SocialAware: Context-Aware Multimedia Presentation via Mobile Social Networks

by

Charles M. Gartrell

B.S., Virginia Tech, 2000

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Master of Science

Department of Computer Science

2008

This thesis entitled:
SocialAware: Context-Aware Multimedia Presentation via Mobile Social Networks
written by Charles M. Gartrell
has been approved for the Department of Computer Science

_____

Dr. Richard Han

_____

Dr. Shivakant Mishra

_____

Dr. Leysia Palen

Date _____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Gartrell, Charles M. (M.S., Computer Science)

SocialAware: Context-Aware Multimedia Presentation via Mobile Social Networks

Thesis directed by Dr. Richard Han

The rise of online social networking and mobile computing presents a unique opportunity for context-aware computing. In this thesis, I present a framework for building social-networking-enabled context-aware services, called SocialAware, and two prototype applications that implement this architecture. The first application, called SocialAwareTunes, is a context-aware music jukebox player that plays music which reflects the preferences of a group of co-located users. The second application, called SocialAwareFlicks, is a context-aware video system that plays movie trailers which reflect the preferences of one or more users watching a video display together. These two prototype applications demonstrate the feasibility of the SocialAware framework, and point to the potential for some interesting future work.

Dedication

To my family and friends, who are my invaluable advisors.

Acknowledgements

Many people were helpful in the creation of this thesis, and I'm grateful to all of them. I'd like to thank my advisor, Rick Han, for his guidance throughout this work. Thanks to my committee members, Leysia Palen and Shiv Mishra, for their feedback and support.

A number of fellow graduate students were helpful along the way. In particular, Aaron Beach stands out. Our collaborative efforts have proved invaluable.

Finally, I thank my family and friends for their patience, support, and understanding.

Contents

Chapter

Figures

Figure

Chapter 1

Introduction

Online social networks have become quite popular in recent years [5] [14] [9]. As of November 2008, Facebook had over 120 million active users [6]. While 85% of four-year U.S. college students have a Facebook profile, the fastest growing group of Facebook users are those 25 years old and older. With millions of active users, social networks present a clear opportunity for applications that can leverage this data.

In [22] a vision was presented for WhozThat, a system for answering the basic social question of "Who's that?". WhozThat ties together online social networks with mobile smartphones, and provides an infrastructure for building a new class of context-aware applications that leverage social network information. Social networks provide access to an extensive collection of personal information about users, including name, gender, contact information, interests, music preferences, movie preferences, book preferences, and friendship connections with others, which represent but a few of the fields that users populate on their social network profiles. Once a computationally-enabled environment knows personal contextual information about its users, it can take a number of actions in response to this information. One class of such actions involves tailoring multimedia, such as music and video, for local presentation to one or more co-located users based on the preferences of those users.

In this thesis I present SocialAware, a framework for building social-networking-enabled context-aware services. To demonstrate the feasibility of this framework, pro-

totypes of two context-aware multimedia presentation applications have been implemented. The first application, called SocialAwareTunes, plays music that reflects the preferences of one or more users residing in a common physical space such as a bar or restaurant. The second application, called SocialAwareFlicks, displays recommended movie trailers that match the movie preferences of one or more users jointly watching a common display. SocialAwareFlicks could be deployed in locations such as video-rental establishments for marketing purposes, to make customers aware of new video rentals that match their interests.

SocialAware applications use WhozThat's identity sharing protocol for exchanging users' social network identifiers among mobile devices. While the WhozThat protocol is designed to be agnostic of particular wireless networking technologies, the current implementation for this thesis uses Bluetooth. The current version of Bluetooth, Bluetooth 2.1, provides a peak data rate of 3 Mbits/s and a typical range of approximately 10 m for most cell phones. Bluetooth is widely implemented in many cell phones today, and is currently available in at least one billion devices worldwide [17]. Since Bluetooth is so widespread, it is an ideal technology for implementing the WhozThat protocol.

The contributions of this thesis include the SocialAware framework, and the SocialAwareTunes and SocialAwareFlicks multimedia applications. In contrast to known existing frameworks for context-aware systems, the SocialAware framework enables applications to leverage social context and personal information from online social networks. To the best of our knowledge, SocialAwareTunes and SocialAwareFlicks are the first multimedia applications that adapt their presentation content to individual and group social networking identity and preferences.

I have prototyped the SocialAwareTunes and SocialAwareFlicks applications using the Java Micro Edition (ME) Mobile Information Device Profile (MIDP) 2.0 platform, which is supported on many mobile phones, and the Java Standard Edition (SE) platform, which is supported on operating systems such as Windows, Mac OS X, and

Linux. The prototypes were evaluated using the MicroEmulator [12] and BlueCove JSR-82 Emulator [3] open source tools. These tools allowed me to test and evaluate the performance of SocialAwareTunes and SocialAwareFlicks using mobile phone device emulators running on a standard Macbook Pro notebook computer.

In the following, Chapter 2 summarizes related work. Chapter 3 describes the design and architecture of the SocialAware framework and the SocialAwareTunes and SocialAwareFlicks applications, while Chapter 4 describes the prototype implementation of these applications. In Chapter 5, I describe the future work left to be explored regarding SocialAware and these applications. Chapter 6 concludes the thesis.

Chapter 2

Related Work

This chapter describes related work in the areas of mobile social networking, context-aware computing, and recommender systems. Prior work in each of these areas has bearing on the SocialAware framework and the SocialAwareTunes and SocialAware-Flicks applications.

2.1 Mobile Social Networking

Most current approaches toward integrating social networks with mobile devices have missed the opportunity to bind social context tightly with the local context of interacting people. These approaches simply extend the Web interface of the social network to the mobile device, by providing a view of the user's social network on his/her mobile phone[2]. Recent work such as CenceMe has sought to enrich the amount of information flowing in the direction from the mobile device to the social network, e.g. the location of the user and perhaps context cues such as whether the person is talking [27]. While these cues provide more information about the user's context, they do not integrate social network information about nearby users with the user's local context. In contrast, the WhozThat system used by SocialAware applications exploits mobile computing technology to import contextual information from social networking sites into the user's local physical environment. Serendipity [25] is a system similar to WhozThat that imports social context into the local context using mobile devices. How-

ever, it populates its own database of social context information rather then connecting with popular online social networking sites, and does not consider the development of context-aware applications as discussed in this thesis.

Commercial location-aware mobile social networking services, such as BrightKite [4] and Loopt [10], provide some of the functionality found in WhozThat. However, like Serendipity, these services populate their own databases with social networking and context information, rather than leveraging popular online social networking sites such as Facebook. Also like Serendipity, these services do not consider the development of context-aware applications such as those enabled by SocialAware.

2.2 Context-Aware Computing

The term "context-aware" was first introduced by Schilit et al. in [31] and [30]. They described context-aware software as software that "adapts according to the location of use, the collection of nearby people, hosts, and accessible devices, as well as to changes to such things over time". In [30], context-aware computing systems consist of a collection of mobile and stationary computing devices which communicate and cooperate on behalf of a user. The authors describe four aspects of context-aware applications: proximate selection, automatic contextual reconfiguration, contextual information and commands, and context-triggered actions. Proximate selection is a user interface technique for facilitating the selection of nearby objects, such as a nearby printer, over objects that are located further away from the user. Proximate selection is not considered in the SocialAware framework. However, the remaining three aspects are present in SocialAware applications. For example, in SocialAwareFlicks, automatic contextual reconfiguration is used to reconfigure the playlist of currently playing movie trailers as users enter and leave the physical vicinity of the shared screen. In the SocialAwareTunes architecture, contextual information and commands are supported through a provision for allowing the user information to view information about the currently playing music

track on his/her mobile device. Context-triggered actions are used in SocialAwareFlicks to specify the rules (heuristics) for generating the movie trailer playlist based on the interests of nearby users.

As described in [32], much of the early work on context-aware computing focused on location as context, providing location-awareness using the Global Positioning System (GPS) or beacons [33]. Work such as [32] [24] reminds us that context is about more than just location. Context also includes information about physical conditions such as light and temperature, nearby computing resources, the user's social environment (who else is nearby, group dynamics, etc.), and the user's tasks (current task that the user is engaged in, general goals, etc.). The SocialAware framework is one of the first projects to focus on the importance of social context and how information from online social networks can be used to enrich social context information.

In [24], Dey presents the Context Toolkit architecture, which can be used to add context information to non-context-aware applications and enhance existing context-aware applications. The Context Toolkit consists of abstracted sensing units, called widgets; interpreters, which interpret sensor data from one or more widgets to determine a more general context; and aggregators, which collect all context information about a single entity such as a person. Baldauf et al. provide a recent survey of context-aware systems and frameworks in [21]. Many of the frameworks surveyed in this paper seek to provide general purpose solutions for context-aware systems. In contrast to these general purpose frameworks, SocialAware is a focused effort to provide a framework that applications can use to leverage social context and personal information from online social networks.

2.3 Recommender Systems

Recommender systems seek to recommend items (movies, music, news, etc.) that may be of interest to a user. Recommender systems became an independent area of

research in the mid-1990s [20]. [20] provides a recent survey of recommender systems. There are three main types of recommender systems described in this paper: content-based, collaborative, and hybrid. Content-based recommenders use information about items that the user has rated highly in the past to recommend new items that are similar to these previously rated items. Collaborative recommenders recommend new items to a user (the target user) based on items previously rated by other users. Collaborative recommenders seek to find other users that have tastes similar to the target user. Hybrid recommenders combine content-based and collaborative methods. Many recommender systems have been been built in academia and industry. [23] briefly describes how the Netflix [15] recommendations system, called Cinematch, works. The Cinematch system was recently exposed to the public through the Netflix API Web service [16]. This Web service is used by SocialAwareFlicks to generate playlists of recommended movie trailers.

While recommender systems are not the focus of this thesis, the use of contextual information in recommender systems brings up a number of interesting challenges. In SocialAwareTunes and SocialAwareFlicks, we are concerned with methods for generating a playlist of songs or movie trailers that satisfies the group of detected users. [19] describes a multidimensional approach to recommender systems that incorporates contextual data such as with whom the user will be watching a movie, and compares the multidimensional approach to the standard two-dimensional approach. [28] presents PolyLens, which is a collaborative filtering recommender system for groups of users. The authors consider two approaches to generating group recommendations. The first approach is to produce recommendations for a "pseudo-user" that represents the group's tastes. The second approach is generate recommendation lists for each group member and merge the lists. The authors chose to implement the second approach, and both user surveys and observation of user behavior indicated that users liked and used the group recommendations generated by PolyLens. 77% of the users in their survey found

group recommendations more useful than individual recommendations when deciding on a movie to see, which suggests that users may find group viewing of recommended movie trailers in SocialAwareFlicks to be useful. In designing SocialAwareTunes and SocialAwareFlicks, I chose the approach of generating recommendations for each user and merging the lists. Several heuristics for generating group playlists are presented in Chapter 3.

Chapter 3

SocialAware System Design and Architecture

This chapter describes the design and architecture of the SocialAware framework and the SocialAwareTunes and SocialAwareFlicks applications. SocialAware applications consist of two components: the stationary application component, and the mobile application component. The stationary application component is embedded into the user's environment, while the mobile application component resides on mobile devices. The stationary component is responsible for detecting the presence of users, obtaining information about these users from a social networking Web site, and performing actions based on this information. The mobile component on a user's mobile device is responsible for advertising that user's social network profile identifier using a wireless technology such as Bluetooth so that it is available to the stationary component. A number of security and privacy implications are associated with SocialAware applications, but a consideration of these issues is beyond the scope of this thesis. Figure 3.1 shows the architecture of SocialAware.

3.1 Stationary Component

The SocialAware framework provides a stationary component that is embedded into the user's environment. In the SocialAwareTunes application, this component detects the Facebook ID of each nearby user and obtains information about the users' music preferences from their Facebook profiles. In the SocialAwareFlicks application,

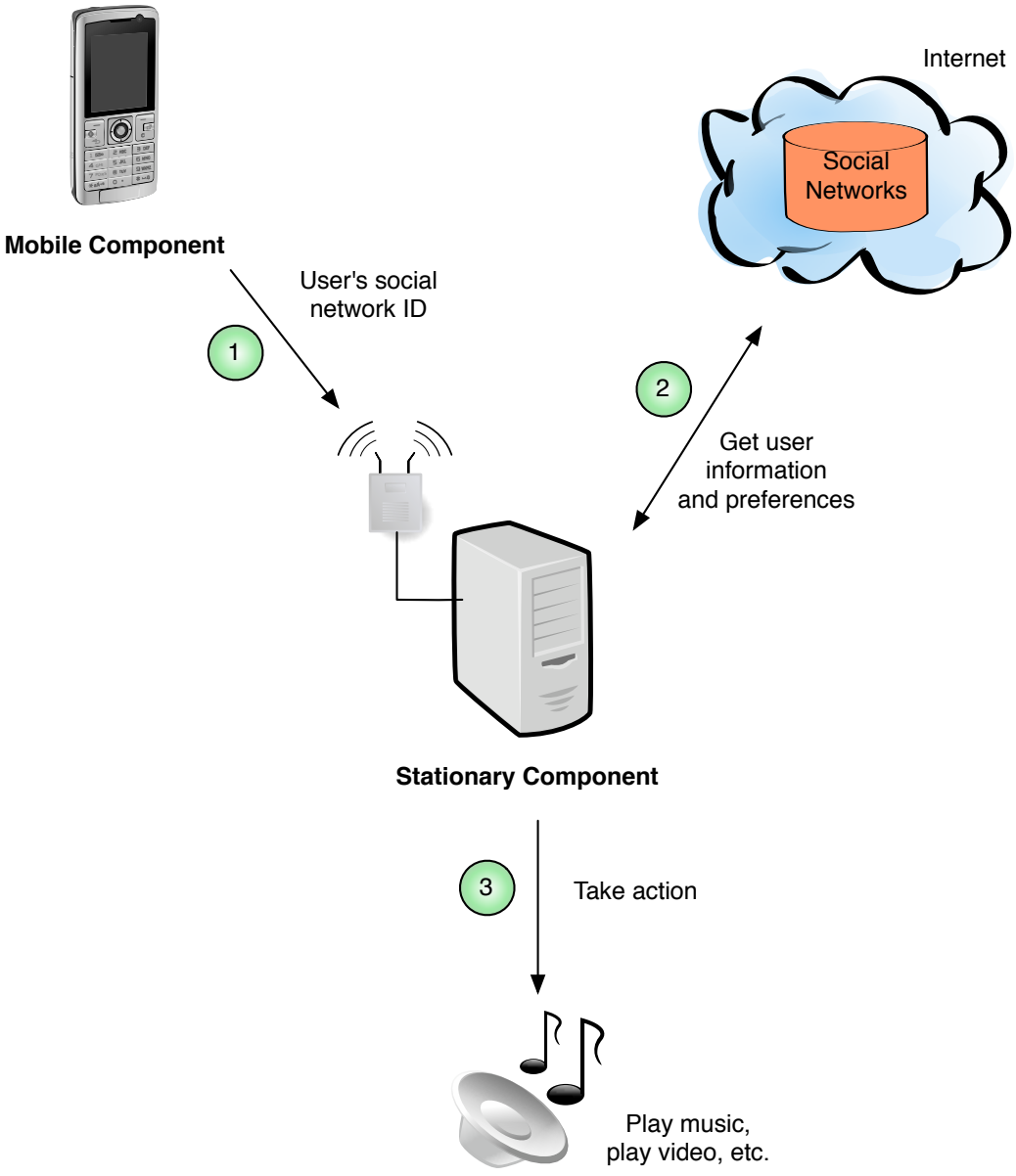

Figure 3.1: SocialAware Architecture

this component detects the Facebook ID of each nearby user and obtains information about the users' movie preferences from their Facebook profiles.

After obtaining information about detected users from a social networking Web site, the stationary component must use this information to infer something about the users and take appropriate action based on this inference. For the SocialAwareTunes and SocialAwareFlicks applications, an external Web service is accessed to obtain music or movie recommendations based on the preference information provided on a user's Facebook profile. After obtaining these recommendations for each user, we can use a number of heuristics to create a music or movie trailer playlist that accommodates the possibly varying tastes of the detected users in the group. The next subsection, section 3.1.1, describes some of the heuristics that were considered for SocialAwareFlicks.

Figure 3.2 shows the architecture for the stationary component in SocialAwareFlicks.

3.1.1 Playlist Generation Heuristics for SocialAwareFlicks

A number of heuristics can be used to create the movie trailer playlist in SocialAwareFlicks. These heuristics can be grouped into two categories: heuristics for generating the playlist for one user, and heuristics for generating the playlist for a group of users. This discussion will assume that the user's social network profile provides a list of the titles of the user's favorite movies. We also assume we have access to a recommender service for programmatically obtaining a list of recommended or similar movie titles for a specified movie title. As will be shown in the Implementation chapter of this thesis (Chapter 4), these are reasonable assumptions.

Several heuristics were explored for generating the playlist for one user. Most of these heuristics depend upon the metadata that we can obtain for a movie. Here I will assume that the release date, rating (zero to five stars), genre, the list of directors, and the list of cast are available for a specified movie title. After obtaining a list of

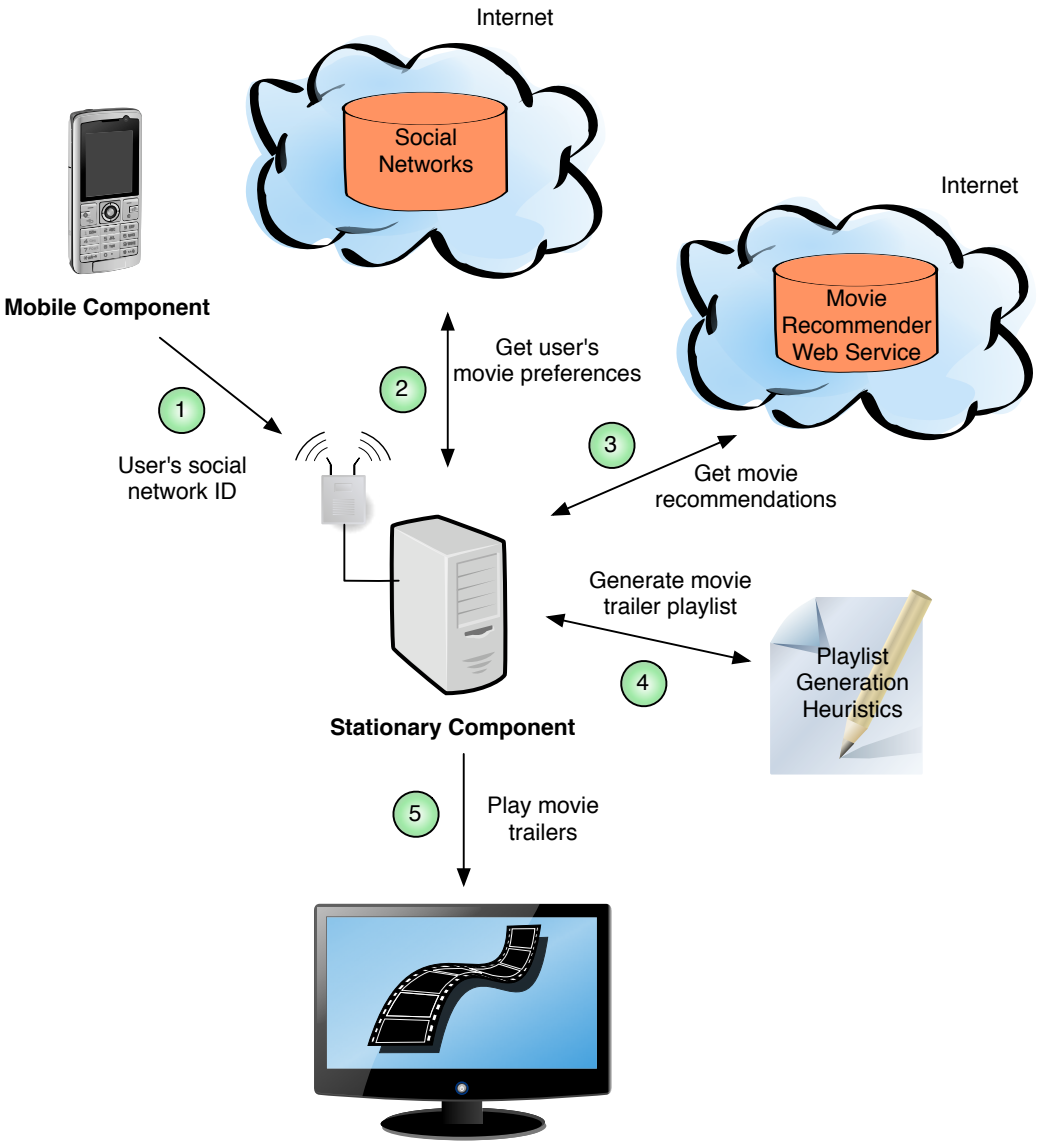

Figure 3.2: SocialAwareFlicks Stationary Component Architecture

recommended movies for each of the user's specified favorite movies, one method for generating the playlist would be to sort the aggregate list of recommended movies using criteria based on the available movie metadata. For example, we could sort the list of recommended movies by release date, under the assumption that the user would prefer to see movie trailers for the most recently released movies first. We could also sort the list of recommended movies by rating, under the assumption that the user would prefer to see trailers for the most highly rated movies first. Yet another heuristic would be to select a limited subset of the list of recommended movies and then sort this limited subset using the aforementioned criteria. For example, we could obtain the first five recommended movies for each of the user's favorite movie titles, and then sort this aggregate list by release date. Figure 3.3 summarizes these heuristics for generating the movie trailer playlist for a single user and shows examples of these heuristics for a user who's favorite movies are "The Matrix: Revolutions" and "Mission: Impossible".

The task of generating a movie trailer playlist for a **group** of co-located users watching the same screen presents unique issues and challenges. How do we account for the possibly varying tastes of the users watching the screen? Several heuristics were explored for generating a group playlist. The first heuristic assumes that single-user playlists have been generated for each user in the group using one of the methods described previously. After generating these single-user playlists, we can select a subset of each playlist and combine each subset into an aggregate playlist for the group. For example, we could list the top two most highly rated movies from each user's playlist and concatenate these lists to form a playlist for the group. The second heuristic that was considered takes a different approach. The objective of this heuristic is to find common features that are shared by all or most of the single-user playlists. We begin by generating single-user playlists for each user in the group using one of the methods described previously. Then, we examine each single-user playlist and look for common features that are found across these playlists. For example, we can list the dominant

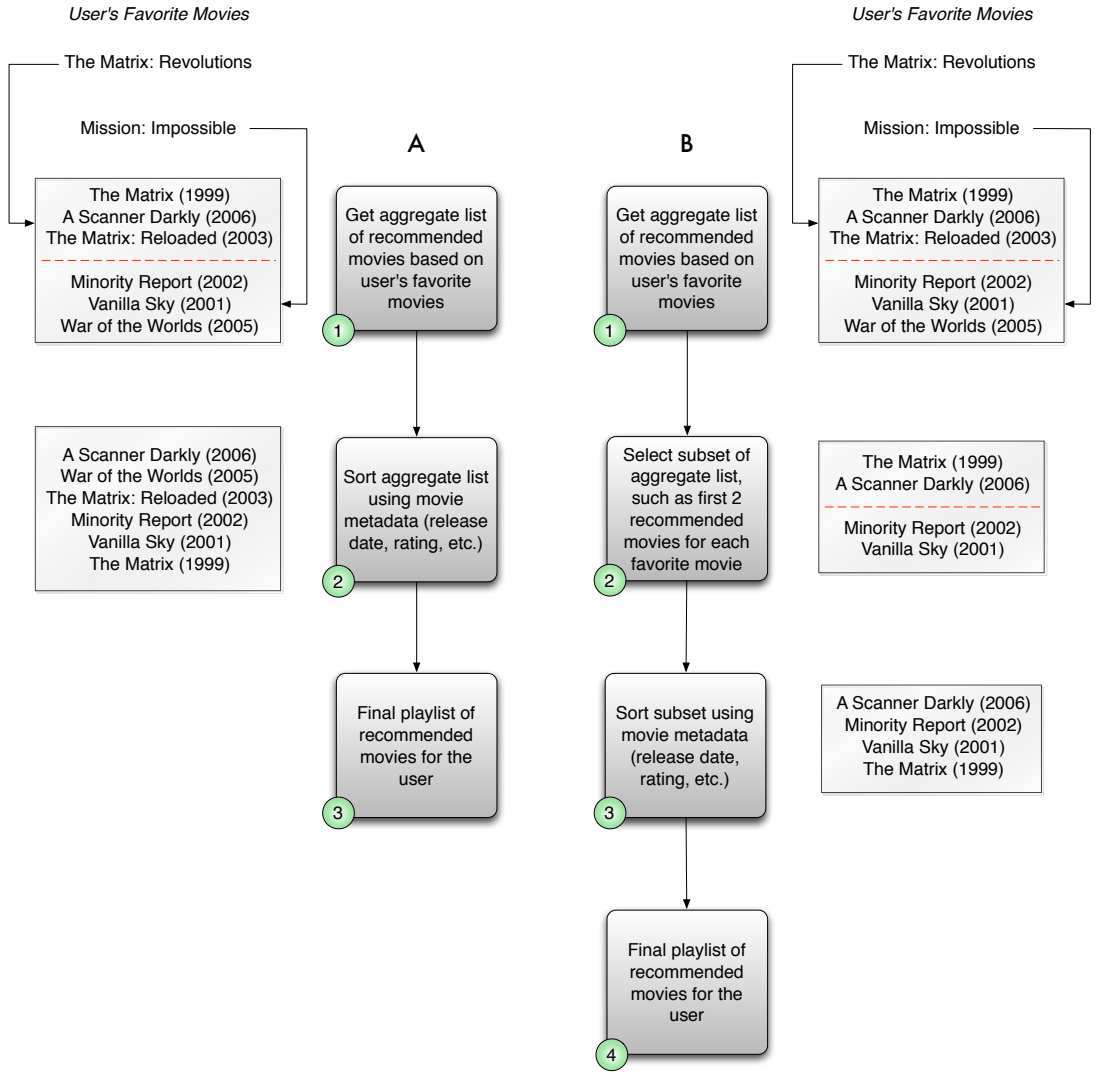

Figure 3.3: Single-User Playlist Generation Heuristics for SocialAwareFlicks

genre in each playlist, where the dominant genre is defined as the genre for which 50% or more of the movies in the playlist belong. After finding the dominant genre in each single-user playlist, we find the dominant genre among the group of users. Again, the dominant genre for the group in this example as the defined as the genre for which 50% or more of the single-user playlists have this genre as their dominant genre. If we have found a dominant genre for the group, then we proceed by selecting two films for this genre from each single-user playlist and concatenating these selections to build a playlist for the group. The goal here is to build a movie trailer playlist that satisfies the preferences of the majority of the group. Figure 3.4 provides an illustration of both classes of group playlist-generation heuristics and shows examples of these heuristics for a group composed of two fictional users, Mike and Fred.

3.2 Mobile Component

The mobile component of the SocialAware framework is found on a user's mobile device. The mobile component allows users to log in to their social networking accounts. After the user logs in, the mobile component will obtain the user's social network identifier and share this ID with the stationary component of SocialAware. The user's social network ID will typically be shared with the stationary component using a short-range wireless technology such as Bluetooth, although other mechanisms may be used since SocialAware treats the ID sharing technology as an abstraction.

The SocialAware framework also allows the user to interact with the local context-aware service using the mobile component. In SocialAwareTunes, the user might use the mobile component to view the information about the song that is currently playing, such as song name and artist, and view a list of upcoming songs in the current SocialAwareTunes playlist. In SocialAwareFlicks, the user might use the mobile component to view information about the movie trailer that is currently playing, such as movie title and release date, and request that SocialAwareFlicks skip the remainder of the current movie

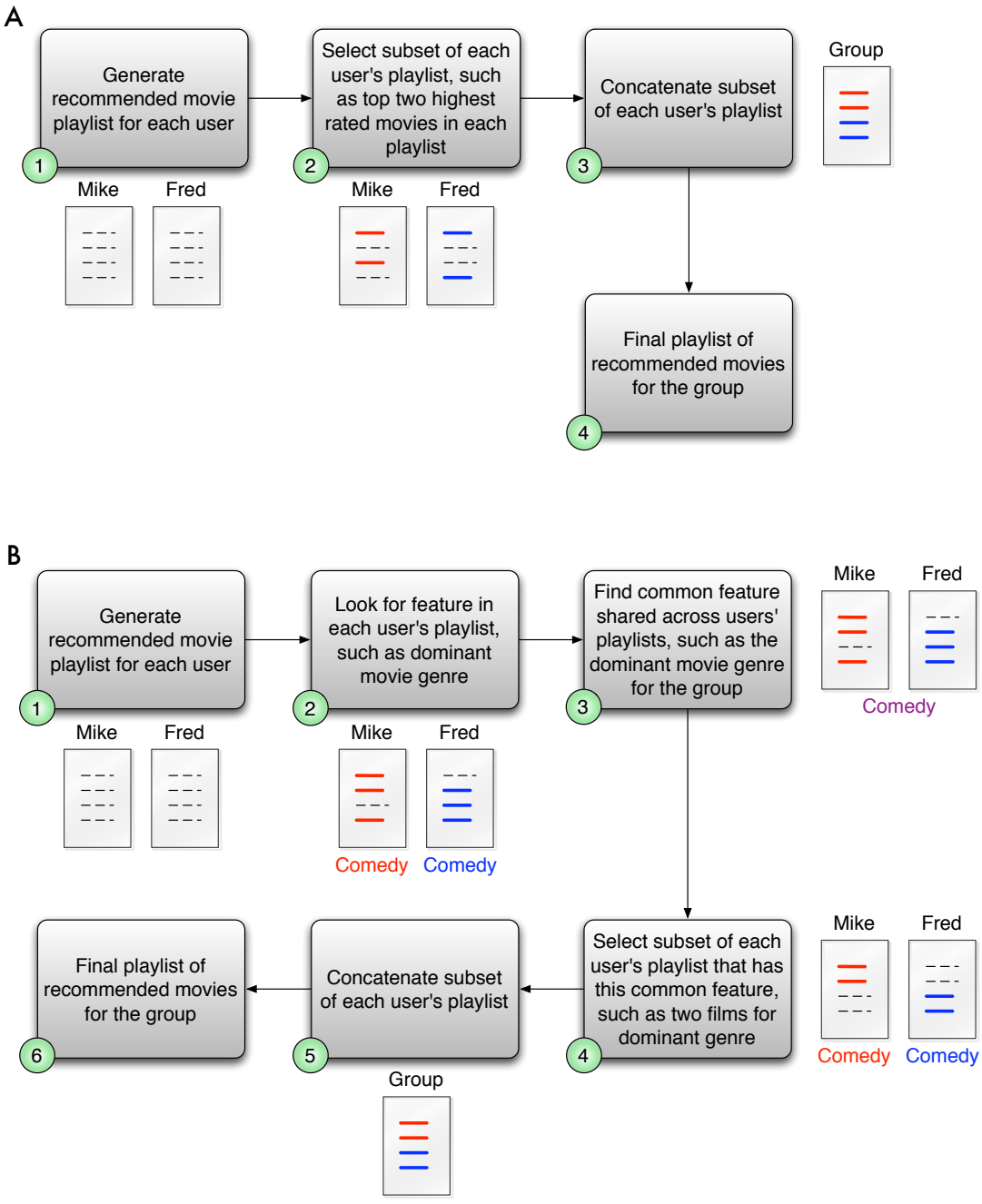

Figure 3.4: Group Playlist Generation Heuristics for SocialAwareFlicks

trailer and begin playing the next movie trailer in the playlist. SocialAware applications treat the mechanism used for interaction with the user as abstracted bidirectional communication channel. This abstraction is flexible enough to support user interaction over a variety of network technologies, including short-range wireless technologies such as Bluetooth or WiFi, or wireless wide area network (WWAN) connectivity available on many smartphones.

Figure 3.5 shows the architecture for the mobile component in SocialAwareFlicks.

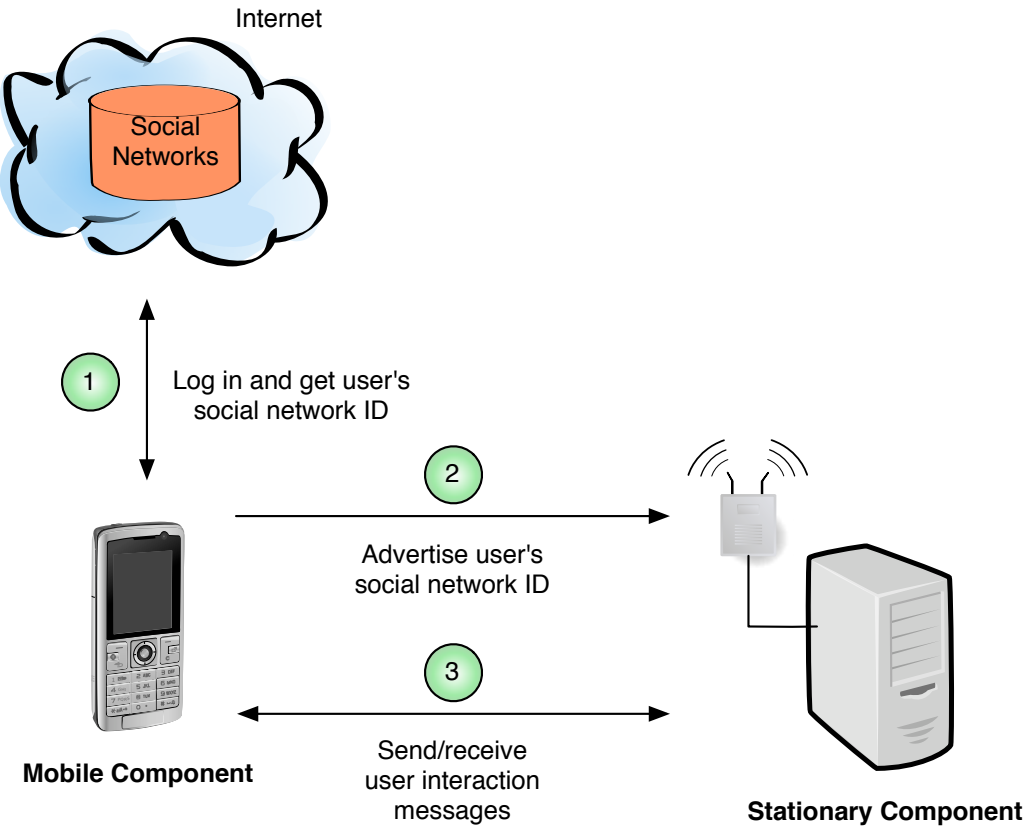

Figure 3.5: SocialAwareFlicks Mobile Component Architecture

Chapter 4

Implementation Results

This chapter describes the implementation of prototypes for the SocialAwareTunes and SocialAwareFlicks multimedia applications. Like all applications that use the SocialAware framework, both of these applications consist of stationary and mobile components. The stationary components were prototyped using the Java Standard Edition (SE) 5.0 platform, while the mobile components were prototyped using the Java Micro Edition (ME) Mobile Information Device Profile (MIDP) 2.0 platform. All testing was performed using a Macbook Pro notebook running Mac OS X 10.5. The mobile components were evaluated using the MicroEmulator [12] tool, which is a pure Java implementation of Java ME in Java SE. All interactions between the stationary and mobile components are managed using Bluetooth. The BlueCove JSR-82 Emulator [3] tool was used to emulate all Bluetooth connectivity.

The Marge framework [11] was used to implement all of the Bluetooth communication logic in SocialAwareTunes and SocialAwareFlicks. The Java API for Bluetooth (JSR 82) [7] is fairly low-level and reasonably difficult to use for those who are not familiar with the technical details of the Bluetooth standard. Marge provides an abstraction layer above JSR-82 that hides the low-level details of managing Bluetooth connections, message exchanges, protocols, device discovery, and service discovery.

4.1 SocialAwareFlicks

4.1.1 Mobile Component

The mobile component (MC) in SocialAwareFlicks is responsible for sharing a user's Facebook ID with the stationary component (SC) using Bluetooth. The MC user interface presents that Facebook login screen to the user on the user's mobile phone. Note that the mobile device running the MC component needs Internet access over a WWAN or wireless local area network (WLAN) connection so that the Facebook Web site can be accessed. After the user enters his/her Facebook user name and password, the MC logs into Facebook and obtains the user's Facebook ID using the Facebook REST API Web service [1]. The MC then proceeds to launch a SocialAwareFlicks Bluetooth service that enables the user's Facebook ID to be shared with the SC. The MC uses the Marge framework to set up and launch the SocialAwareFlicks Bluetooth service.

4.1.2 Stationary Component

As described in Chapter 3, the SC in SocialAwareFlicks is responsible for detecting the presence of users watching a single common screen managed by the system and showing recommended movie trailers based on the favorite movie preferences expressed on each user's Facebook profile. The SC begins by searching for available mobile devices running the SocialAwareFlicks Bluetooth service. Since Bluetooth radios on mobile phones are Bluetooth Class 2 devices with an approximate range of 10 meters, only nearby users will be detected. The SC registers several listener objects with the Marge framework to manage the search for available mobile devices and to manage the state of the connection to the mobile device. After detecting an appropriate mobile device, we query the device to determine the user's Facebook ID. In the current implementation, the user's Facebook ID is encoded as the Bluetooth service name of

the SocialAwareFlicks Bluetooth service running on the mobile device. We are able to identify the SocialAwareFlicks Bluetooth service by looking for a particular universally unique identifier (UUID) advertised by all mobile devices that run this service.

After obtaining the Facebook ID of each detected user, the SC constructs a playlist of recommended movie trailers for each user. First the SC uses the Facebook REST API Web service to obtain the contents of the "favorite movies" field on the user's Facebook profile. We assume that this is a comma-separated list of the user's favorite movie titles. Next, the SC uses the Netflix REST API Web service [16] to obtain a list of similar movies for each of the user's favorite movies. We assume that the user will prefer to watch movies that are similar to his/her favorite movies. If just one user was detected by the SC, then one of the single-user playlist generation heuristics can be applied to generate the final movie trailer playlist. For the SocialAwareFlicks prototype, a version of the second class of single-user heuristics described in section 3.1.1 was implemented, where the first five recommended movies for each of the user's favorite movie titles are concatenated into a single aggregate list, and this aggregate list is sorted by movie release date. If multiple users were detected, then one of the group playlist generation heuristics can be applied to generate the playlist. A version of the first class of group heuristics described in section 3.1.1 was implemented for this prototype, where the top three most highly rated movies from each user's playlist are concatenated to form a playlist for the group. After generating the playlist, the SC searches for available video files on the local filesystem that correspond to the movie trailers on this playlist. For this prototype, publicly available movie trailer QuickTime video files were downloaded from the robertkaye.org Movie Trailer Archive Web site [13]. After attempting to find a movie trailer file for each title on the playlist, the SC begins playing the trailers on the screen using the VLC media player [18].

The SC performs the search for available mobile devices every ten seconds, although this search frequency can be adjusted. The current list of detected users is

compared with the previous list of detected users. If these two lists are the same, then SocialAwareFlicks continues playing movie trailers from the previously generated playlist. If the current list is different from the previous list, then SocialAwareFlicks has detected the group of users watching the screen has changed, and a new playlist is generated for the current group of users. Several options are being investigated for determining when to begin playing trailers from the new playlist. One option is to wait for the current movie trailer file to finish playing before beginning to play trailers from the new playlist. This would prevent abrupt changes to trailer playback when there is a lot of user movement by establishing some hysteresis. Due to a limitation in the current interface between SocialAwareFlicks and VLC, playback of the current movie trailer immediately terminates and trailers from the new playlist begin playing.

Figure 4.1 shows a screenshot of the current prototypes of the SocialAwareFlicks MC and SC. The MC for one user is running in the MicroEmulator window, while the SC has generated a movie trailer playlist for this user and invoked VLC to begin playing trailers from this playlist. VLC is playing the trailer for the film "A Scanner Darkly".

4.1.3 SocialAwareFlicks Performance Measurements

Performance metrics for SocialAwareFlicks were gathered for each functional subcomponent of the SocialAwareFlicks SC. All performance testing was performed using a Macbook Pro notebook running Mac OS X 10.5, with a 2.4 GHz Core2Duo processor, 4 GB of RAM, and a 6 Mbits/s cable Internet connection.

For the first round of performance testing, the execution time for each SocialAwareFlicks functional subcomponent was measured, which includes the subcomponents that search for available mobile devices running the SocialAwareFlicks MC (user discovery), query the Facebook API Web service, and query the Netflix API Web service. Testing was performed for one detected user with a Facebook profile listing five favorite movies, and execution of this test was repeated ten times. Figure 4.2 shows the average ex-

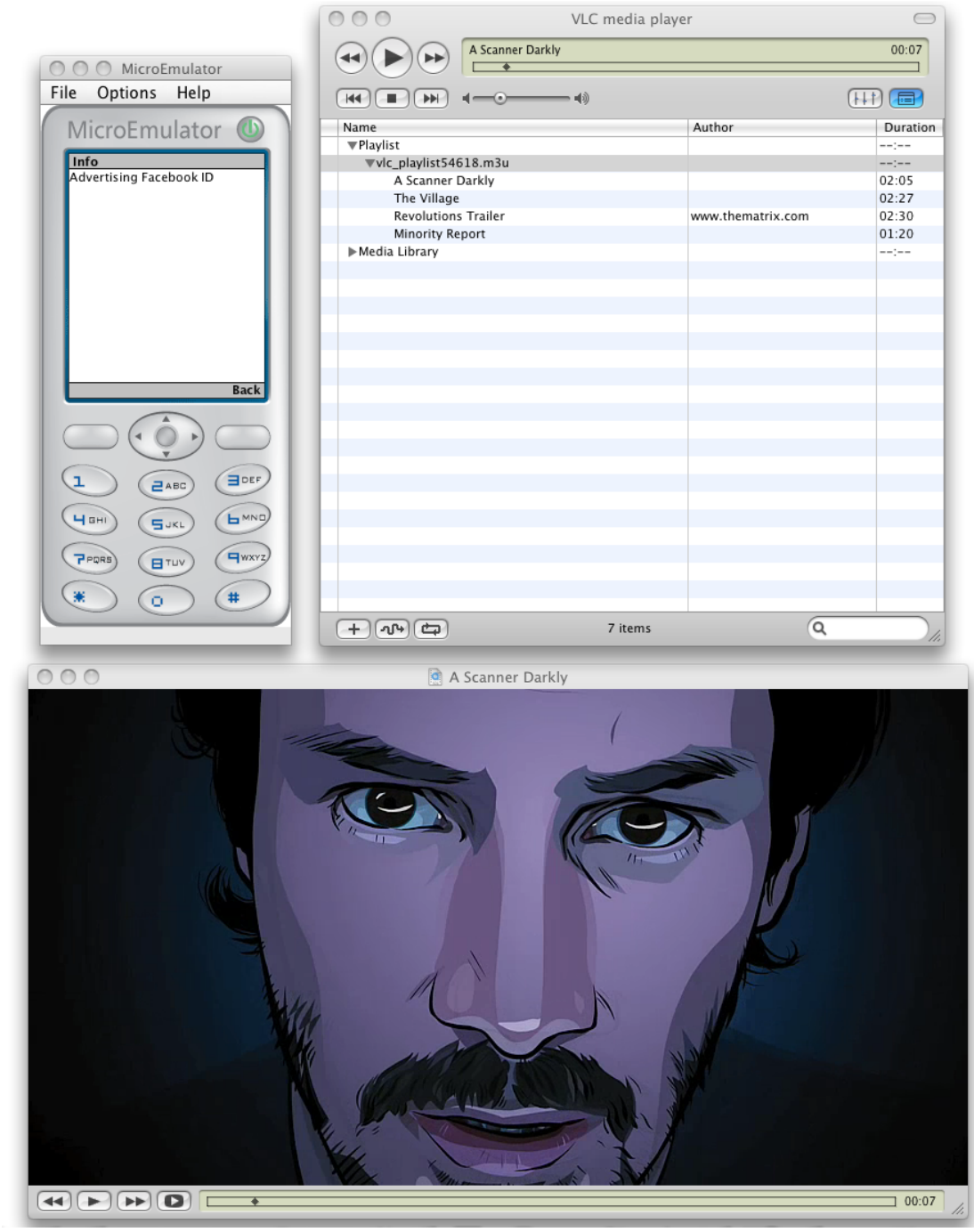

Figure 4.1: Screenshot of prototypes of the SocialAwareFlicks mobile and stationary components

ecution time for each subcomponent and the end-to-end execution time. End-to-end execution time was measured starting with the initial detection of the user by the SC and ending when the final movie trailer playlist is passed to VLC to begin playback. Execution time for the Netflix subcomponent accounts for approximately 74% of the total execution time for the SC. The current implementation of the Netflix subcomponent performs all operations in a serial fashion. Performing some of these operations in parallel, such as submitting concurrent HTTP requests to obtain list of similar movies for each of the user's favorite movies, may reduce the execution time of the Netflix subcomponent. Note that this testing was performed using the BlueCove JSR-82 Bluetooth emulator, which emulates all Bluetooth functionality in software. It is likely that the execution time for the user-discovery subcomponent would increase when testing is performed with real mobile devices and Bluetooth hardware.

The second round of performance testing involved measurements that show how the SocialAwareFlicks SC scales as the number of favorite movies for a user increases. Total end-to-end execution time was measured. Testing was performed for a favorite movie list ranging from one to ten movie titles, and each test was repeated five times. Figure 4.3 shows a graph of the average execution time for each favorite movie list used for this testing. The average execution time scales almost linearly as the favorite movie count increases. The average execution time increases linearly because the Netflix API query execution time increases linearly as the favorite movie count increases, and the execution time of the other SC subcomponents remains approximately constant.

4.2 SocialAwareTunes

4.2.1 Mobile Component

The current implementation of the MC in SocialAwareTunes is nearly identical to the MC implementation in SocialAwareFlicks. However, instead of using the

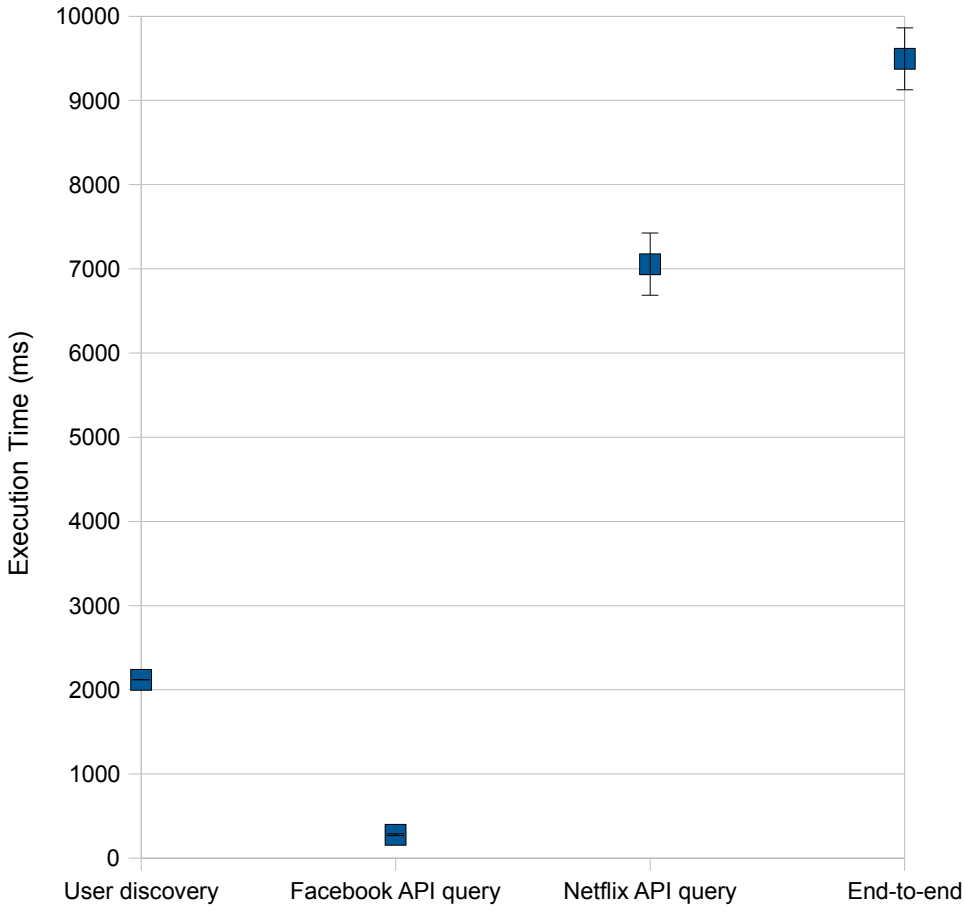

Figure 4.2: Execution time for each SocialAwareFlicks SC subcomponent

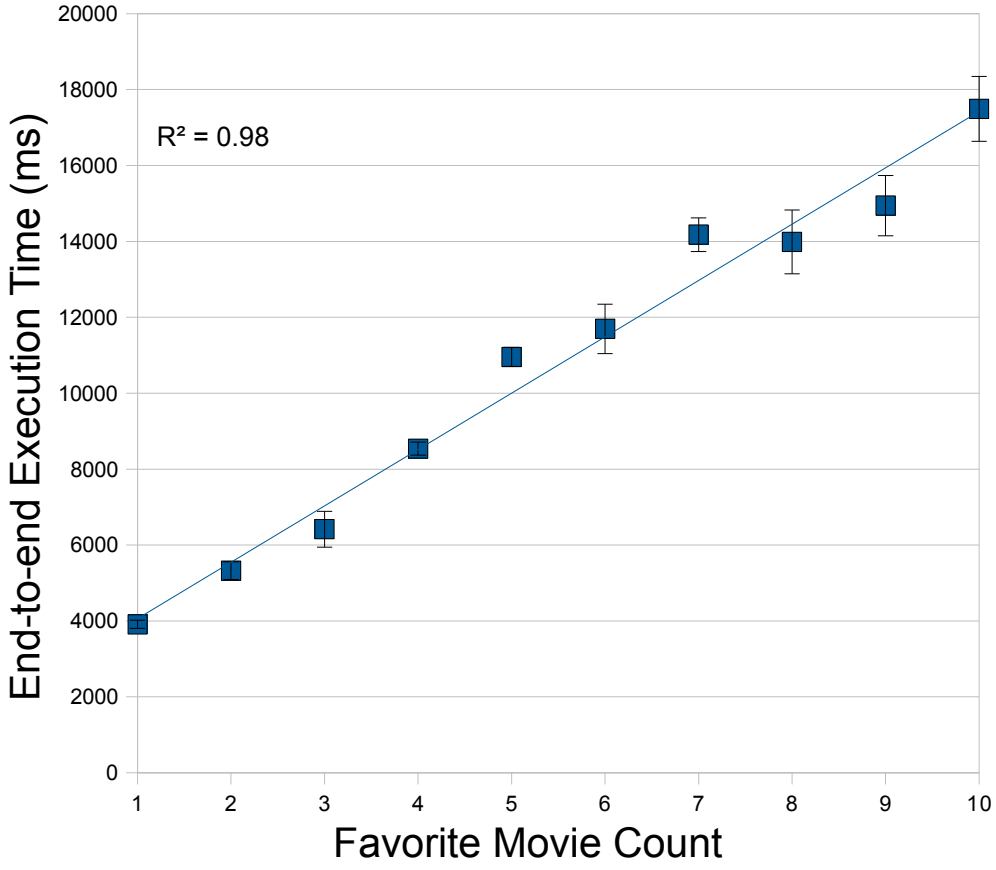

Figure 4.3: Scaling of SocialAwareFlicks SC as favorite movie count increases

SocialAwareFlicks Bluetooth service, the SocialAwareTunes Bluetooth service is advertised by the MC instead. In the current implementation, the only difference between these two Bluetooth services is that the SocialAwareTunes service uses a different UUID than the SocialAwareFlicks service.

4.2.2 Stationary Component

The SC in SocialAwareTunes is responsible for detecting the presence of nearby users within earshot of a set of speakers managed by the system and playing recommended music tracks based on the favorite music preferences expressed on each user's Facebook profile. The user detection functionality in the SocialAwareTunes SC is similar to that in the SocialAwareFlicks SC. The SC begins by searching for available mobile devices running the SocialAwareTunes Bluetooth service. Only nearby users within an approximate Bluetooth range of 10 meters will be detected. The SC registers several listener objects with the Marge framework to manage the search for available mobile devices and to manage the state of the connection to the mobile device. After detecting an appropriate mobile device, we query the device to determine the user's Facebook ID. In the current implementation, the user's Facebook ID is encoded as the Bluetooth service name of the SocialAwareTunes Bluetooth service running on the mobile device. We are able to identify the SocialAwareTunes Bluetooth service by looking for a particular UUID advertised by all mobile devices that run this service.

After obtaining the Facebook ID of each detected user, the SC constructs a playlist of recommended music tracks for each user. First the SC uses the Facebook REST API Web service to obtain the contents of the "favorite music" field on the user's Facebook profile. We assume that this is a comma-separated list of the user's favorite music artists. Next, the SC uses the Last.fm Web Services API [8] to obtain a list of the top five tracks for a given artist as determined by collaborative filtering technology on Last.fm. We assume the user will prefer to listen to the top tracks of his/her favorite

artist. If only one user was detected by the SC, then we simply concatenate the lists of top tracks for each of the artists to generate the final music playlist. If more than one user was detected by the SC, then the list of the top two tracks for each of the favorite artists for each user are concatenated to generate the final music playlist. After generating the playlist, the SC searches for available audio files on the local filesystem that correspond to the music tracks on this playlist. After attempting to find an audio file for each track on the playlist, the SC begins playing the audio files using the VLC media player.

The SC performs the search for available mobile devices every 30 seconds, although this search frequency can be adjusted. Just as in the SocialAwareFlicks SC, the current list of detected users is compared with the previous list of detected users. If these two lists are the same, then SocialAwareTunes continues playing music tracks from the previously generated playlist. If the current list is different from the previous list, then SocialAwareTunes has detected that the group of users watching the screen has changed, and a new playlist is generated for the current group of users. Just as with SocialAwareFlicks, several options are being investigated for determining when to begin playing tracks from the new playlist. One option is to wait for the current track to finish playing before beginning to play tracks from the new playlist. This would prevent abrupt changes to music playback when there is a lot of user movement by establishing some hysteresis. Due to a limitation in the current interface between SocialAwareTunes and VLC, playback of the current track immediately terminates and music from the new playlist begins playing.

Chapter 5

Future Work

My work on the SocialAware framework and the SocialAwareTunes and SocialAwareFlicks applications has raised a number of context-aware research issues. The straightforward implementation of SocialAwareTunes described in this thesis simply concatenates the top tracks of each favorite artist found for each discovered user. There has been much research into understanding what types of music a user would like based on user history and stated music preferences [26] [29]. These methods of automatically generating playlists based on user behavior and seed songs could be used to enhance the playlist generated by SocialAwareTuness. I could also use more sophisticated methods to create and manage changes to the playlist over time. For example, some policy could be developed to control how music is removed from the playlist over time. A user aging policy could be implemented so that tracks for newly discovered users are placed at the top of the playlist, while tracks for users discovered some time ago are placed at the bottom of the playlist. Tracks for previously discovered users that are no longer visible to the SC could be deleted after some elapsed time. There are also open questions about how to properly account for all of the musical tastes in a particular local group of users. If the musical preferences conflict for two or more users, is there a way to generate a playlist that is appropriate for these users? For example, if one user prefers music from rap artists, while another user prefers music from classical artists, how do we create a playlist that best accommodates these preferences if the user who enjoys rap dislikes

classical music?

Integrating sensing presence information provided by a system such as CenceMe [27] could enhance context-aware media applications like SocialAwareTunes and SocialAware-Flicks. For example, a system such as CenceMe could detect if a SocialAware user is currently engaged in a conversation with one or more nearby users. If this is detected, we may want to reduce the music playback volume or play a quiet instrumental music track to avoid interfering with the conversation.

I would like to conduct user-centered testing for the SocialAwareFlicks application. Such testing should reveal which of the single-user and group playlist generation heuristics described in this thesis are preferable for most users. This testing will also illuminate issues regarding the current user interface for SocialAwareFlicks. It it possible that users will find controls for adjusting volume and moving to the next or previous trailer in the playlist to be useful. The SocialAware framework permits such controls to be added to the application MC found on the user's mobile device.

The SocialAwareFlicks application could be enhanced to play more than one movie trailer concurrently on the shared screen watched by all nearby users. This raises several issues. How do we determine which movie trailers to play concurrently? Also, how do we determine which of the currently playing movie trailers should have audio enabled? Iterative user-centered design and testing will be required to answer these questions.

Chapter 6

Conclusions

As this thesis has shown, the SocialAware framework provides a solution for building social-networking-enabled context-aware services. The implementation of the SocialAwareTunes and SocialAwareFlicks multimedia applications demonstrate the feasibility of this framework and show how applications can adapt their behavior based on the identities and personal preferences of users in the immediate vicinity. These SocialAware applications reveal novel ways in which user preferences from an online social network can be used to tailor the local presentation of multimedia to co-located users. The development of these applications has raised a number of interesting questions to explore regarding context-aware services that leverage mobile social networks, and these questions point to some exciting opportunities for future work.

Bibliography

[1] Api - facebook developers wiki. http://wiki.developers.facebook.com/index.php/API.

[2] Blackberry facebook application. http://www.facebook.com/apps/application.php?id=2254487659.

[3] Bluecove jsr-82 emulator module. http://www.bluecove.org/bluecove-emu/.

[4] Brightkite. http://brightkite.com.

[5] Facebook developer resource. http://developers.facebook.com/resources.php.

[6] Facebook statistics. http://www.facebook.com/press/info.php?statistics.

[7] The java community process program - jsrs: Java specification requests - detail jsr 82. http://www.jcp.org/en/jsr/detail?id=82.

[8] Last.fm web services. http://www.last.fm/api.

[9] Linkedin professional network. http://linkedin.com.

[10] Loopt. http://www.loopt.com.

[11] Marge. https://marge.dev.java.net.

[12] Microemulator. http://www.microemu.org//.

[13] Movie trailer archive. http://robertkaye.org/trailers.

[14] Myspace. http://myspace.com.

[15] Netflix. http://www.netflix.com.

[16] Netflix api. http://developer.netflix.com.

[17] One billion bluetooth devices and growing. http://www.engadgetmobile.com/2006/11/27/one-billion-bluetooth-devices-and-growing/.

[18] Vlc media player. http://www.videolan.org/vlc.

[19] Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, and Alexander Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. ACM Transactions on Information Systems, 23(1):103–145, January 2005.

[20] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6):734–749, June 2005.

[21] Dustdar S. Baldauf, M. and F. Rosenberg. A survey on context-aware systems. International Journal of Ad Hoc and Ubiquitous Computing, 2(4):263–277, June 2007.

[22] A. Beach, M. Gartrell, S. Akkala, J. Elston, J. Kelley, K. Nishimoto, B. Ray, S. Razgulin, K. Sundaresan, B. Surendar, M. Terada, and R. Han. Whozthat? evolving an ecosystem for context-aware mobile social networks. IEEE Network, 22(4):50–55, July-August 2008.

[23] James Bennett and Stan Lanning. The netflix prize. In Proceedings of KDD Cup and Workshop 2007, August 2007.

[24] Anind K. Dey. Understanding and using context. Personal and Ubiquitous Computing, 5(1):4–7, February 2001.

[25] N. Eagle and A. Pentland. Social serendipity: Mobilizing social software. IEEE Pervasive Computing, 4(2), April-June 2005.

[26] B. Logan. Content-based playlist generation: Exploratory experiments. In Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR 2002), October 2002.

[27] E. Miluzzo, N. D. Lane, S. B. Eisenman, and A. T. Campbell. Cenceme - injecting sensing presence into social networking applications. In Proceedings of the 2nd European Conference on Smart Sensing and Context (EuroSSC 2007), October 2007.

[28] Mark O'Connor, Dan Cosley, Joseph A. Konstan, and John Riedl. Polylens: a recommender system for groups of users. In ECSCW'01: Proceedings of the seventh conference on European Conference on Computer Supported Cooperative Work, September 2001.

[29] E. Pampalk, T. Pohle, and G. Widmer. Dynamic playlist generation based on skipping behavior. In Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR 2005), September 2005.

[30] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In Proceedings of the Workshop on Mobile Computing Systems and Applications, December 1994.

[31] B.N. Schilit and M.M. Theimer. Disseminating active map information to mobile hosts. IEEE Network, 8(5):22–32, September-October 1994.

[32] A. Schmidt, M. Beigl, and H.-W. Gellersen. There is more to context than location. Computers & Graphics, 23(6):893–901, December 1999.

[33] Roy Want, Andy Hopper, Veronica Falcao, and Jonathan Gibbons. The active badge location system. ACM Transactions on Information Systems, 10(1):91–102, January 1992.