place of the interaction. The engines running on the kiosk and the mobile phone were derived from the same code base, thus allowing the mobile phone to be a reflection of the kiosk, as well as easing software updates. Not only was a user able to download information about events displayed on the kiosk, but the user could submit comments as well as photos related to events to the kiosk so that other users could view contributions in the future. Although this architecture provided for some interesting user scenarios, it required that custom software be developed for and installed on a specific mobile platform. (3)

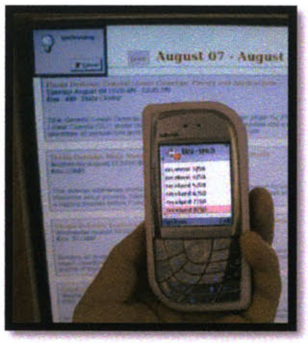


Figure 3: Kimono System

2.2.2 Plasma Posters

Elizabeth Churchill set up a digital community bulletin board system in Fujitsu Xerox Palo Alto Laboratory, called Plasma Posters. The purpose behind the system was quite similar to that of DomeView, with the added focus of creating interactions around the physical space of the displays. Members of the laboratory would submit posts to three screens distributed throughout the laboratory through email as well as web interfaces. These posts consisted of static and animated media, as well as URLs to web pages, which were visible on the Posters. People could then interact with those posts via touch-screen, with the option of scrolling through the posts as well as sending comments to the submitter of the post. An extension to the system was created, enabling users to annotate posts via their PDA. However, the PDA required Wi-Fi access to the network, as well as customized software to be installed on the PDA. Although the idea of allowing users to annotate posts via a PDA is quite interesting, the restrictive requirement of a device with Wi-Fi access and custom software is rather limiting in a public deployment. (4)

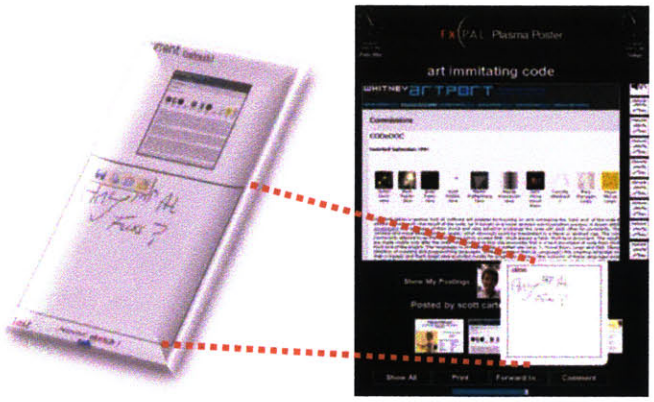


Figure 4: Digital Graffiti on Plasma Posters

2.2.3 Photo Sharing

A group of researchers at Lancaster University deployed a display to be utilized for photo sharing among members of the community. Users could potentially add photos to the display via Multimedia Message Service (MMS), email, as well as send photos from their mobile phone via Bluetooth. The display was also equipped with a touch-screen, which enabled a user to scroll through the photos, as well as send photos from the display to his/her mobile phone. The researchers measured the approximate time of sending a receiving photo via Bluetooth to be a minute, which included 25 seconds for discovery, and around another 30 seconds to transmit a 250 Kbyte photo. This amount of time seems excessively long, and is something that should be kept in mind when sending and receiving media. Although an interesting deployment, the interaction with the screen via the mobile phone was quite limited. (5)

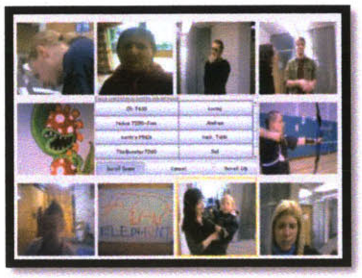


Figure 5: Hermes Photo Display

2.2.4 MobiLenin

MobiLenin is a system created by musician and engineer **Jürgen Scheible**, which enables a group of users to interact with a music video via their mobile phones. Each mobile phone has an application which communicates via HTTP over GPRS with a server, enabling the users to make changes to what is displayed on the screen, specifically the type of events that are occurring in the music video. Scheible specifically mentioned that he chose to communicate over HTTP rather than Bluetooth because each Bluetooth device is limited to 7 connections, and there is a distance restriction for interaction. Although these are legitimate concerns, our implementation deals with creating interactions near the screens, and one could increase the number of connections available via Bluetooth by installing more adapters at the kiosk. (6)

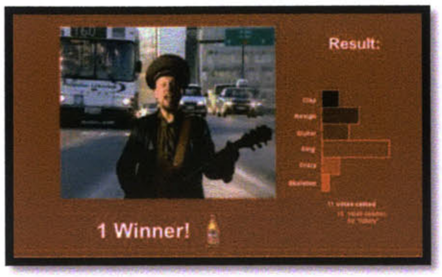


Figure 6: MobiLenin

3 Background

On a university campus such as MIT, there are so many opportunities for community members to learn and to participate in, that it is often overwhelming. The MIT community is made up of over 4,000 undergraduates, 6,000 graduate students, and 900 faculty members, and it is often difficult to communicate with all or a subset of these individuals. The primary methods of publicity for organizers of different groups and events are:

- ***Email.*** There are thousands of email lists on the MIT servers, each with its own specific membership. The email lists are often used to publicize events, by the members of these lists as well as those who aren't members. However, because of clutter caused by spam emails, individuals often ignore emails that aren't from a previously known or trusted source.
- ***Poster boards.*** There are poster boards all across campus where MIT community members and non-members can post flyers. These boards are administered by the Association of Student Activities, a group of students elected by their peers. Although postering is somewhat effective, most boards are totally covered haphazardly, and it is often difficult for passersby to pick out posters that interest them. Additionally there is much competition for poster space among different groups, leading groups to go around postering at the wee hours of the night after the facilities staff clears the boards.
- ***Newspaper Ads.*** The MIT Tech supports itself by selling ad space. This space has a significant cost, and thus can't be taken advantage of by all groups. However, it is important to note that the locally relevant content (including ads) of the newspaper increases the likelihood that one will pick up the newspaper and spend time reading it.
- ***Word of Mouth.*** This of publicity is probably the most effective method, but it doesn't scale very well. If your friend tells you about an event, you are more likely to attend than if you saw some random flyer or email from someone you don't know.

Even if one were to engage in all of these modes of communication when publicizing, it seems to put an undue burden on the organizer who is trying to advertise and may add a substantial amount of noise to the physical and mental environment of the consumer. There seemed like there must be a technical solution that could combine all the positive elements of current publicity mechanisms, while making the advertising and the consumption of that advertising more efficient and effective. Thus, the idea of DomeView was born. Although DomeView may not address all of the drawbacks of traditional publicity mechanisms, it takes steps toward that goal.

3.1 DomeView

DomeView is a web-based distributed media display and communications system tailored for the MIT community to better facilitate notification between groups and individuals about campus events, announcements, and opportunities. It consists of digital displays distributed throughout campus, which present visual media, provided by groups through an intuitive web interface, allowing passersby to easily absorb the information presented on screen. A parallel web portal provides an additional medium to view the information shown on the displays.

Currently, we have five displays deployed in two main locations. We have three 32 inch LCD displays (Zenith Z32LZ5R) distributed throughout the first floor of MIT's Student Center, a naturally high traffic area (Figure 7). At least one screen is visible from each front entrance to the Student Center.

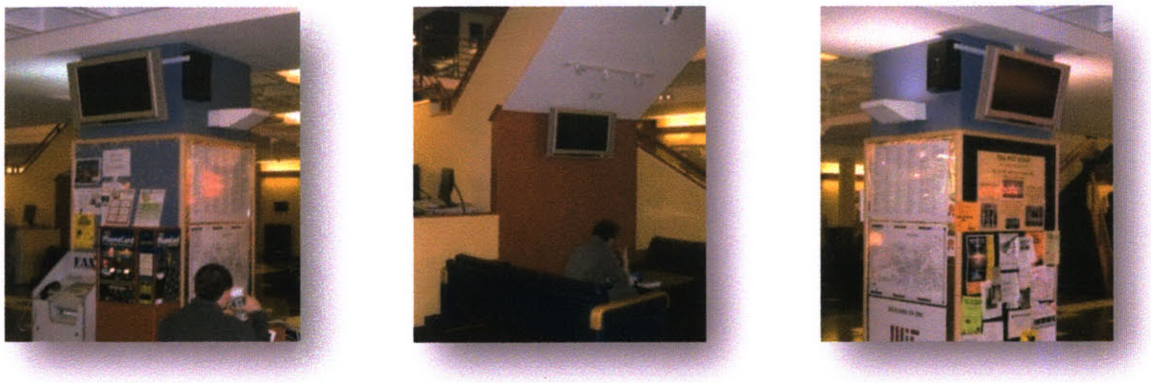


Figure 7: DomeView Displays in Student Center

We also have two 46 inch LCD displays (Samsung 460P) deployed in the Stata Center, the computer science and philosophy building at MIT. One display is located by the Café and the other is located in the Main Student Street. There are always people gathered by the Café, mainly waiting to order coffee and the Student Street is the highest traffic area within the Stata Center.

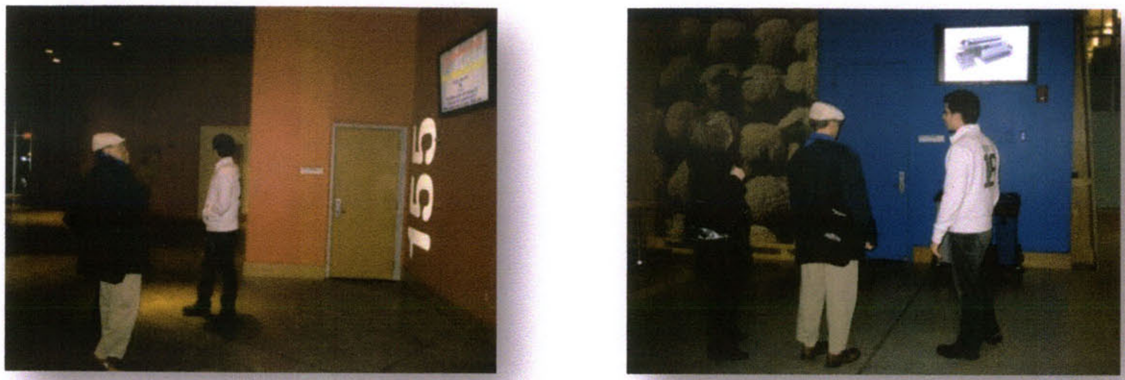


Figure 8: DomeView Displays in Stata Center

3.2 DomeView Design Principles

There are a number of principles that we adhered to in designing and deploying the DomeView system. Some of the principles weren't fully realized in our initial deployment, which is why PhoneView, the mobile phone integration presented in this thesis is so essential. We believe that with the new implementation proposed in this thesis, all of the principles have been accomplished:

- ***Augmentation of Current Publicity Mechanisms***. DomeView should not displace current mechanisms of advertising, but augment them significantly. The deployment of DomeView should significantly enhance other mechanisms of publicity by removing significant noise from the environment.
- ***24/7 System Reliability***. The DomeView kiosks should very rarely, if ever, go down. Since DomeView is a networked system, if the network connection goes down, the kiosks should continue functioning, without any perceivable errors to the users. Also, the displays should be able to automatically restore once the network connection returns. In the rare cases that the kiosks do crash, a simple reset should return it to an operating state.
- ***Easy Deployment of Additional Kiosks***. The Deployment of DomeView displays around campus should be as easy as possible, thus encouraging groups to add their own displays to DomeView. Easy deployment entails the ability of the kiosk software to run on multiple platforms with minimal system requirements.

- ***Community Ownership***. As compared to most public kiosk systems today, DomeView should be a system that is accessible to all members of a community, in terms of a system that can be used for advertising as well as a tool for discovering what is happening on campus.
- ***Eliminate Potential for Abuse***. At MIT, posterboard usage is often abused, making it necessary for the creation of a board that monitors the poster boards. We are all familiar with the advertising abuse that occurs via email, resulting in the annoyance known as spam.
- ***Intuitive Submission Mechanism***. The mechanism by which users submit material to the DomeView system should be very easy and should only be an extension of their typical tasks of publicity. For example, a user would generate a graphic to print on a paper poster to advertise; the user should be able to take that piece of media he/she has already created and submit it to the DomeView system, electronically.
- ***Active Interaction at the Kiosks***. DomeView was always meant to be a system that enabled interaction at the site of the displays. This mechanism, however, was delayed because the other aspects of the system took higher priority due to their prerequisite nature. It also took some time to find the proper venue for our interaction, because of the physical location of the screens as well as coming up with a mechanism that was a significant improvement to the popular interaction method via touch-screen.
- ***Interaction Not Dependent on Network***. If the network connection is severed, we do not want to prevent users from interacting with the kiosks. Therefore, the venue for interacting with the screen must not indirectly move across the network first, but must communicate directly with the display. Certain interaction mechanisms available today use Wi-Fi or cellular networks to interact with the displays indirectly. We decided that such a solution wouldn't be flexible enough, and decided that any interaction mechanism must communicate with the kiosks directly.

3.3 DomeView Users

The users of the DomeView system can be divided into two categories: producers and consumers. Ideally, all members of the MIT community would be producers as well as consumers in some sense. This concept is what makes DomeView unique as compared to other kiosk deployments, where there are a few people who create the content to be displayed on the kiosk, thus making the overwhelming majority of users, consumers of the content.

3.3.1 PRODUCERS

Producers are those who use the DomeView system in order to publicize on behalf of their group by submitting a post to the system. These groups consist of student groups, labs, academic departments, and administrative departments. We currently have over 500 groups and 2000 users registered on our system.

3.3.1.1 MANAGING RESOURCES

Since there are a limited number of displays and quite a few groups vying to post to the screens, we needed a way to manage resources so that the displays were not over-utilized, or worse, unavailable to most users for publicity.

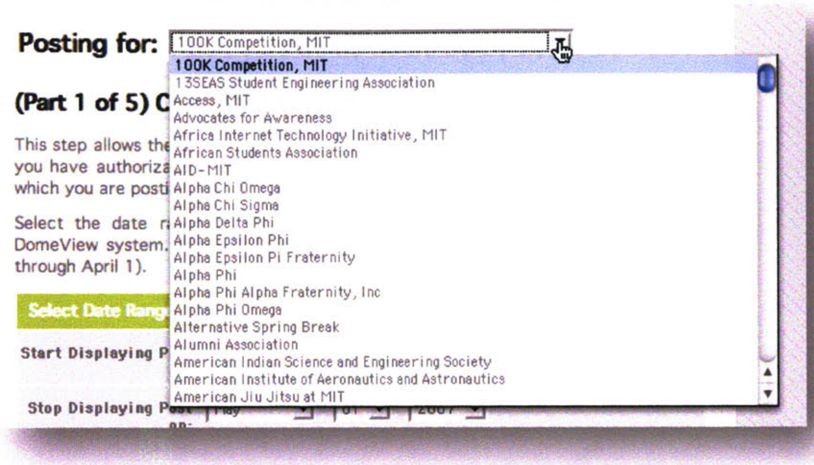
To solve this problem, we came up with the concept of distributing points to groups which could be used in order to post to the displays. Each display has a field in the database for how many posts it will display per day (currently set to 20 system-wide). Additionally, each display has a defined cost, which is how many points a group must pay in order to display a single post for an entire day on a specific display. Therefore, if a display is more popular, the cost for that display could be increased (at the moment all display costs are set to one point). This point concept also has the added benefit of discouraging individuals from submitting bogus posts because they would ultimately be wasting a valuable resource of their respective group.

Points are allotted to student groups each semester, as the system is meant to be free for student use. Other MIT affiliate groups can purchase points, and student groups can purchase additional points once they use up their allotment.

3.3.1.2 POST SUBMISSION PROCESS

In order to better understand what it is like to be a producer on the DomeView system, we will demonstrate the steps involved in submitting a post through the web interface:

1. ***Select which group on behalf of which you would like to post***


Posting for: 100K Competition, MIT

(Part 1 of 5) C

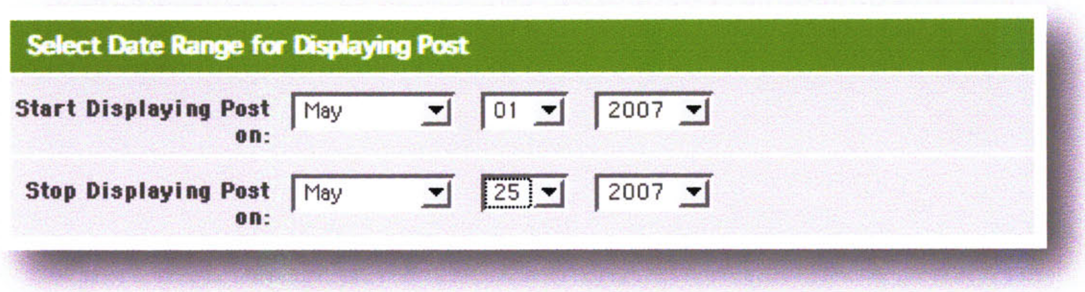
This step allows the
you have authoriza
which you are posti

Select the date r
DomeView system.
through April 1).

100K Competition, MIT
13SEAS Student Engineering Association
Access, MIT
Advocates for Awareness
Africa Internet Technology Initiative, MIT
African Students Association
AID-MIT
Alpha Chi Omega
Alpha Chi Sigma
Alpha Delta Phi
Alpha Epsilon Phi
Alpha Epsilon Pi Fraternity
Alpha Phi
Alpha Phi Alpha Fraternity, Inc
Alpha Phi Omega
Alternative Spring Break
Alumni Association
American Indian Science and Engineering Society
American Institute of Aeronautics and Astronautics
American Jiu Jitsu at MIT
Select Date Rang
Start Displaying P
Stop Displaying P

A user is able to select from the groups for which he/she is a member. This membership is determined by syncing with certain ACL databases on campus, as well as membership specified by group administrators. All members of the MIT community can request to join a group. The request would then be sent to the group administrator for approval.

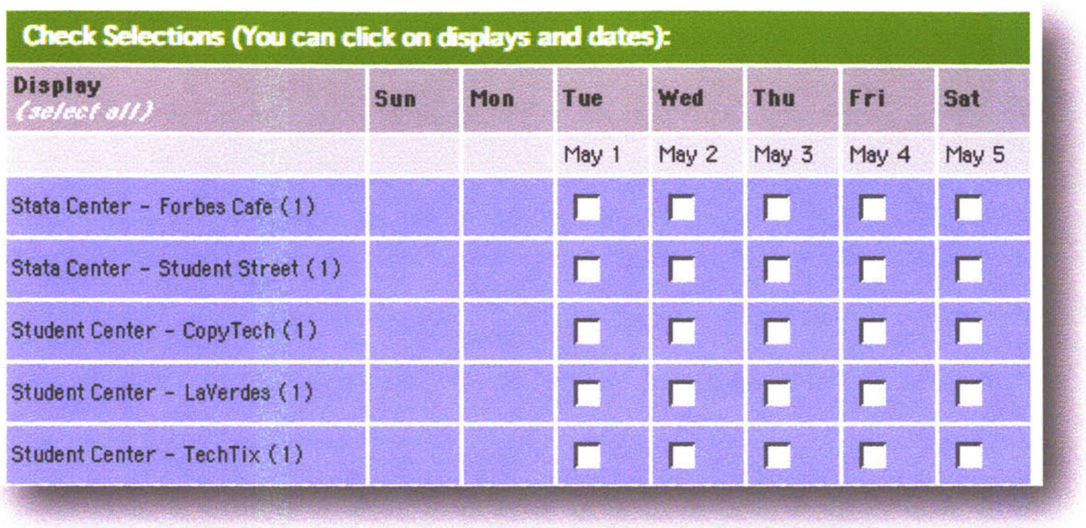
2. ***Select date range for which you would like to post***


Select Date Range for Displaying Post

Start Displaying Post on:	May	01	2007
Stop Displaying Post on:	May	25	2007

The input entered by the user during this step will be used to check to see which displays are available during the requested time. We restrict the user from checking more than three month blocks because this is a rather intensive database query. At this point in time, a user can request to post to a display any time in the future. This functionality may have to be adjusted so that groups don't reserve popular dates, years in advance.

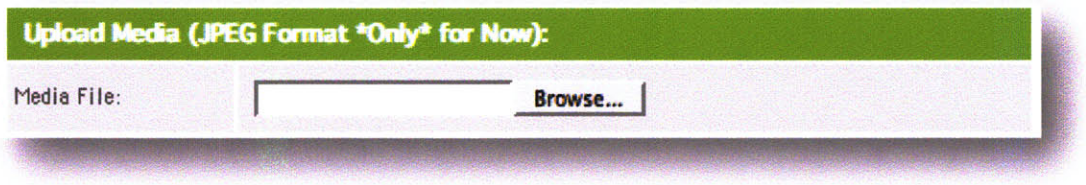
3. Select the displays on which you would like your post to appear


Check Selections (You can click on displays and dates):

Display (select all)	Sun	Mon	Tue	Wed	Thu	Fri	Sat
			May 1	May 2	May 3	May 4	May 5
Stata Center - Forbes Cafe (1)			☐	☐	☐	☐	☐
Stata Center - Student Street (1)			☐	☐	☐	☐	☐
Student Center - CopyTech (1)			☐	☐	☐	☐	☐
Student Center - LaVerdes (1)			☐	☐	☐	☐	☐
Student Center - TechTix (1)			☐	☐	☐	☐	☐

The user is presented a grid in the format of a calendar where he/she can select the desired displays and dates. If a screen is unavailable on a certain day, the check box would be grayed out. To the right of the name of the display, the cost of the display is listed within parentheses. The system will not allow a user to make selections that exceed the amount of points his/her group has left in its account.

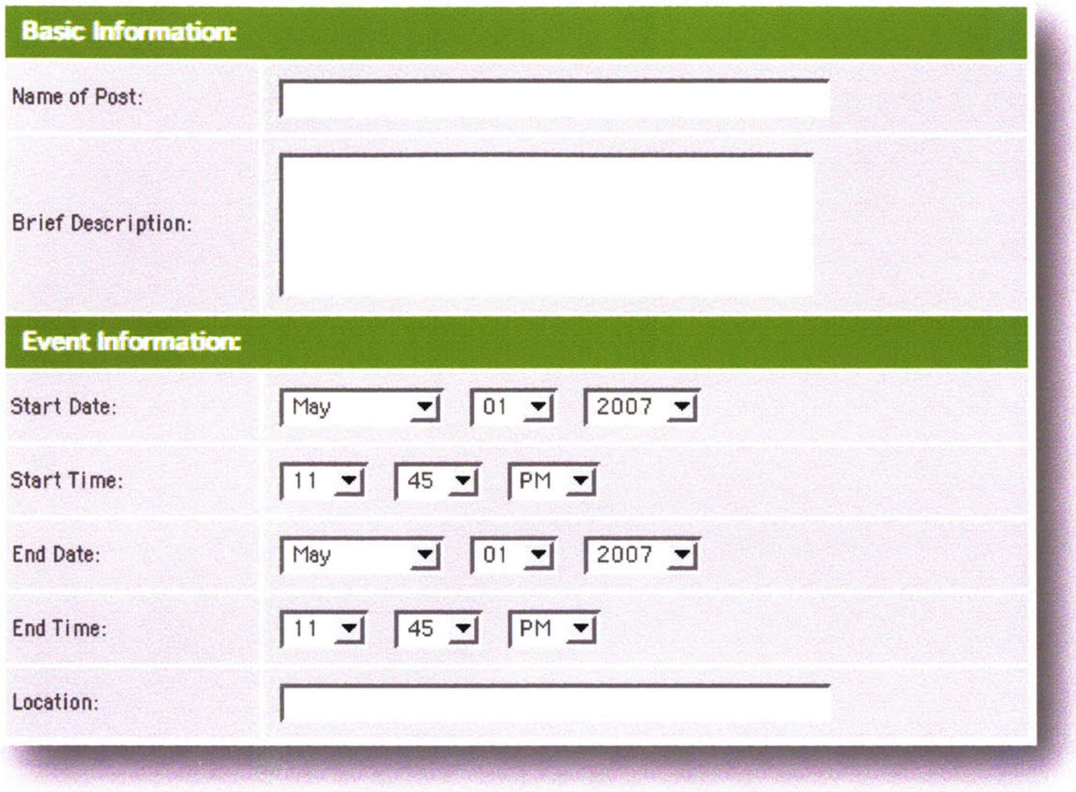
4. Upload media


Upload Media (JPEG Format *Only* for Now):

Media File:

The user uploads a jpeg, which will in turn be displayed on the display or displays selected in the previous step. The user is also able to choose whether the media should be scaled or stretched to match the resolution of the display(s).

5. Enter information about post



Basic Information:

Name of Post:	
Brief Description:	

Event Information:

Start Date:	May 01 2007
Start Time:	11 45 PM
End Date:	May 01 2007
End Time:	11 45 PM
Location:	

The user can enter additional metadata about the post, such as a name and a description. Since most posts are usually advertising some type of event, we also ask the user for information concerning the event, such as start time, end time, and location.

6. Confirm Post Before Submission

This is the last step in the posting process. It is simply one last opportunity for the user to check the information before completing the post submission.

3.3.1.3 Post Examples

There are a number of trends in posts submitted to the system. The most popular type of submission is that of advertising events. Users often include the usual information on their posted image, such as time and location of event, and some text describing the event. Many users also include a URL to find out more information and a logo representing their group. Some also give time a clearer context, such as adding the text "Tomorrow!" or "Today!" to the image, because they can specify when certain images go up on each display. Of course, all of this information must be 'remembered' by a passerby without the PhoneView implementation.

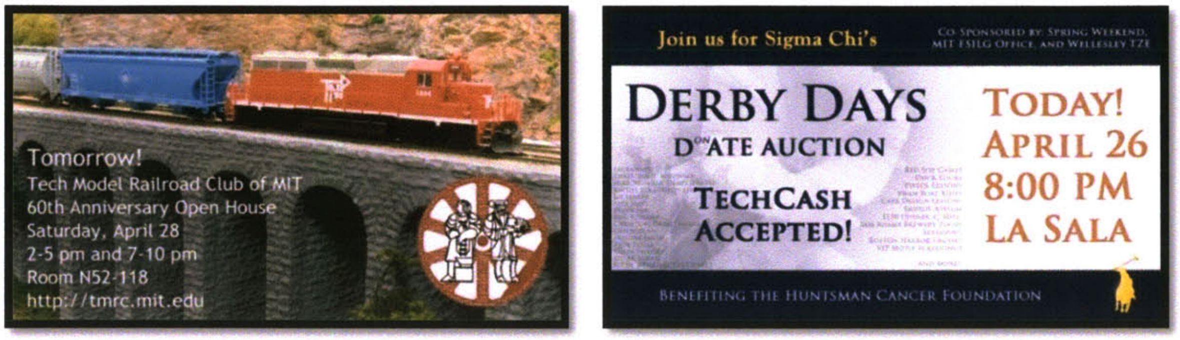

Tomorrow!
Tech Model Railroad Club of MIT
60th Anniversary Open House
Saturday, April 28
2-5 pm and 7-10 pm
Room N52-118
http://tmrc.mit.edu
Join us for Sigma Chi's
Co-sponsored by: Spring Weekend, MIT FSILG Office, and Wellesley TZE
DERBY DAYS
DATE AUCTION
TECHCASH ACCEPTED!
TODAY!
APRIL 26
8:00 PM
LA SALA
Benefiting the Huntsman Cancer Foundation

Figure 9: Event Post Examples

Another popular type of post is the notification of deadlines for applications and such. These almost always include a URL where one can find the application.

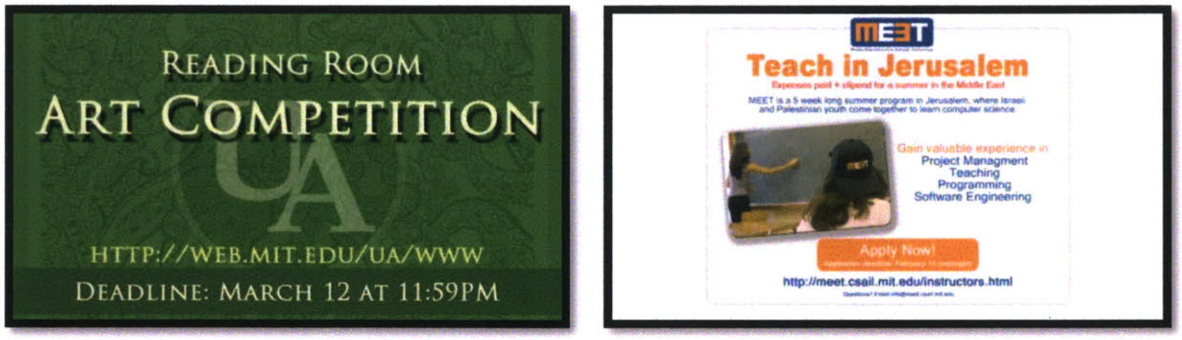

READING ROOM
ART COMPETITION
UA
HTTP://WEB.MIT.EDU/UA/WWW
DEADLINE: MARCH 12 AT 11:59PM
MEET
Teach in Jerusalem
Expenses paid + stipend for a summer in the Middle East
MEET is a 5 week long summer program in Jerusalem, where Israeli and Palestinian youth come together to learn computer science
Gain valuable experience in:
Project Managment
Teaching
Programming
Software Engineering
Apply Now!
http://meet.csail.mit.edu/instructors.html

Figure 10: Deadline Post Examples

3.3.1.4 *Usage Feedback*

From March 19th 2007 through May 19th 2007, the official public launch of DomeView to student groups, there have been over 120 unique posts submitted to the DomeView system from over 90 groups.

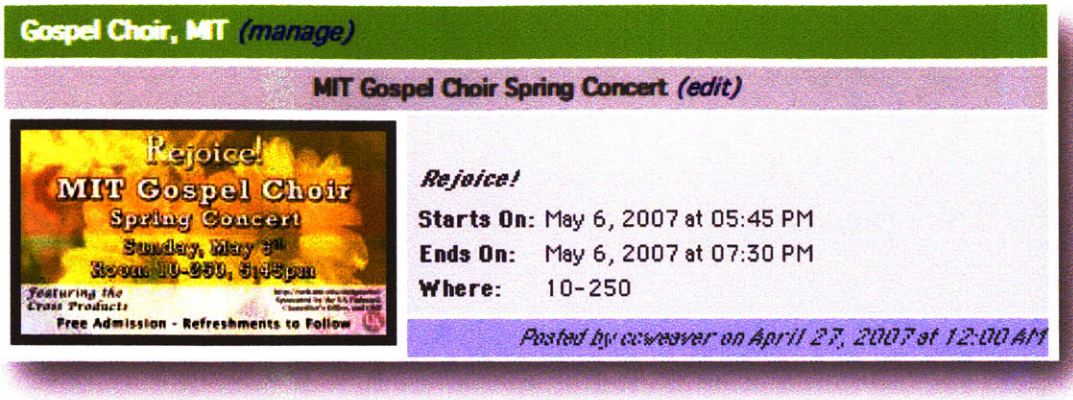
Based on informal feedback from users, almost all users find the posting submission process "user-friendly" and "straight-forward." Overall, users have had a very positive experience. Here is some email feedback we received from users:

> *"Thanks for putting this together! It is awesome! It makes advertising for events soo much easier! We used it a few weeks ago to promote an event, and we had about double the attendance that we have ever had before! It was great! I also like how it doesn't look cluttered like the bulletin boards, and each group gets equal space that they can decide how to use. This far superior to the bulletin boards in the infinite where a group with many members and gynormous posters can repost daily and monopolize the public advertising space. People also look at it more because it doesn't look like trash. Each announcement gets equal time. It's great!"*
>
> *"Domeview is awesome. What a great idea! the website is very nice and easy to use!"*
>
> *"Just a note to say how great this service is!"*

The most popular request among users is the ability to submit more complex media, such as animations and movies. This is definitely a feature we hope to incorporate, but wanted to solidify the underlying functionality before we began supporting additional formats.

3.3.2 Consumers

Consumers consist of all the members of the MIT community and visitors who want to learn more about the opportunities available on campus. The primary mode of consumption of the ads posted to DomeView is by viewing the displays. However, at the moment there is no way of interacting with the screen, so it is a purely passive medium. The consumers must remember what they see on the screen, write down the information, or visit the DomeView website to find out more information about a specific post. A user can view a post at the DomeView website (Figure 11), but it is not searchable and may be difficult for a consumer to reach. Also, the consumer may not even be aware that this information is available on the DomeView website.


Gospel Choir, MIT *(manage)*

MIT Gospel Choir Spring Concert *(edit)*

Rejoice!

Starts On: May 6, 2007 at 05:45 PM
Ends On: May 6, 2007 at 07:30 PM
Where: 10-250

Posted by ccweaver on April 27, 2007 at 12:00 AM

Figure 11: View of Post from Website

Although consumers do seem to prefer viewing the media on the displays versus the poster boards, the displays rotate the posts every 30 seconds, so if a user misses a post that he/she is interested in, he/she has to wait for the display to go through the entire rotation before the post comes up again. This lag time is not a problem with paper flyers. Unfortunately there is currently no method of interaction whereby the user can shuffle the posts in the queue for a specific display.

3.4 DomeView Architecture

The DomeView system employs a basic client-server architecture, where each display is accompanied by a PC, connected to the MIT network via Ethernet. A single server, located in the MIT server collocation farm, provides services to the displays, as well as users who post media to the displays. The displays essentially pull data from the server at certain intervals, to discover what media should be displayed at any given time. Additionally, the display code has been designed to withstand network outages by displaying information from the last connection to the server that is relevant to the actual time period.

3.4.1 DomeView Server

A single server provides web and database services, as well as storage for the media provided by users. The server runs a Ubuntu Linux distribution (Edgy Eft), Apache Web Server (2.0), PostgreSQL (8.1), and PHP (5.1). All of the data is exposed to the users as well as the displays through the web services, allowing the manipulation and viewing of the data stored in the database.

3.4.1.1 *AUTHENTICATION*

We authenticate the users of DomeView through MIT certificates. Each member of the MIT community is issued a new certificate every year, and thus it is a very reliable and secure method for authentication. As DomeView is a community based system, we leverage the community infrastructure for authentication rather than creating our own process of managing user and group accounts. Our web services run over SSL, and most of the site is not viewable by those outside the MIT community. We hope to open up non-private portions of the site to the external community in the near future.

The displays are identified by the static IPs assigned to the PCs associated with them. Although this is not secure, the information being provided to the displays is meant for public consumption and thus we are not really concerned with the spoofing of the IPs. However, it would be trivial to issue unique certificates to each display if we wanted to transmit data to the displays in a secure fashion.

3.4.1.2 *DATABASE STRUCTURE*

There are four main entities to the DomeView system: Users, Groups, Displays, and Posts. Users can be members of multiple groups, and a group is made up of multiple members, indicating a many-to-many relationship (Figure 12).

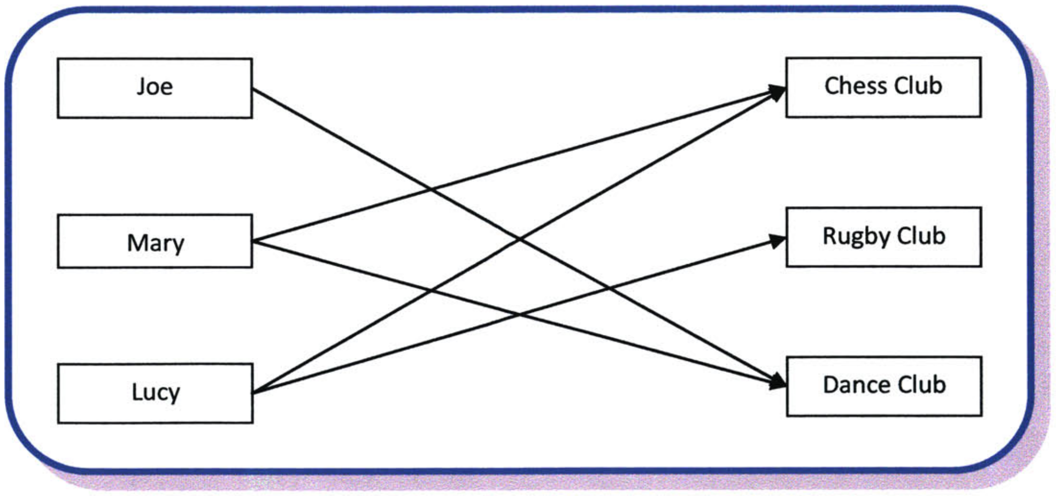


Figure 12: User and Group Relationship

The relationship between users and groups is a little bit more complicated than that, however. Within each group, a user could have one or more levels of membership, which grant the user additional permissions within that group. Some of these levels of membership include the ability to post on behalf of a group and the ability to add/remove members to/from a group. This membership level is an additional field included in the many-to-many table defining the relationship between users and groups.

Additionally, groups have global group levels, which indicate that the members of the group are able to commit certain actions on behalf of all the other groups, system-wide. For example, if a group has a group level that says it can post on behalf of all other groups, every member of that group can then submit a post on behalf of all other groups on the DomeView system. There is an analogous group level for each membership level.

Each individual post is submitted on behalf of a single group, and a group can submit many posts over time. Therefore there is a one-to-many relationship between a group and a post (Figure 13).

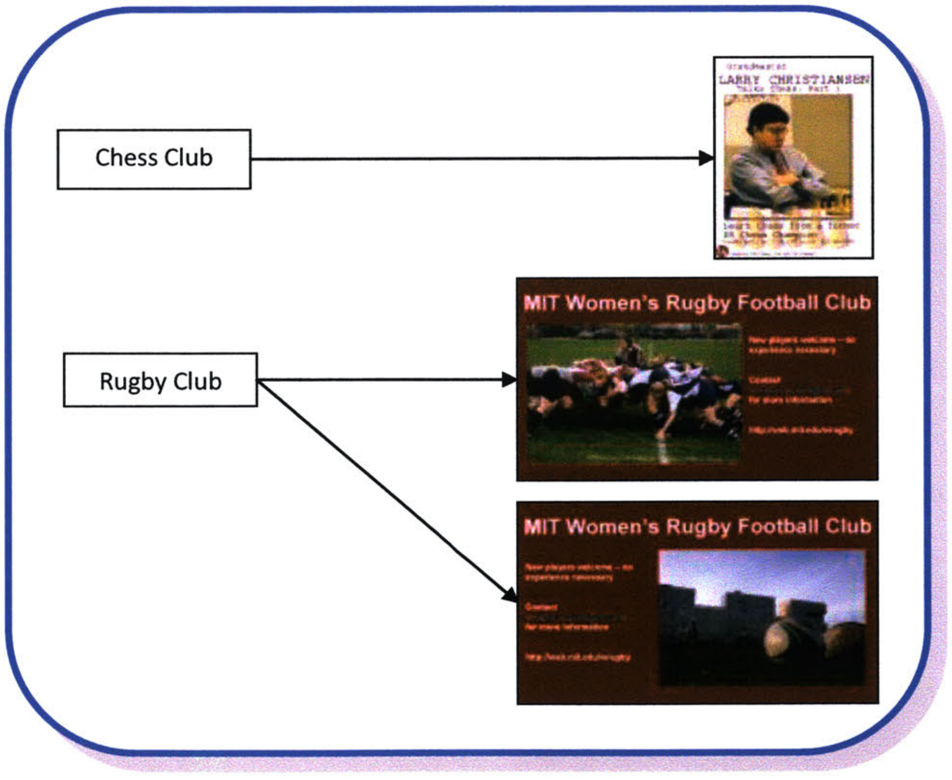

Chess Club
Rugby Club
LARRY CHRISTIANSEN
MIT Women's Rugby Football Club
MIT Women's Rugby Football Club

Figure 13: Group and Post Relationship

There is also a one-to-many relationship between users and posts, but the group-post relationship is significantly more important because membership of groups can change, and when a user posts, he/she is always doing so on behalf of a group.

Each display can have multiple posts submitted to it, and a single post can be posted on multiple displays. Therefore there exists a many to many relationship between posts and displays (Figure 14).

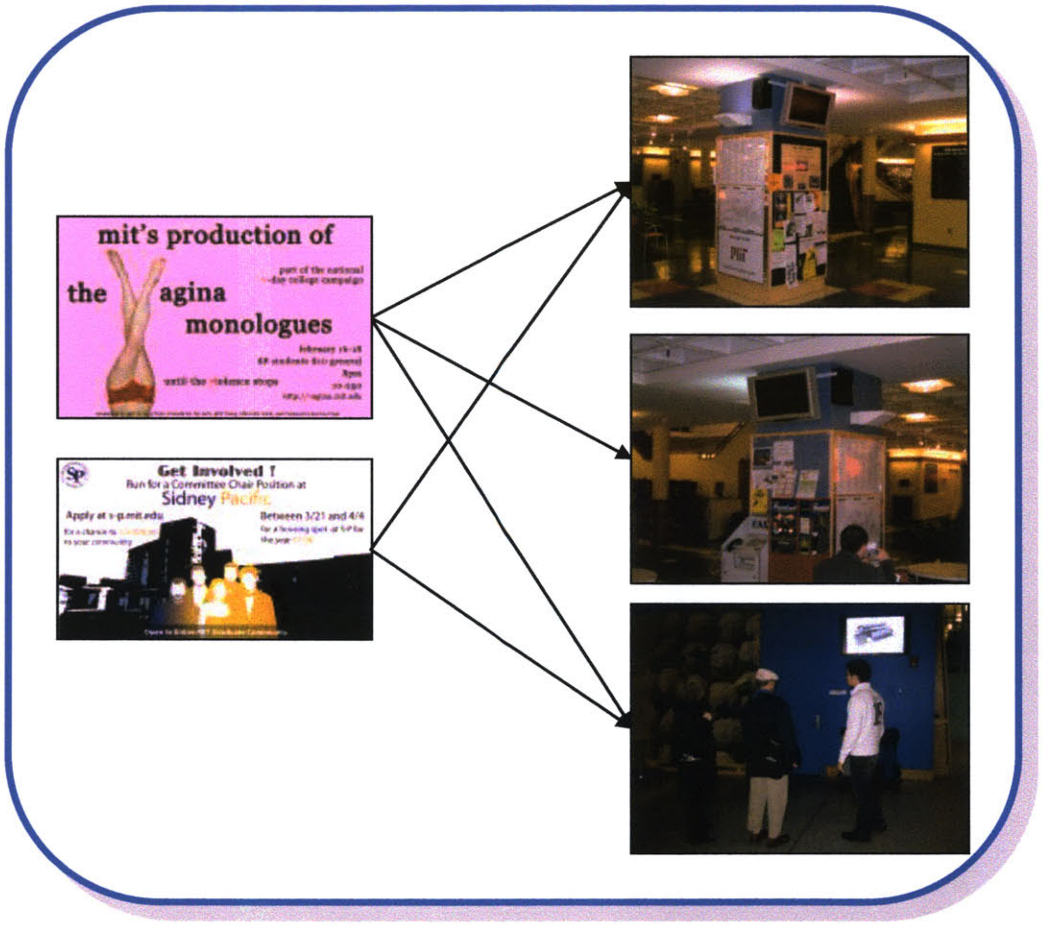


Figure 14: Post and Display Relationship

The post-display many-to-many relationship table also contains two fields that specify when the post should start and stop displaying on the specific display.

There are many other relationships represented in the database structure, but we have chosen to focus on the most important ones.

3.4.2 DomeView Display

As mentioned previously, each display is attached to a PC that has an Ethernet connection to the MIT network. Ethernet was chosen over wireless due to higher reliability as well as increased bandwidth, which will become important once we begin to support more complex media.

3.4.2.1 Web-based Architecture

Our goal with any of the code running on the display was to make it multi-platform, thus making the deployment of additional DomeView displays around MIT extremely easy. We wanted the minimum requirements for a display to be almost any PC and display with an Ethernet connection. Although we first thought about writing all of the display code from scratch using some multi-platform development language, we ultimately decided to not reinvent the wheel, and simply have our code run within a browser. A browser, in fact, seemed well-suited for a public kiosk type deployment.

Currently, we have three displays that are backed by a Boldata Mini PC running Ubuntu, and two displays that are backed by Mac Minis running OS X. All the machines are configured to boot up and load a browser (Firefox and Safari, respectively) in full-screen mode, where the URL is pointed to a specific location on our server. The page which is loaded has extensive JavaScript that communicates with the server via asynchronous calls and flips through the posts that are to be displayed, as indicated by the server. This deployment illustrates that any machine that is set up to load a browser in full-screen, can in fact act as a DomeView display.

The JavaScript code asks the server every couple of seconds if there is anything new to display, and therefore we can update the queue of posts for the display almost immediately. Additionally, we are able to detect when a display goes offline, indicated a network outage or possible vandalism.

3.4.2.2 Display Life and Energy Considerations

Since very few people pass the displays late at night, we decided to turn the displays off at those times to conserve energy and display life. Currently, we have the displays configured to turn off at 2 am and turn back on at 8 am. However, the machines associated with the displays do not turn off, so that we are able to monitor the display status during these times. We are, however, looking at more efficient ways of managing the turning on and off of displays, by possibly deploying motion sensors that turn the displays on only when there are passersby present, and turning them off when there has been extended inactivity around the displays.

3.4.2.3 Physical Security

Three of our displays are mounted in a building that is open to the public 24/7, and the two others are mounted in a building that is open to the public during the day, but employ MIT card access during the night. In both cases it is important to secure the displays as well as the computers attached to them so that they are not stolen or vandalized.

All of our displays are mounted high up to discourage curious passersby from touching the displays, and are secured with locked brackets. Our PCs in the Student Center are secured within lock boxes mounted along-side the display. Although this prevents access to the PC, we initially had some overheating problems because the boxes weren't well ventilated. Our PCs in the Stata Center are simply mounted on the back of the display, and are locked to the display bracket via a standard laptop lock. Although this prevents one from stealing the PC and is more aesthetically pleasing as compared to our other deployment, someone could easily press buttons on the PC to turn it off. This hasn't happened as of yet, and one could take measures to disable the functionality of external buttons on the pc. However, it is difficult to prevent someone from pulling out the power plug.

4 PhoneView: Mobile Phone Integration

Although DomeView, without interaction at the displays, serves a useful purpose to the community, one can imagine that creating a mechanism for interaction at the displays would enhance the benefits and user experience of a digital bulletin board system many times over. The most prevalent form of interaction among public kiosks to date seems to be displays equipped with touch-screens. Although useful in certain applications, there are a number of drawbacks.

4.1 Interaction Drawbacks of Touch-Screen

- ***Single-User Restriction.*** With a touch screen interface, only one user is able to interact with a kiosk at a time.
- ***Viewing Obstruction.*** When a user is interacting with a touch-screen, the user usually places him/herself directly in front of the screen, thus obstructing the view of other passersby.
- ***Display Location Restriction.*** Since the user must 'touch' the touch-screen, the display must be placed at a height and a location that is physically accessible to passersby. Not only might this be inconvenient to the display administrator, but the display becomes more vulnerable to vandalism. The administrator must go to great lengths to ensure the display's physical security, which would be less of a concern if the display could be mounted at a significant height, or behind a display case.
- ***Hygiene.*** Since the touch-screen is touched by a multitude of users that may or may not wash their hands often, the opportunity for the transfer of bacteria and viruses is prevalent.
- ***Expense.*** Touch-screen displays are as a rule more expensive than displays that do not have such interfaces. Additionally, for large-format displays one has to purchase a touch-screen overlay, which has to be configured separately.
- ***Restricted Data Transfer.*** The only way to transmit data to the display is through the touch-screen. Additionally, the only way for the display to transfer information back to the user is by displaying it on the screen, or possibly sending data to the user by email or other such internet-based communication protocol.

Other mechanisms for creating interactions at kiosks have been explored, but aren't yet in wide use. These mechanisms include voice, gesture, writing, and face recognition. Even if such mechanisms were perfected in their own right, there applications to public kiosks have similar drawbacks to touch-screens as outlined above.

With the proliferation of mobile phones, a method for interaction that doesn't have most of the drawbacks attributed to other methods, in addition to providing opportunities for interaction that have yet be explored, is to enable kiosks to interact with mobile phones via Bluetooth.

4.2 Purpose & Principles

The current mobile phone market is estimated at selling 1 billion units per year(7), and most new phones come equipped with Bluetooth technology, whose main use is with handsets and hands-free devices, such as those built into many new automotive vehicles. Additionally, according to a recent study by Millard Brown, 81 percent of consumers have heard of Bluetooth technology. (8) Thus, Bluetooth-enabled mobile phones seem like a ripe venue of interaction for public displays.

Additionally, mobile phones have evolved from simple devices that were only used for telephony purposes, to full-fledged personal data assistants, where users store their contacts, to-do lists, as well as daily schedule. Mobile phones have also found alternate use as media devices that could enable a user to view/hear as well as capture photos, videos, and audio. If a public kiosk can tap into those data stores, it could significantly enhance the user experience.

There are a number of principles that we wanted to adhere to when trying to create an optimal and enhanced interaction with a kiosk and Bluetooth-enabled mobile phone. It is also essential that these principles fall in line with the original principles for designing DomeView, as described in Section 3.2.

4.2.1 PhoneView Interaction Principles

- ***No Software Installation.*** We do not want to require users to install software on their phones in order to interact with the displays. Not only is there a very small chance that one could convince many users to install custom software on their phones due to technical ignorance or impatience, but it would require us to create custom software for many different platforms which are constantly being updated and changed.

- ***Quick and Intuitive Initialization.*** The initialization steps that must be taken by the user in order to begin the interaction with our displays must be quick and should be intuitive to the user.
- ***Intuitive Interface.*** Whatever the user interface is, it must be intuitive to the user, or at least should be easy to learn.
- ***Ability to Transfer Data Back and Forth.*** If the mobile phone supports it, the ability to transfer data back and forth must be available.
- ***Multi-User interaction.*** The interaction mechanism should allow for multiple users to interact with the screen at the same time, and possibly even facilitate interaction between users.
- ***Unique Identification.*** There should be a way that a user can be uniquely identified every time he/she interacts with a kiosk. Thanks to the unique IDs associated with every Bluetooth device, this is possible.
- ***Responsiveness.*** There shouldn't be any perceived delay in responsiveness when the user is interacting with a display.
- ***Distance Interaction.*** The interaction with the display must be able to occur at a distance of at least 10 feet.
- ***Noninvasive Interaction.*** We would not want the display to initiate communications with all discoverable Bluetooth mobile phones that pass within range, but rather have the user initiate the interaction on his/her side.
- ***Support for Most Bluetooth-Enabled Phones.*** Whatever the interaction is, it should be supported on the majority of Bluetooth-enabled mobile phones, not just smart phones. However, the interaction should be able to scale up to take advantage of advanced features.

4.3 Initial Ideas

In this section we will discuss some of the initial ideas that ultimately did not measure up to most of our principles, and afterwards we present a solution that does so quite elegantly.

4.3.1 Custom Software for Each Mobile Phone

The idea here was to create very simplistic piece of software for each mobile phone which basically mapped key presses to commands that would be sent over Bluetooth RFCOMM, which is essentially serial port emulation, to the display. The key presses would include all those located on the number pad. In the simplest case, this would allow the user to use the key pad as a directional pad, as illustrated in Figure 15.This would leave us with the ability to provide additional mappings for keys 'Five', 'Zero', 'Pound', and 'Star'. Possibilities include a 'Select' command, as well as a 'Back' and 'Forward' command to and from changing contexts, as well as a 'disconnect' command.

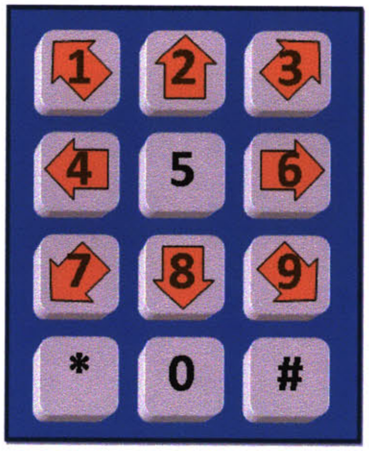


Figure 15: Number Pad Mapping

Although programming software for a number of mobile phone platforms that simply maps key presses to RFCOMM commands would be relatively easy, but somewhat tedious, our bigger concern was trying to get individuals to install the software on their phones. The chances that a user would go to a website to download the software and then have the cables, software, and know-how to transfer our application to the phone is highly unlikely.

One idea to address this would be to have an additional touch-screen by each kiosk where a user could setup the transfer of the appropriate software to their phone, by inputting the type of phone they have. It would also require that the user set their cell phone as discoverable so that the display would know which phone to send the software to. However, there are many phones out there that do not support Object Exchange (OBEX) and the Object Push Profile (OPP), which are generally only supported by smart phones and those with PDA capabilities. This method would take too many steps, and would also not be able to handle less advanced Bluetooth-enabled phones. Additionally, the installation of another screen by each kiosk seems a bit extravagant.

4.3.2 vCard Transmission

This idea was even less flexible than the one listed in the previous section, but would be fairly easy to implement. Basically, a user would transmit vCards, a standardized format for contacts, to the display that had data encoded for certain fields, thus causing the display to respond in various ways. For example, if the user wanted to receive an event representing one of the posts on the screen, he/she could transmit a vCard where the 'Last Name' field was the command, such as 'Event,' and the 'First Name' field would be the number or string representing the post the user is interested in.

There are a number of problems with this solution, first of which is that many phones don't support the transmission and/or creation of vCards. Second, there would have to be an extensive vocabulary which the user would have to learn. Third, it takes quite a bit of time to construct a vCard and then to submit it, especially if the mobile phone does not have a keyboard. However, features that do incorporate the transmission of Vcards might prove useful, but not as the main mode of interaction on a display.

4.4 DTMF Detection Over Bluetooth

Although Bluetooth phones vary in the profiles they support, most, if not all of them, support the headset and hands-free profiles. The headset profile enables a user to make calls through their mobile phone using a wireless headset, such as the Plantronics Voyager (Figure 16).

Figure 16: Bluetooth Headset

The hands-free profile is similar to the headset profile, but provides for some additional functionality. It is mainly utilized in automotive vehicles, so that the user can pipe the audio from his/her mobile phone through the vehicle's stereo system, or additional equipment installed within the vehicle, such as the Parrot EasyDrive (Figure 17).

Figure 17: Bluetooth Hands-Free Car Kit

Due to the wide-ranging support of the headset and hands-free profiles, if we could create an interaction mechanism that works through such profiles, then we could enable virtually all Bluetooth-enabled phones to interact with kiosks, among other devices, without installing any software. Additionally, many users are already familiar with establishing a connection between their phones and Bluetooth headsets, thus if the mechanism was the same for interfacing with our kiosk, most users wouldn't need to learn anything new.

The steps to initiating the interaction would occur like so:

1. The kiosk's Bluetooth adapter is set as discoverable, advertising the hands-free profile
2. The user comes within range of the kiosk and takes out mobile phone
3. The user begins the pairing sequence with the kiosk, as if it were a hands-free device
4. Once pairing has been established, the kiosk initiates the audio connection with the user's mobile device
5. From this point on, all audio from the mobile phone is sent to the display over Bluetooth

At this point, the kiosk will begin to process all the audio coming from the mobile phone, specifically the tones that are generated when the user hits keys on the number pad. Since each key generates a unique combination of frequencies, the display can tell which key the user is pressing, details of which are discussed further in Section 5.3.

If the kiosk can detect what key the user is pressing on the mobile phone, than we can map each key on the number pad to perform any process we desire on the kiosk. We could interpret the keys as a directional pad as described in Section 4.3.1, and even accept text input as one would enter text in a text message. This mechanism of interaction, combined with our ability to detect what other features are supported by the mobile devices, enables us to create an enhanced multi-user interaction that isn't possible with previous mechanisms. These possibilities are further explored in Section 5.3.3 and Chapter 7.

5 PhoneView Implementation

The basis for our solution, PhoneView, was developed on the Apple OS X platform, primarily due to the fact that two of the DomeView displays are backed by Mac Minis which were already equipped with built-in Bluetooth adapters. The implementation involved delving into the OS X Bluetooth Framework (9), as well as OS X's Audio Framework, also known as CoreAudio (10). The languages involved included Objective-C, C, and C++, as well as the scripting language AppleScript. Additionally, the Bluetooth specification was needed in order to learn how to advertise certain profiles as well as to establish connections between the mobile phone and the kiosk.

We investigated the possibility of implementing PhoneView in Linux via the BlueZ Bluetooth stack. Since the stack provides access to the SCO Audio Stream, it does indeed seem feasible. Additionally, Linux provides the developer access to the lower level HCI Layer, which isn't the case in OS X. This is definitely an item that warrants further exploration.

5.1 Implementation Overview

5.1.1 Search and Advertise

The first part of the interaction that must take place is the advertising of the Hands-Free service by the kiosk. The kiosk would be continuously advertising the service so that passing users can detect the service with their Bluetooth-enabled mobile phone.

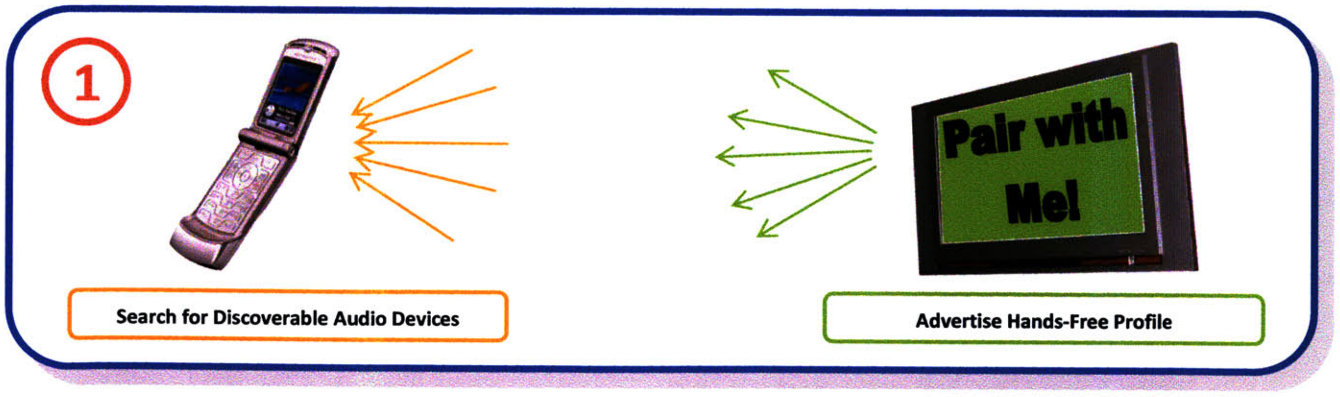

1
Pair with Me!
Search for Discoverable Audio Devices
Advertise Hands-Free Profile

Figure 18: Search and Advertise

5.1.2 Device Pairing

Once the kiosk shows up in the search done by the user via his/her mobile phone, the user has to initiate a Bluetooth connection with the kiosk. The user would pair with the kiosk as if it were an actual Hands-Free device, and thus would be asked to transmit a Pin for the Bluetooth connection. The kiosk would detect the initiation of this request, and would display the Pin to be used on the screen. The user would then enter the Pin displayed on the screen and complete the pairing.

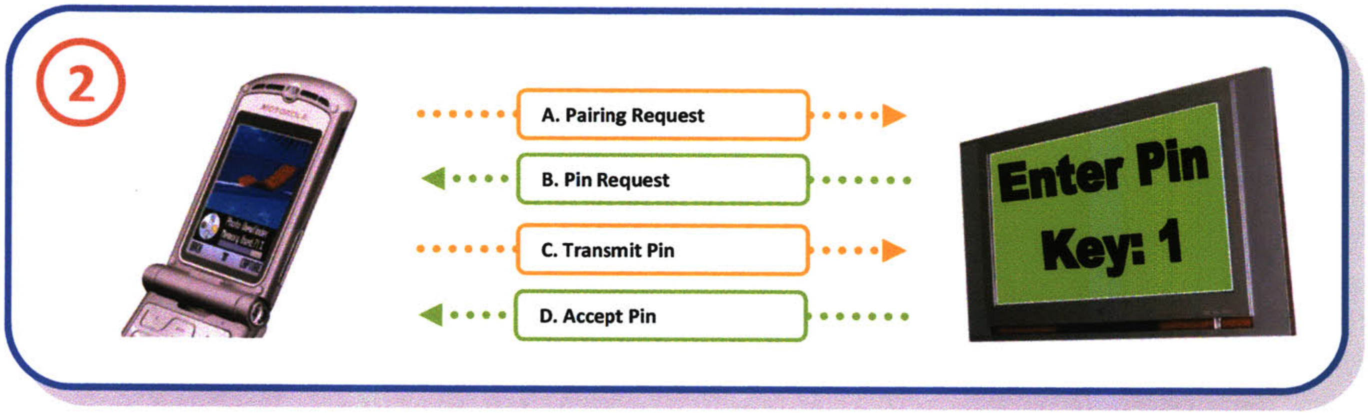


Figure 19: Device Pairing

5.1.3 Establishing Service Level Connection

A prerequisite for establishing an audio connection is the initiation is the establishing of an RFCOMM connection, over which a number of commands are transmitted. This connection remains alive while the audio connection is established.

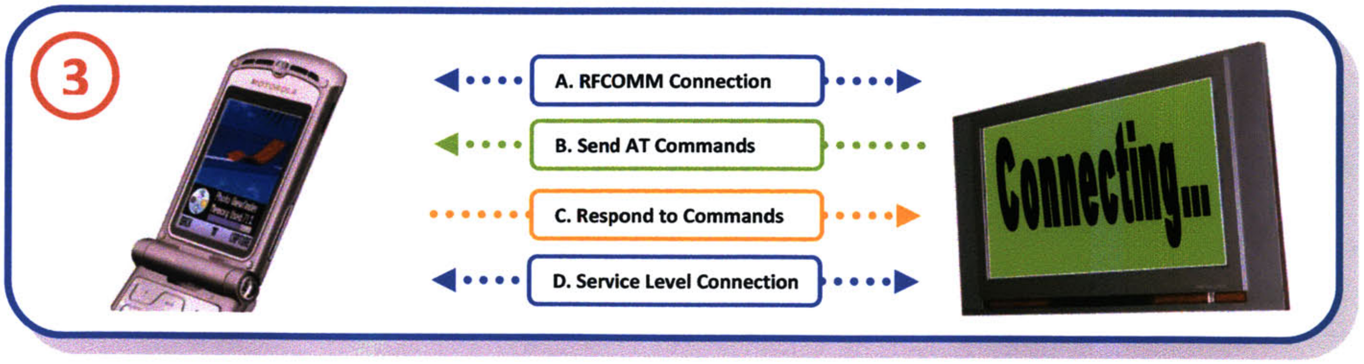


Figure 20: Establishing Service Level Connection

5.1.4 SCO Audio Connection

Finally, the audio connection is established between the mobile phone and the kiosk. At this point, all audio generated by the mobile phone is routed towards the kiosk. The kiosk then processes the audio so that it can detect which keys are being pressed on the mobile phone keypad.

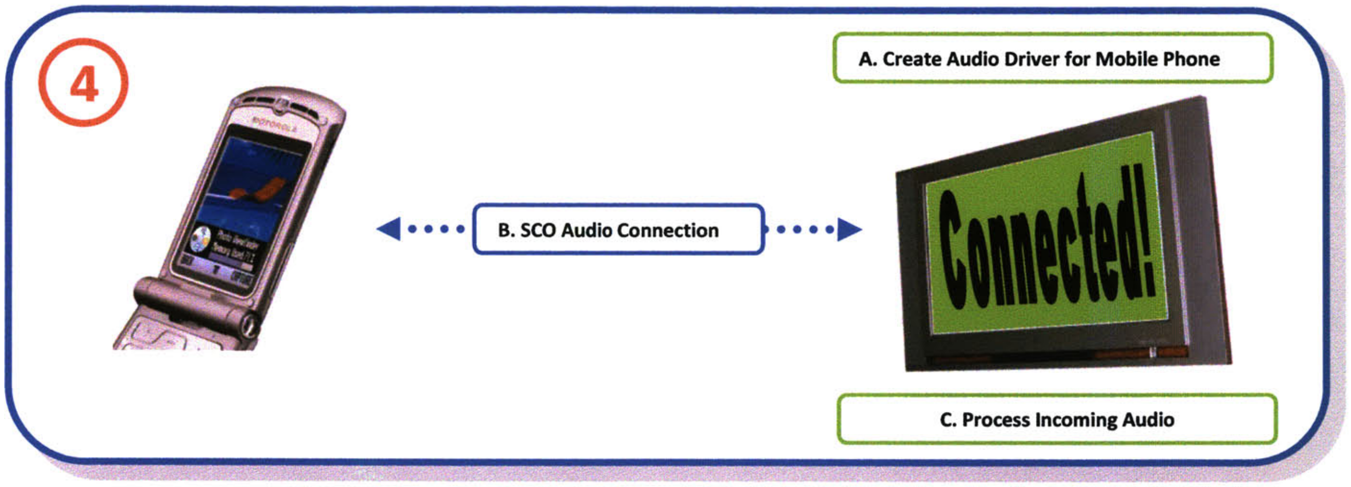

4
A. Create Audio Driver for Mobile Phone
B. SCO Audio Connection
Connected!
C. Process Incoming Audio

Figure 21: Establish SCO Audio Connection and Process Audio

5.2 Bluetooth

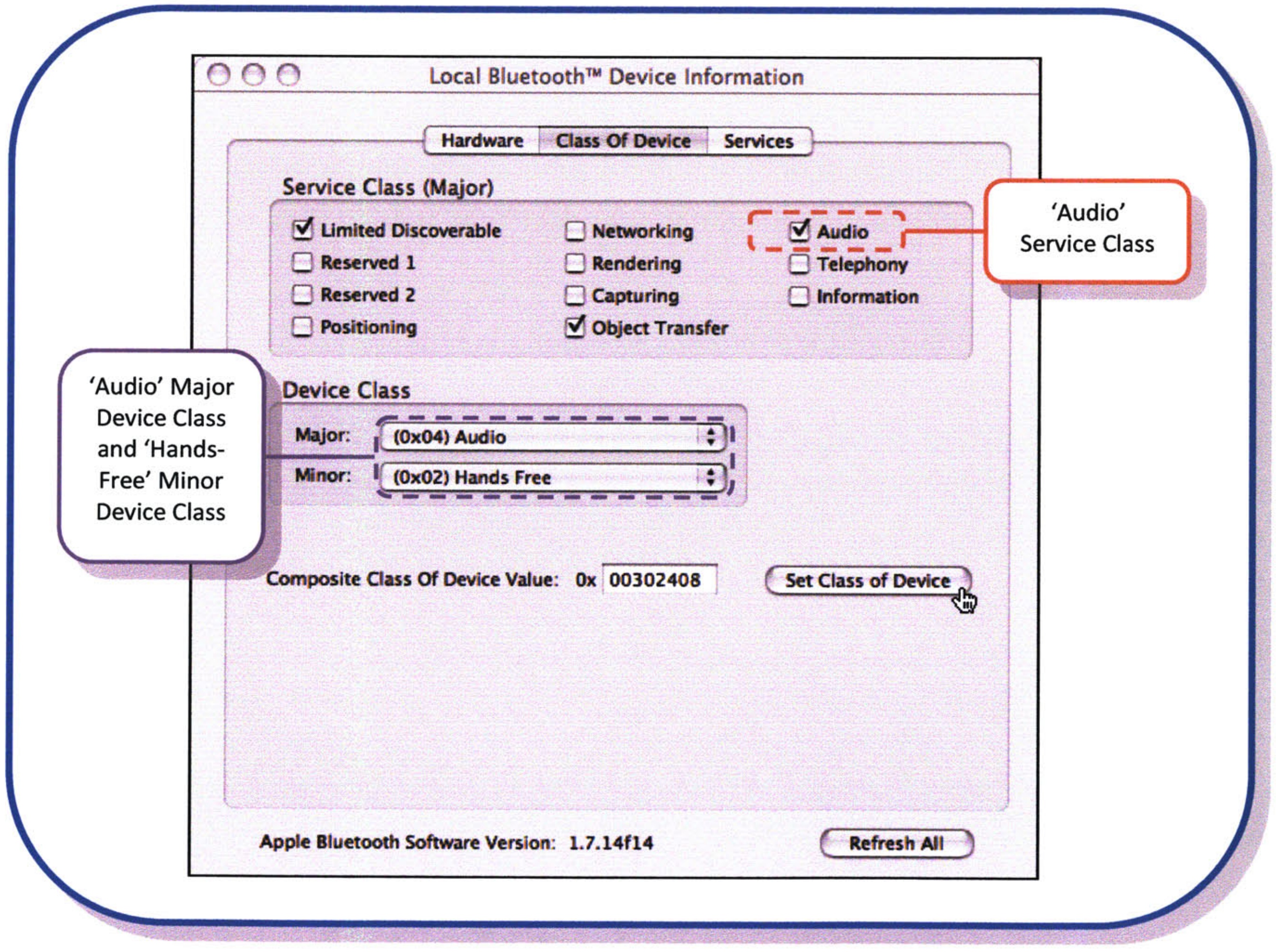
There were a number of steps that needed to be taken in order to get the audio flowing from the mobile phone to the computer, including setting up the proper profile on the computer, and establishing the appropriate connections with the mobile phone.

5.2.1 Advertising Hands-Free Profile

5.2.1.1 Service Class and Class of Device

Prior to any other operation, we need the computer to appear to be a Hands-Free Device, so that the mobile phone recognizes it and agrees to make the needed connections with it. Even prior to advertising the Hands-Free Profile, we need to set the **Service Class** and **Class of Device** field of the computer. If these fields are not set properly, some mobile phones will not agree to communicate with the computer. The 'Audio' bit of the **Service Class** field must be set, the **Class of Device** field must include 'Audio' as a **Major Device** class, and 'Hands-Free' as a **Minor Device** class. (11 p. 85)

Setting the **Service Class** and **Class of Device** fields is done via the Bluetooth HCI layer, and unfortunately the OS X APIs for accessing that layer are private. However, there is an application provided with the Bluetooth Developer SDK for OS X 10.4 called 'Bluetooth Explorer,' which allows one to modify HCI level settings via the GUI (located at `/Developer/Applications/Utilities/Bluetooth`). If you click on the 'Utilities' menu item, then 'Get Local Device Info,' and then click on the 'Class of Device' tab, it'll bring up the interface that allows you to change the **Service Class** and **Class of Device** fields (Figure 22).


Local Bluetooth™ Device Information
Hardware
Class Of Device
Services
'Audio' Service Class
'Audio' Major Device Class and 'Hands-Free' Minor Device Class
Set Class of Device
Refresh All

Service Class (Major)

- [✓] Limited Discoverable
- [ ] Reserved 1
- [ ] Reserved 2
- [ ] Positioning
- [ ] Networking
- [ ] Rendering
- [ ] Capturing
- [✓] Object Transfer
- [✓] Audio
- [ ] Telephony
- [ ] Information

Device Class

Major: (0x04) Audio

Minor: (0x02) Hands Free

Composite Class Of Device Value: 0x 00302408

Apple Bluetooth Software Version: 1.7.14f14

Figure 22: Changing Service Class and Class of Device in OS X

As mentioned previously, although changing these fields directly through the HCI layer is not possible, in our implementation we automated the process outlined in the previous paragraph through the use of AppleScript, a scripting language provided by Apple that allows users to automate interactions with GUI-based applications. (12) Once this step is completed, all mobile phones should be able to discover the kiosk as a Hands-Free device.

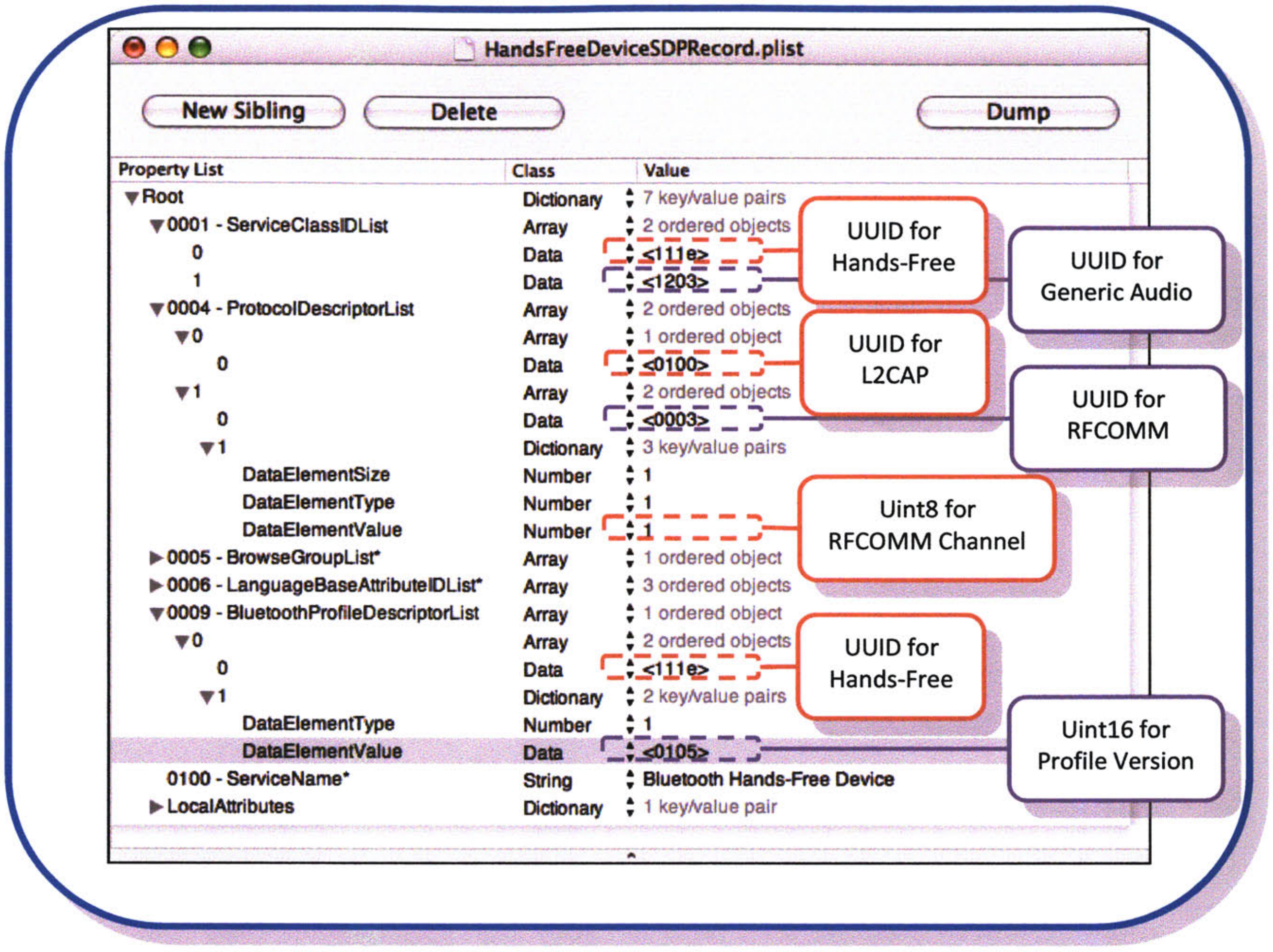
5.2.1.2 Adding Profile to SDP Database

Although the kiosk appears as a hands-free device to mobile phones after taking the steps outlined in Section 5.2.1.1, in order for the mobile phone and computer to begin communicating with each other, we need to add the Hands-Free Service Record to the Service Discovery Protocol (SDP) Database on the kiosk, where as the mobile phone has the Audio Gateway Service Record in its SDP Database. The service record is made up of key-value pairs that identify it with the appropriate profile, as defined by the Bluetooth specification, as well as specifying some specifics, such as which RFCOMM channel to communicate over and what additional features are supported by the device. The OS X Bluetooth framework allows users to build a property list using an application called 'Property List Editor,' and then load that property list into the SDP database. OS X 10.4 also ships with a property list for the Hands-Free Profile (`HandsFreeDeviceSDPRecord.plist`) located within the IOBluetooth Framework Bundle (Figure 23).

The following items need to be specified in the Hands-Free Service Record (11 pp. 81-82):

- ***ServiceClassIDList*** –The list of the service classes associated with the profile. Should match the service classes indicated in Section 5.2.1.1
  - ***ServiceClass0*** – The primary service class for the profile. Should be set to the UUID for Hands-Free, as specified by the Bluetooth Specification.
  - ***ServiceClass1*** – The secondary service class for the profile. Should be set to the UUID for Generic Audio, as specified by the Bluetooth Specification.
- ***ProtocolDescriptorList*** – A list of protocols upon which the hands-free profile operates.
  - ***Protocol0*** – The primary protocol for the hands-free profile. Should be set to the UUID for L2CAP, as specified by the Bluetooth Specification.

        - ***Protocol1*** – The secondary protocol for the hands-free profile. Should be set to the UUID for RFCOMM, as specified by the Bluetooth Specification.
            - ***ProtocolSpecificParameter0*** – This is an underlying parameter for the RFCOMM connection. In this case, it specifies the channel (as a Uint8) over which RFCOMM should communicate.
- ***BluetoothProfileDescriptorList*** – A list of additional details about the supported profile.
    - ***Profile0*** – The first profile supported by the device. Should be set to the UUID for Hands-Free, as specified by the Bluetooth Specification. Should match **ServiceClass0**.
        - ***Param0*** – A parameter for the supported profile. In this case, it specifies the version of the Hand-Free Profile supported. In our case, should be set to the Uint16 representing the latest version, which is 0x0105 (representing version 1.5).
- ***ServiceName*** – This is a string representing a displayable service name. This is an optional entry, and generally mobile devices do not display this when pairing, but rather the name of the Bluetooth Device.
- ***SupportedFeatures*** – This is a Uint16 representing additional features beyond the basics supported by the profile. In our case, it is set to 0x000, representing that we don't support any additional features when it comes to the Hands-Free Profile.



Property List	Class	Value
▼Root	Dictionary	7 key/value pairs
▼0001 - ServiceClassIDList	Array	2 ordered objects
0	Data	<111e>
1	Data	<1203>
▼0004 - ProtocolDescriptorList	Array	2 ordered objects
▼0	Array	1 ordered object
0	Data	<0100>
▼1	Array	2 ordered objects
0	Data	<0003>
▼1	Dictionary	3 key/value pairs
DataElementSize	Number	1
DataElementType	Number	1
DataElementValue	Number	1
▶0005 - BrowseGroupList*	Array	1 ordered object
▶0006 - LanguageBaseAttributeIDList*	Array	3 ordered objects
▼0009 - BluetoothProfileDescriptorList	Array	1 ordered object
▼0	Array	2 ordered objects
0	Data	<111e>
▼1	Dictionary	2 key/value pairs
DataElementType	Number	1
DataElementValue	Data	<0105>
0100 - ServiceName*	String	Bluetooth Hands-Free Device
▶LocalAttributes	Dictionary	1 key/value pair

Figure 23: Property List for Hands-Free Service Record

Once one has prepared the property list, it is rather easy to load the record into the SDP database using the C APIs for Bluetooth (9 pp. 44-45):

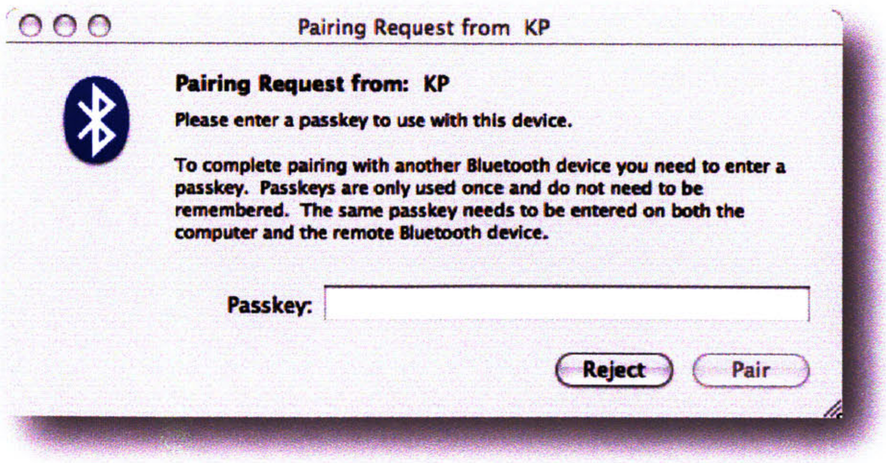
```
#import <IOBluetooth/IOBluetoothUserLib.h>

NSString                  *dictionaryPath = nil;
NSMutableDictionary       *sdpEntries = nil;

// Get the path for the dictionary we wish to publish, assuming the plist file is in
      the Resources folder of our project
dictionaryPath = [[NSBundle mainBundle] pathForResource:@"HandsFreeDeviceSDPRecord"
      ofType:@"plist"];
```

```
sdpEntries = [NSMutableDictionary dictionaryWithContentsOfFile:dictionaryPath];

IOBluetoothSDPServiceRecordRef serviceRecordRef;

// Add SDP dictionary
IOBluetoothAddServiceDict( (CFDictionaryRef) sdpEntries, &serviceRecordRef );
```

5.2.2 Device Pairing

Prior to communicating with another device, most mobile phones require that they be paired with another device using a numeric pin or key. Most Bluetooth headsets and hands-free devices come with a set key for pairing (the most common one being '1111'), while a few of these devices allow the owner to set the key manually.

To replicate the process that users go through when pairing with a headset, we wait for a mobile phone to initiate the pairing connection. The pairing mechanism occurs on the HCI Layer, so unfortunately we once again cannot directly manipulate it on OS X. Similarly to when we ran into such trouble in Section 5.2.1.1, we use AppleScript to circumvent this obstacle. In OS X, when a device requests to pair with the computer, a request is sent to the computer via Bluetooth, and a 'Pairing Request' dialog box pops up.


Pairing Request from KP

Pairing Request from: KP

Please enter a passkey to use with this device.

To complete pairing with another Bluetooth device you need to enter a passkey. Passkeys are only used once and do not need to be remembered. The same passkey needs to be entered on both the computer and the remote Bluetooth device.

Passkey:

Figure 24: Pairing Request Dialog Window

Generally, the user would enter in a numeric key into the 'Passkey' field and click 'pair.' The computer would then send a response to the mobile phone asking for a Passkey. Then the user would enter the same key he/she entered previously, and the phone would indicate to him/her that the devices have been paired.

Obviously, this process would not work in our case, because there is no external input to the kiosk besides the Bluetooth connection. Therefore, our application detects when the 'Pairing Request' dialog appears, and runs an AppleScript that enters a Passkey of '1' and clicks 'Pair.' Of course, we indicate to the user that the Passkey to be entered on his/her mobile phone is '1.' We could create a random passkey each time, thus improving security, but decided to simplify Passkey entry for the user.

It is important to note that most mobile phones will not pair with the computer unless the service record is being advertised correctly, as described in Section 5.2.1. Now we are ready to do the real work in establishing a SCO audio connection.

5.2.3 RFCOMM Connection

According to the Hands-Free Profile Specification, either the mobile phone (Audio Gateway) or the computer (Hands-Free Device) can be the one to initiate the RFCOMM link on the channel specified on the hands-free device's service record. The initialization commands are sent across the RFCOMM link, and the RFCOMM link has to remain active in order for the audio to continue to flow between the two devices.

Some mobile phones attempt to open up the RFCOMM link as soon as pairing is complete, while others wait for the hands-free device to initiate the connection. Therefore, our implementation waits a few seconds after pairing to see if the mobile phone opens the RFCOMM channel specified in the SDP. If the RFCOMM channel is not opened after time runs out, then our application opens an RFCOMM channel with the mobile phone.

When the RFCOMM channel is opened completely, initiated by the computer or the mobile phone, the computer begins to send AT commands across the link, as specified in the Hands-Free Profile

Specification. These set of commands help establish the 'Service Level Connection,' which is a prerequisite to establishing the SCO Audio Connection. Here is the communication required in order to establish a 'Service Level Connection':

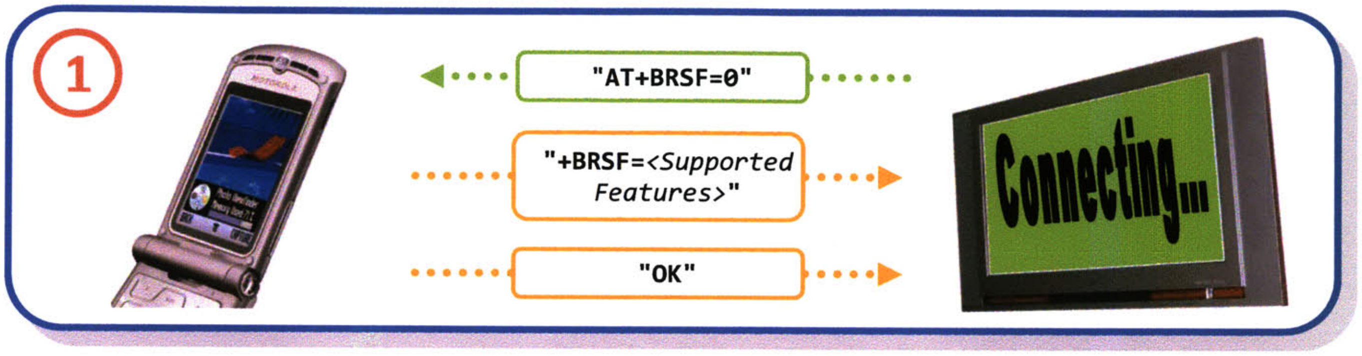


Figure 25: Supported Features Exchange

First, the kiosk sends the "AT+BRSF=" command to the mobile phone. Here, the kiosk tells the mobile phone what features of the profile it supports, and requests that the mobile phone returns with the features that it supports. By appending a "0" to the first command, we are telling the mobile phone that the kiosk does not support any additional features. These additional features could include '3-way calling' and voice recognition.

The mobile phone responds with the features that it supports that are relevant to hands-free device, and sends "OK" to indicate that the command was received and responded to properly. Since we aren't really interested in the additional hands-free features offered by the mobile phone, we don't do anything with the information besides acknowledging that the communication completed and that we can move on to the next command.

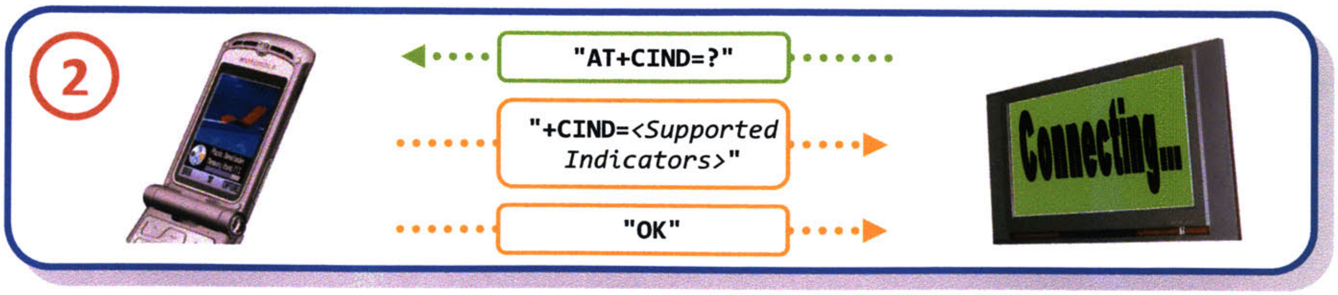


Figure 26: Supported Indicators

The kiosk sends the command "AT+CIND=?" to mobile phone, which asks the mobile phone which indicators it supports, such as signal indicator and battery charge indicator. Once again, we don't really care what the response is because we don't use the information, but simply want to know that the command was received properly.

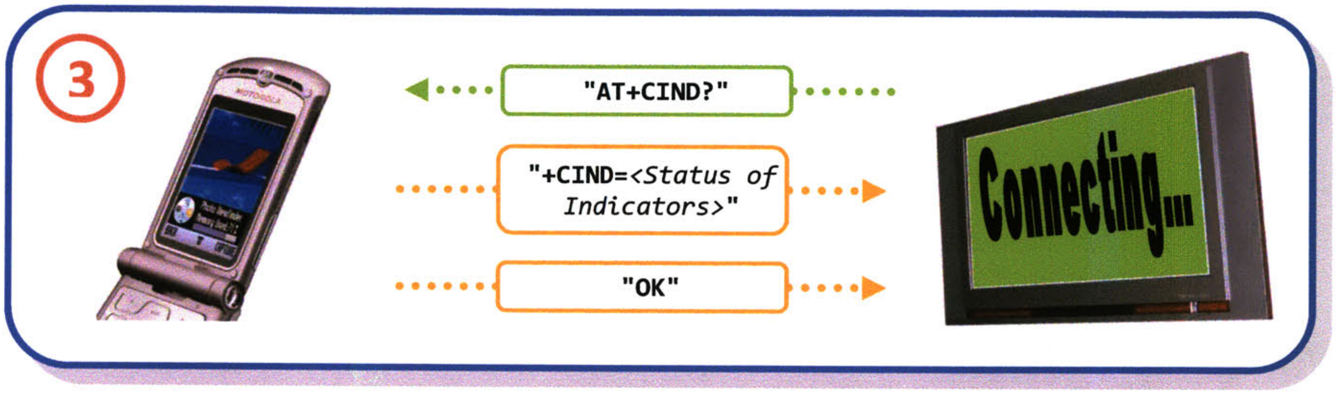


Figure 27: Get Current Status of Mobile Phone Indicators

The kiosk then sends the "AT+CIND?" command, which asks the mobile phone for the current status of its indicators. Once again, the kiosk waits to see that the mobile phone responds appropriately.

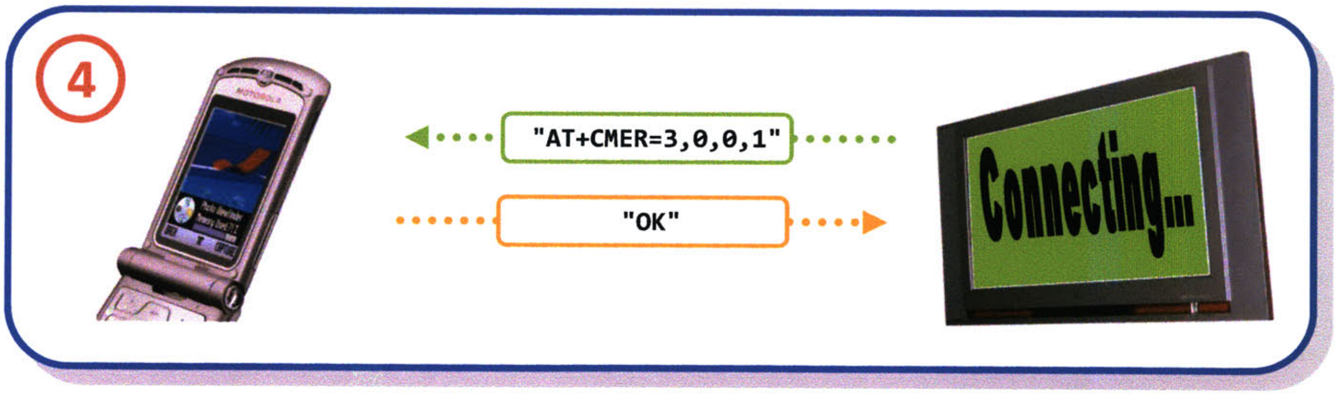


Figure 28: Activate Event Reporting for Status Indicators

The "AT+CMER=" command activates and deactivates event reporting, so that when indicator statuses change on the mobile phone, it will send a command to the hands-free device letting it know. Once the mobile phone responds with "OK," the service level connection has been established, and we are ready to initiate the audio connection.

5.3 Dual-Tone Multi-Frequency Detection

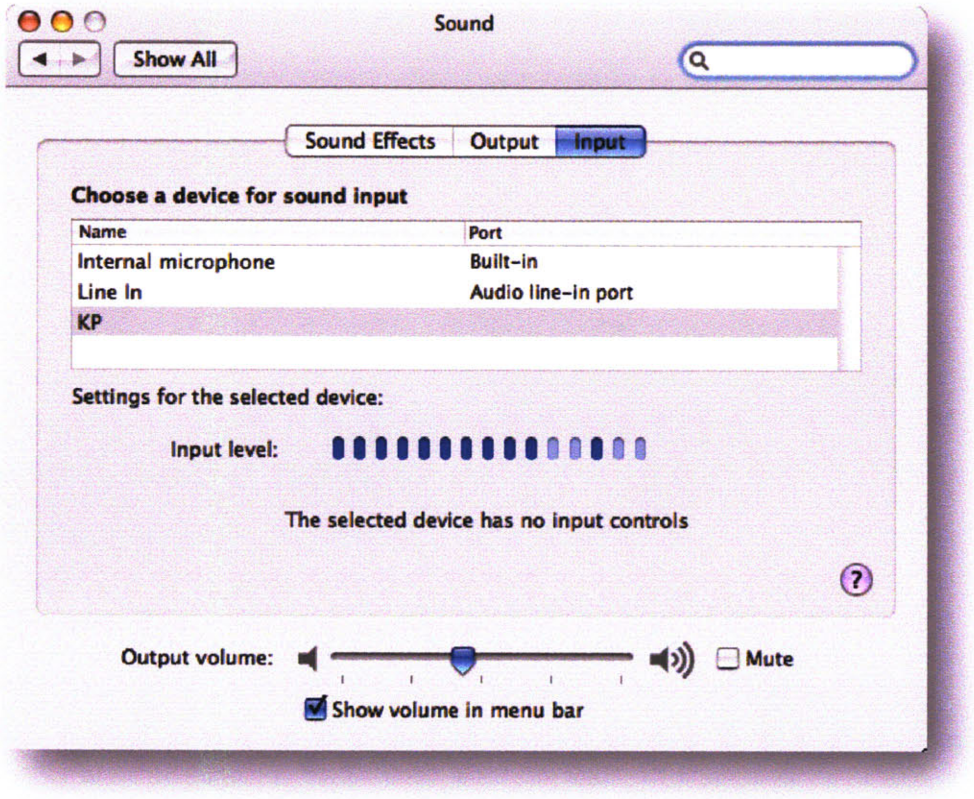
As mentioned in Section 4.4, our goal is to route all the audio from the mobile phone to our kiosk so we can process the audio to detect which numbers on the key pad the user is pressing. Up to this point we have properly configured the service level connection. Now, we have to create a SCO Audio Connection between the kiosk and the mobile phone.

5.3.1 Audio Device Driver for SCO Audio Connection

In order to establish the SCO Audio connection, we need to create an audio device driver in OS X that is associated with the mobile phone of the user. Luckily, there is a method provided in the Bluetooth APIs that does just that:

```
#import <IOBluetooth/IOBluetoothUserLib.h>

NSDictionary * emptyDic = [[NSDictionary alloc] init];

NSArray * innerArray = [[NSArray alloc] initWithObjects:emptyDic,nil];

NSDictionary * dict = [[NSDictionary alloc] initWithObjectsAndKeys: innerArray,
      @"IOAudioControls", nil];

// mBluetoothDevice is an instance of IOBluetoothDevice, representing the mobile
      phone
IOBluetoothAddSCOAudioDevice([mBluetoothDevice getDeviceRef], (CFDictionaryRef)dict);
```

`IOBluetoothAddSCOAudioDevice` takes in a reference to a Bluetooth device, as well as a dictionary specifying certain options about the audio device driver. As one can see, the dictionary we provided was empty, but if one wants to fiddle around with it, please refer to the header files of IOBluetoothUserLib. Assuming everything has been done correctly up until this point, a new device should appear under 'Sound' in 'System Preferences,' with the name of the user's mobile phone (Figure 29). It'll create two streams, one which is located under 'Output' and the other which is located under 'Input.' The stream under 'Input' represents the audio sent from the phone to the kiosk, while the stream under 'Output' represents the audio sent from the kiosk to the phone (which we aren't utilizing in our implementation). This audio device will remain even if the Bluetooth connection is severed, so we need to make sure to remove the audio device with `IOBluetoothRemoveSCOAudioDevice` when we are done with it.



Choose a device for sound input

Name	Port
Internal microphone	Built-in
Line In	Audio line-in port
KP	

Settings for the selected device:

Input level:

The selected device has no input controls

Output volume:

☐ Mute

☑ Show volume in menu bar

Figure 29: Audio Device Listing for Mobile Phone in OS X

Additionally, when the user causes an event that plays audio on the mobile phone, the 'Input Level' should change. It is also important to make sure that the user didn't mute the sounds on his/her phone, especially the touch-tones for the keypad. Now that we have added an audio driver, we need to get a handle on the audio stream programmatically.

5.3.2 Get Handle on Audio Stream

In order to get a handle on the audio stream, we used OS X's CoreAudio as well as an Objective-C Wrapper for CoreAudio called MTCoreAudio, written by Michael C Thornburgh. (13)

5.3.2.1 Audio Units

One of the tools provided by OS X CoreAudio is called Audio Units, which greatly simplifies audio processing and manipulation. Basically, one can take an input of audio and transmit it through a succession of audio units in order to provide different effects. Additionally, audio units can be re-used by many applications that understand the audio unit framework.

OS X provides a number of audio units that can be used to represent audio input as well as output. We created a string of audio units beginning with the audio input from the mobile phone using the driver created in Section 5.3.1. The special audio unit for external audio devices is called the **AudioDeviceOutput**. Although by default it is configured to act as an audio unit to output audio to an external audio device, it can be configured to accept input as well. (14) We connected this audio unit to another audio unit we created that processes the audio for the touch-tone frequencies, which fires certain actions depending on which key press is detected. In turn, this audio unit is connected to the sound output for the computer. The audio unit for the sound output is called the **DefaultOutputUnit**. These connections are illustrated in Figure 30. Although outputting the sound to the computer speaker is not necessary in our deployment, it does help in debugging the implementation. However, the ability to stream audio from one's mobile phone to one's Mac computer speakers provides a platform for many interesting applications.

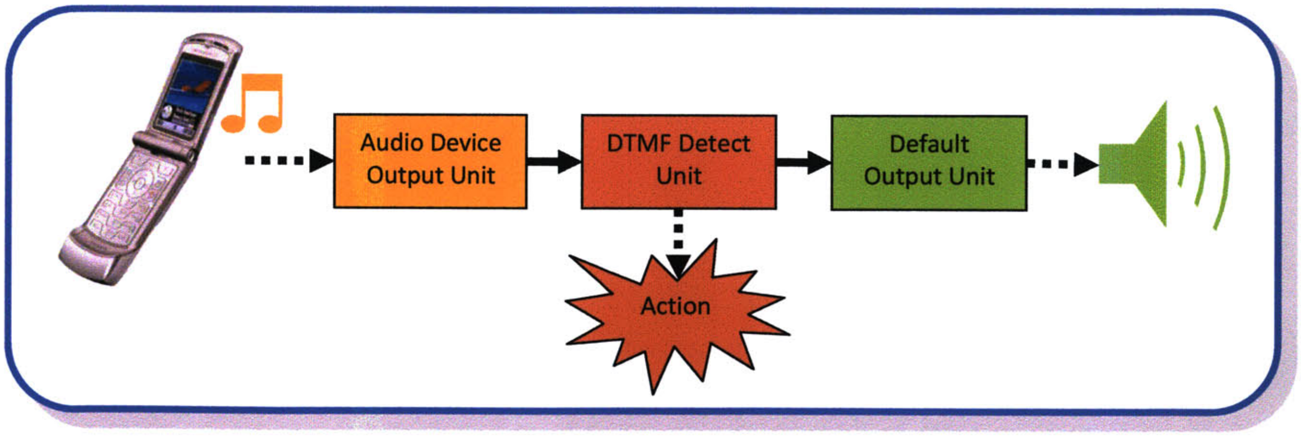


Figure 30: Audio Unit Connections

5.3.3 DTMF Detector Audio Unit

When one creates one's own audio unit, the CoreAudio SDK provides quite a bit of scaffolding that makes sure it operates as an audio unit should, and thus can be used by other applications that support audio units. Audio units that use OS X's scaffolding provide the ability to do simple conversions between audio formats (For more complex format changes, one would need to insert audio units that act as converters into the audio unit chain). In order to do digital signal processing in an audio unit, one must create an 'Effect Unit'(a type of audio unit), which allows one to override a method called 'Process.' The 'Process' method gives the audio unit access to each sound sample, and thus the ability to manipulate the sound stream. Within the 'Process' method, we implemented the Goertzel Algorithm, a well known technique for efficiently identifying the frequencies of a signal.

5.3.3.1 Goertzel Algorithm

The Goertzel Algorithm is the primary technique used in detecting key presses of a telephone. Each key of a telephone pad emits a signal with two overlapping frequencies, which uniquely identifies it from the other keys:

	1209 Hz	1477 Hz	1633 Hz
697 Hz	**1**	**2**	**3**
770 Hz	**4**	**5**	**6**
852 Hz	**7**	**8**	**9**
941 Hz	*****	**0**	**#**

Figure 31: Touch-Tone Frequency Mapping

The Goertzel algorithm computes a sequence $s(n)$, given an input sequence, $x(n)$:

$$s(n) = x(n) + 2cos(2\pi\omega)s(n-1) - s(n-2)$$

where ω is the frequency we are looking for, normalized with respect to our sampling frequency, which is 8000 KHz for SCO Audio. We run this algorithm on a sequence of sound samples for each frequency we are interested in (7 total, refer to Figure 31). This will provide us with the energy levels for each frequency. At some set interval, we then find the frequency with the max energy level for the rows and the columns, respectively. The higher the interval, the more accurate the detection, but the longer the process takes. The two frequencies with the maximum energy level then determine which key was pressed. Although this was a rather short explanation of how the process works, there are many resources on the web explaining how to implement this algorithm, and one can also refer to the source code for the DTMF Detector Unit. (15)

5.3.4 SCO Audio-Specific Considerations

Since common DTMF detectors usually operate on a more consistent audio connection, there are some specific issues that need to be monitored when it comes to executing DTMF detection over the Bluetooth SCO Audio connection:

- ***Ignore non-numeric-keypad audio.*** Often on mobile phones, there are other keys that generate their own tones, such as directional pads, and possibly keyboard keys. There has to be some tolerance that ignores these tones so that the kiosks do not respond to such events.
- ***Ignore hisses and pops.*** Occasionally the audio driver resets and a popping sound is generated. This is pretty easy to ignore because the decibel level is quite a bit higher than the normal level of key presses.
- ***Acknowledge Silence.*** There is a possibility that the user's phone is configured not to send touch-tones when a call is not in progress. If silence is detected for an extended period of time from the initial connection, we would display a notice to the user to check that touch-tones are generated and that the phone is not muted.
- ***Detect Noisy Connection.*** A SCO audio connection with a lot of static on it probably means that the mobile phone is too far away from the kiosk's Bluetooth adapter, and a noisy signal makes it more difficult to detect key presses. In this case, the kiosk would advise the user to move closer to the kiosk.

All of these issues can be handled with some relatively straight forward additions to the digital signal processing code of DTMF Detector, or adding an additional audio unit to the chain that handles a specific issue.

6 USAGE SCENARIOS

By taking advantage of the implementation outlined in Chapter 5, there are a number of interesting usage scenarios that can be deployed within the context of the DomeView system, as well as similar applications.

6.1 SIMPLE REMOTE CONTROL & KEYBOARD

Simple remote control and keyboard usage was touched upon in section 4.3.1, and is basically a prerequisite for any of the other usage scenarios in this chapter. At the very least, every Bluetooth-enabled mobile phone will be able to interact with applications running on the kiosk, by mapping numeric key-presses to emulated keyboard presses at the kiosk. The kiosk can have two modes. In one mode, the numeric keypad on the mobile phone will act as a directional pad. In the other mode, the numeric keypad can act as a keyboard input, as if the user was sending a text message.

Within the context of DomeView, every mobile phone user will be able to flip through ads that are posted to the kiosk ('4' navigates to previous post, '6' navigates to next post, Figure 32), as well as bring up a menu with additional options. The options presented on this menu are dependent upon the features supported by the mobile phone, which we can detect using SDP.

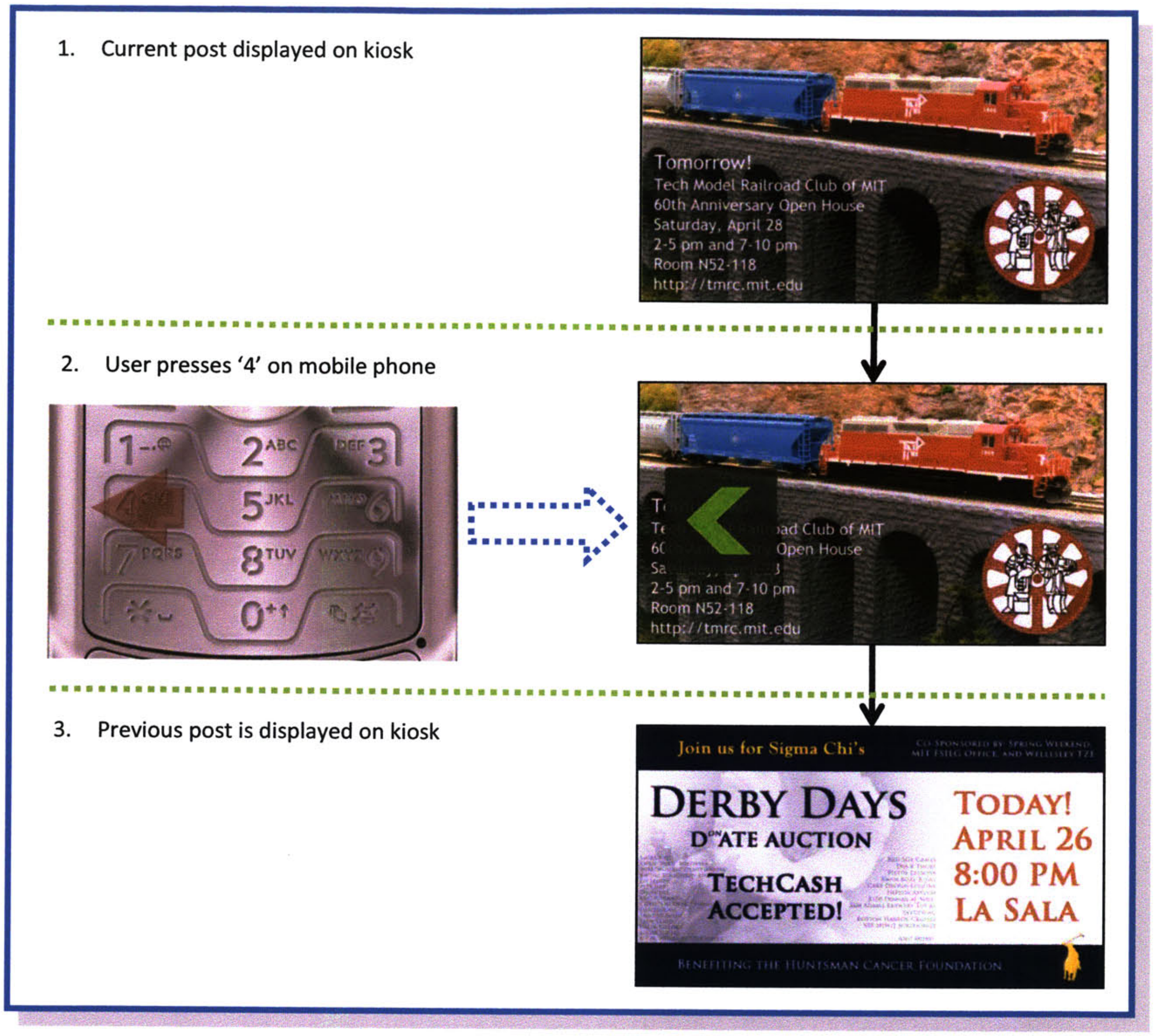

1. Current post displayed on kiosk
Tomorrow!
Tech Model Railroad Club of MIT
60th Anniversary Open House
Saturday, April 28
2-5 pm and 7-10 pm
Room N52-118
http://tmrc.mit.edu
2. User presses '4' on mobile phone
3. Previous post is displayed on kiosk
Join us for Sigma Chi's
DERBY DAYS
DATE AUCTION
TECHCASH ACCEPTED!
TODAY!
APRIL 26
8:00 PM
LA SALA
BENEFITING THE HUNTSMAN CANCER FOUNDATION

Figure 32: Navigate to Previous Post

6.2 Unique Identification

Since each Bluetooth adapter has a unique MAC address, we can uniquely identify every user that interacts with the kiosk. With this information, there is a number of interesting things that we can do in the context of the DomeView System:

- ***Track User Preferences.*** We can record which ads the user spends his/her time looking at, and based on that we can present future ads to the user that may align with his/her interests.

- ***Associate Mobile Phone with Email.*** If the user at some point associates his/her mobile phone's unique MAC address with an Athena account or other email address, we can enable the user to send emails to his/her self containing relevant information.
- ***Associate Mobile Phone with other Web Services.*** We can tap other user information, such as data provided to social networking sites like Facebook and MySpace, and leverage it to give the user a more personalized experience, such as providing updates on which 'friends' have interacted recently with the kiosk.
- ***Track User Location.*** Since we know the location of the kiosks, we can essentially determine the location of users at certain periods of time. The user may also enable others to see where they have been recently.
- ***Notify Ad Providers of Unique Views.*** We can record how many unique views an ad gets and notify/charge the ad providers of/for those views (discussed further in Section 6.5).
- ***Personal Graffiti/Message Wall.*** A user can leave notes on kiosks that are associated with his/her mobile phone, and may enable other users to view the wall, perhaps users who share the same group.

6.3 GAMES

If we extend the simple usage scenario outlined in Section 6.1, if the numeric keys on a mobile phone are mapped to act as a directional pad, then the user could potentially control an avatar on the display, such as a paddle in a Pong game (Figure 33). Additionally, we could accept text from the user as outlined in Section 6.1, and then enable the user to answer trivia questions that are posted on the display. Both of these scenarios could be extended to support multi-user interactions by deploying our implementation using multiple Bluetooth adapters and thus accepting multiple SCO Audio connections.

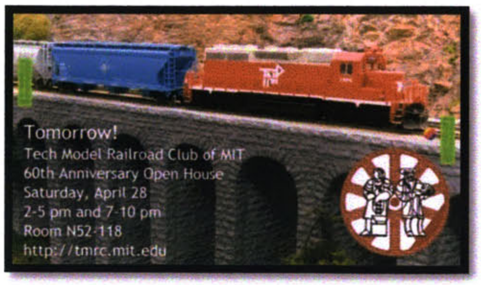


Figure 33: DomeView Pong with Green Paddles and Rainbow Ball

Since DomeView in its current state is primarily used for publicity of campus events and groups, we would not want to take away valuable time from the advertisers. Therefore, it would be interesting to see how one could potentially superimpose games on top of the advertisements, or maybe integrate the advertisements into the game.

6.4 Data Transfer

As more mobile phones begin to support PDA capabilities and the Obex protocol, these devices can exchange data, such as contacts, calendar items, and occasionally media files, with similarly capable devices. We can determine if a device is capable of transferring such items by checking which profiles it advertises in its SDP. If a device supports the 'Obex Object Push' profile, then we know we can send the device contact items and calendar items.

6.4.1 Contacts and Calendar Items

In the context of DomeView, when an advertiser submits a post, he/she enters metadata about the post, such as a title and a description. If the post is for advertising an event, the user can indicate the start and end time of the event. We could use this metadata to generate a calendar item and send it to the kiosk user's mobile phone. Similarly, we can use the advertisers account data to create a contact item that we could also send to the user's phone. The user takes advantage of the directional pad capabilities to navigate through menus, such as the menu illustrated in Figure 34, that provides access to data transfer options, while using the '5' key as a select button.

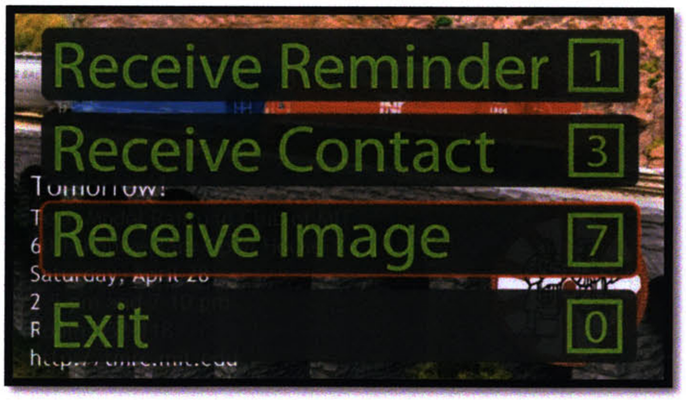


Figure 34: Data Transfer Options

The menu overlays the ad, so that the user can select to receive from the data options provided, and can also utilize keys '4' and '6', as described in Section 6.1, to flip through the ads and receive data relevant to other ads. Also note that the menu items also have numeric key shortcuts so that the user does not have to navigate with the directional pad in all instances.

6.4.2 Post Images and Coupons

If a mobile phone supports receiving images over OBEX (which isn't possible to determine beforehand, so we simply have to try it or try to determine it based on device data if we have it available), we could send a scaled down image of an actual post to that device. Additionally, we could allow advertisers to create coupons that could be sent in this way to the phone. For a coupon to be unique, there would have to be some dynamically generated portion of the image, which is feasible. If a mobile phone is unable to accept images, but the user still wants to receive this image in some fashion, he/she could request that the kiosk submits the image via email.

6.5 Pay-Per-Click

Since we are able to uniquely identify users that interact with the kiosk and can keep metrics on them as outlined in Section 6.2, we could potentially charge advertisers based on how many interactions users have had with a specific ad, as well as gauge the level of interest of users in advertisers' offerings.

7 Conclusion and Future Work

The implementation described in this thesis provides a solution to many of the problems that exist with other mechanisms for interaction with public kiosks. Additionally, it opens up a whole new world of possibilities of interactions that haven't been explored before due to functional as well as adoption limitations. With the advancement of mobile phone and Bluetooth technology, our PhoneView implementation can continue to be leveraged even further.

Although we explored a number of scenarios utilizing our kiosk interaction implementation, a number of our usage scenarios can be tested and evaluated in further depth, and we are sure there are additional usage scenarios out there that we haven't thought of that we hope others will explore.

7.1 Multi-User Applications

Our implementation only allows one interaction at a time, and we weren't able to explore what is technically involved in enabling multiple connections. We are also unsure of how easy it would be to add additional Bluetooth adapters to a Mac and address the adapters individually with the APIs provided from Apple. It may require some lower level device driver hacking on OS X, but it seems to be more readily possible with Linux.

7.1.1 Interface Issues

Once the technical details involved with multi-user interactions are figured out, there are a number of interface issues that need to be evaluated. In the context of DomeView, if you have multiple users connected, how do you decide who controls the screen? There seem to be a couple of possibilities:

- ***Let Users Fight***. This seems to be the easiest answer. We simply have the connected users fight over control of the screen, as if there were two input devices trying to control the kiosk at the same time.
- ***Split the Screen***. We could split the screen so that each user has his/her own sub-window to control. The sub-windows may become too small at some point, though.

- ***Time Allotment***. We have a timer up on the screen that switches to the next user once the timer runs out, or the user disconnects.

7.1.2 Interaction Among Users

An interesting component of multi-user interaction at the kiosks is how to encourage interaction among the users themselves. This may occur naturally as each user observes the other interacting with the same screen, but maybe the kiosk can do more to facilitate this communication by identifying the users in some fashion (possibly by the name associated with their phone), and maybe even facilitate the transfer of data between the mobile phones of the users, such as the exchange of contact information.

7.2 In-Depth Interface Study

We didn't really evaluate what the best user interface would be when it came to interacting with the kiosk through numeric key presses. It would be interesting to compare current UI paradigms in the context of our implementation, evaluating which users find most intuitive and user friendly.

7.3 Sending Data from Phone to Kiosk

Our usage scenarios deal primarily with the kiosk sending data to the phone, but there are many applications that could involve the sending of user generated data from the mobile phone to the kiosk. One interesting application is perhaps the ability to submit a post from your mobile phone by sending an image and some text, or possibly creating a photo sharing service, such as the one described in Previous Work. Another scenario might be a mechanism whereby users could RSVP to events or put themselves in some lottery drawing by submitting a vCard with their contact information.

7.4 Final Words

PhoneView provides a mechanism that can significantly enhance almost any public kiosk deployment in a straightforward and elegant manner. The implementation opens up so many possibilities for enhanced interaction, and can be the focal point for promoting community collaboration and improvements in individual efficiency. We have only touched the surface with the usage scenarios we provided, and look forward to additional scenarios that others investigate. When a mechanism such as PhoneView is paired with a community based information system such as DomeView, The ideals behind community-based information sharing and collaboration are fully realized.

REFERENCES

1. *On the Design of Personal & Communal Large Information Scale Appliances.* **Daniel, Russell M. and Gossweiler, Rich.** Atlanta : Springer Berlin / Heidelberg, 2001. 0302-9743.

2. *The notification collage: posting information to public and personal displays.* **Greenberg, Saul and Rounding, Michael.** Seattle, Washington : ACM Press, 2001. 1-58113-327-8.

3. *Kimono: kiosk-mobile phone knowledge sharing system.* **Huang, Albert, Pulli, Kari and Rudolph, Larry.** Christchurch, New Zealand : ACM Press, 2005. Proceedings of the 4th international conference on Mobile and ubiquitous multimedia . Vol. 154, pp. 142-149. 0-473-10658-2.

4. *Digital graffiti: public annotation of multimedia content.* **Carter, Scott, et al.** Vienna, Austria : ACM Press, 2004. CHI '04 extended abstracts on Human factors in computing systems . Vol. Conference on Human Factors in Computing Systems , pp. 1207-1210. 1-58113-703-6.

5. *Exploring bluetooth based mobile phone interaction with the hermes photo display.* **Cheverst, Keith, et al.** Salzburg, Austria : ACM Press, 2005. Proceedings of the 7th international conference on Human computer interaction with mobile devices & services . Vol. 111, pp. 47-54. 1-59593-089-2.

6. *MobiLenin combining a multi-track music video, personal mobile phones and a public display into multi-user interactive entertainment.* **Scheible, Jürgen and Ojala, Timo.** Hilton, Singapore : ACM Press, 2005. Proceedings of the 13th annual ACM international conference on Multimedia. pp. 199-208. 1-59593-044-2.

7. **Kowalke, Mae.** Gartner:Mobile Phone Sales Up 21.5 Percent During 3Q 2006. *TMCnet on the Web.* [Online] Technology Marketing Corporation, November 22, 2006. [Cited: May 24, 1007.] http://www.tmcnet.com/ce/articles/3751-gartner-mobile-phone-sales-up-215-percent-during.htm.

8. **Millward Brown.** Millward Brown Results 2007. *Bluetooth.com.* [Online] 2007. [Cited: April 16, 2007.] http://www.bluetooth.com/Bluetooth/Press/Millward_Brown_Results_2007.htm.

9. **Apple, Inc.** Bluetooth Device Access Guide. *Apple, Inc.* [Online] May 23, 2006. [Cited: April 17, 2007.] http://developer.apple.com/documentation/DeviceDrivers/Conceptual/Bluetooth/Bluetooth.pdf.

10. **Apple, Inc.** Core Audio Overview. *Apple, Inc.* [Online] January 8, 2007. [Cited: April 17, 2007.] http://developer.apple.com/documentation/MusicAudio/Conceptual/CoreAudioOverview/CoreAudioOverview.pdf.

11. **Bluetooth Sig, Inc.** HANDS-FREE PROFILE 1.5. *Bluetooth: The Official Bluetooth Member Site.* [Online] November 25, 2005. [Cited: April 19, 2007.]

12. **Apple, Inc.** Apple - Software - AppleScript - GUI Scripting. *Apple, Inc.* [Online] Apple, Inc., 2006. [Cited: April 19, 2007.] http://www.apple.com/applescript/uiscripting/.

13. **Thornburgh, Michael C.** MTCoreAudio.framework. *The Armory.* [Online] [Cited: April 21, 2007.] http://aldebaran.armory.com/~zenomt/macosx/MTCoreAudio/.

14. **Apple, Inc.** Technical Note TN2091: Device input using the HAL Output Audio Unit. *Apple Developer Connection.* [Online] Apple, Inc., July 25, 2006. [Cited: April 23, 2007.] http://developer.apple.com/technotes/tn2002/tn2091.html.

15. **Banks, Kevin.** The Goertzel Algorithm. *Embedded.com - Thinking Outside the Box.* [Online] CMP United Business Media, August 28, 2002. [Cited: April 23, 2007.] http://www.embedded.com/story/OEG20020819S0057.