

# Privacy-aware Social Music Playlist Generation

Felix Beierle, Kai Grunert, Sebastian Göndör, Axel Küpper

Service-centric Networking

Telekom Innovation Laboratories / Technische Universität Berlin

Berlin, Germany

{beierle, kai.grunert, sebastian.goendoer, axel.kuepper}@tu-berlin.de

**Abstract**—Two of the most popular applications of smartphones are online social networking and playing back music. In this paper, we present the design and implementation of a prototype that combines those usages by creating an architecture that allows the generation and playback of group music playlists that are based on the musical taste of individual guests attending a meeting. In our architecture, we utilize automatically collected data on smartphones for the automatized generation of group music playlists. We follow the idea of utilizing context data in a preprocessing step to generate a group music profile for the recommendation process that generates a group music playlist. For designing such an architecture, we consider current discussions on privacy, data ownership, and data control.

## I. INTRODUCTION

Nowadays, the smartphone is a highly potent computing device that users carry with them at almost all times. It is often used for media consumption, especially listening to music [1]. The smartphone is a highly personalized device with usually just one single user. At the same time, the smartphone is also a very social device, as it is also often used for online social networking, messaging, sharing pictures, etc. We believe that those two aspects can be combined. The tracking of individual smartphone usage can be leveraged for group scenarios. In the case of listening to music, individual taste can be recorded on the smartphone and used in group scenarios to create a group music playlist, considering the taste in music of each group member that is currently present at the same location.

To infer the musical taste of a single user, context data can help with filtering mechanisms: music listened to in a specific context might not be relevant when creating a taste model of a group. Previous work exists regarding the collection of context data in connection with data about music users listened to [2], [3]. The prototypically implemented applications are mostly based on explicit annotations by the user. Often criticized is the effort that is necessary when manually annotating such data. In the last years, the number and quality of physical sensors in a smartphone as well as the availability of public APIs and SDKs has substantially increased. These developments enable the user and her device to easily and automatically determine context information with respect to a multitude of aspects. At the same time, when designing mobile applications with increased sensor usage and querying of APIs, battery consumption has to be taken into account.

With these developments, we are able to automatize most of the general workflow of our group music playlist generation system. Combining the possibility to automatically determine

the user's context and utilizing the smartphone for social group scenarios, we envision meeting scenarios – formal or informal gatherings of groups of people, with music playing, e.g., a party: Users permanently and unobtrusively collect music and context data on their smartphones. When gathering for a meeting the collected data is consolidated, pre-processed, and a recommender system can generate a music playlist based on the taste of all attending guests. Current technology already allows for a very high degree of automation in such a system. Furthermore, our system offers the possibility to research other areas, such as group recommendations in general or context-based recommender systems in specific.

When designing such a group music playlist generation system, and collecting music playing data and context data, we are dealing with private data about the user, e.g., her location. Merriam-Webster defines "privacy" as "the state of being apart from company or observation" and "freedom from unauthorized intrusion." [4] In the context of web and mobile applications, we regard an application as privacy-aware when the user has the ability to choose what information she wants to share with other users or with service providers. Recent reports (e.g., [5]) can give us insight into systematically approaching the design of a privacy-aware system. In general, we argue that distributed systems, which allow users to choose their service provider or set up their own host, in general can provide more privacy and help better ensure the concept of data ownership.

We designed a highly automatized system collecting data about music that was played and enriching it with context data, all without user interaction. This data is kept locally on the mobile device. The user can analyze it and decide what parts she wants to transmit it to a server. We envision a meeting scenario, where she can decide to send (parts of) her data to a meeting host to help generate a group music playlist for all meeting guests.

Our main contribution is the design and implementation of the vision of highly automatized computing in the social group scenario of music playlist generation, and designing such a system with the concepts of privacy, data ownership, and data control in mind. After discussing related work, we present our prototype, evaluate it, conclude, and point out future work.

## II. RELATED WORK

In this section, we refer to related work in the fields of privacy, data ownership, and data control, as well as context data and music-related applications that utilize context data.

### A. Privacy, Data Ownership, and Data Control

So far, such group scenarios are most commonly realized with services like Facebook. Especially in such online social network scenarios, users have major privacy concerns. The users' data might be used as the basis for a service operator's business model, while users did not intend to share their data to that extent [6]–[8].

In Europe and especially Germany, discussions about privacy and data ownership are very prominent. In a recent report, the concept of data ownership and data control was thoroughly discussed with respect to cloud services [5]. The idea is that a person's private data should always be available for her so that she can access and change it. Servers that store the data should use encryption and deal with the data confidentially. Another related concept is the one of data frugality. Only data that will be used for a specific purpose should be collected, it should only be stored for that purpose, and not be used for other purposes [5]. In the end, there also has to be made some compromise between usability and security.

Another concept to enhance user privacy is the combination of data separation and data frugality [9]: transmitting only the data that is required to use a service as well as using different service providers for different services bring the advantage that the multitude of external service providers cannot derive a holistic profile of a user. In the context of application development, this gives the requirement to build software just for a specific task with well-defined boundaries and to restrict the data collection to just the data needed for the specific task.

### B. Context Data

Context is something that characterizes an entity that is relevant to the user or application [10]. Context thus is application-dependent: For an indoor mobile tour guide for example, weather should not be considered as context [10].

Yurur et al. conclude that privacy and security are open issues, and that users can feel constantly monitored by apps tracking context information [11]. Regarding the acquisition of context data, the authors divide sensors into three categories:

- *Physical Sensors* capture physical data, e.g., GPS for location or accelerometer for activity
- *Virtual Sensors* from software application and/or service, e.g., manually set location or computation power of the device
- *Logical Sensors* are a combination of physical and virtual sensors with additional information through various sources like databases or log files

The authors divide context into four categories:

- *Device Context*: net connectivity, communication cost
- *User Context*: profile, geographic position, neighbors, social situation
- *Physical Context*: temperature, noise level, light intensity, traffic conditions
- *Temporal Context*: day, week, month, season, year

Using Dey's definition of context, for a highly automatized music application, context is anything that might influence the

user's choice of music. Using Yurur's definition, the music listened to can be viewed as a context for other applications, music listening then being, e.g., part of the *user context*. Thus, the part of our application that tracks the music listened to can effectively offer context data as a *logical sensor*.

### C. Music Applications and Context Data

*Flytrap* is described as a "group music environment" [12]. The music users listen to on their computer is tracked. Utilizing RFID badges, the presence of other users is detected. A group playlist is generated and the next track is decided by a voting mechanism. Since the publication of the paper in 2002, technological advances and especially the advent of smartphones allow for more complexity and automation.

*SocialFusion* follows the idea of collecting context data from multiple sources: online social networks, mobile phones and nearby sensors [13]. For *SocialFusion*, one of the major challenges described is the mining of data that is collected. The general process is collecting any available data and analyzing/mining that data for relevant information later, depending on the used application. *SocialFusion* furthermore follows the idea of individual and group recommendations based on the results that the data mining outputs. The key difference in our approach is that, instead of mining existing data that might be related, we specifically collect data in order to use it in a recommender system that creates group music playlists.

*Mobile Music Genius* aims to be a music player on a mobile device that automatically chooses a song according to the user's context [14]. Here, context is used to predict the taste of a single user, instead of for a group of users.

Baltrunas et al. deal with the prediction of the best context for listening to a particular song [2]. For this, context data is collected explicitly with a graphical user interface. The user enters a rating for the song, activity, weather, mood, and most suitable time for listening to the song. In our application, all of that context data (with the exception of mood) is collected unobtrusively in the background and without disrupting the regular listening experience. In another paper about the collection of context data [3], activity and mood had to be annotated by the user, while some other context data was collected by the sensors of Android smartphones.

In [15], we presented the concept of using context data to create an additional layer in online social networking scenarios in order to connect people with similar contexts. The implementation presented in this paper follows the concept of obtaining context data on smartphones. This collected data could be used to connect people with similar tastes in music.

## III. DESIGN AND IMPLEMENTATION

In this section, we detail the requirements for our system before describing the role model. We then describe the components of our architecture and explain the three main processes.

### A. Requirements

As motivated in the introduction with the discussions about privacy and data ownership, a general requirement is (r1)

privacy. In order to discuss privacy for our scenario, in the following, we will introduce the role model of our system. To materialize the vision of highly automatized ubiquitous computing, our second general requirement is to have a high degree of (r2) automation that allows us to keep the necessary user interaction to a minimum. There is another requirement especially for the mobile component: the application should (r3) not significantly drain the battery, which is in general an important aspect for any mobile application.

### B. Role Model

In Figure 1, we give the role model of our system, including data flows. In the center, there is the meeting host. The meeting guest sender is the role of an attending guest that sends her music and context data. The meeting host uses three types of external providers. To the *music metadata service provider*, she sends music data in order to receive meta-data about the music, e.g., genre. To the *music playlist recommendation service provider*, the meeting host sends a group music profile that she created. By utilizing a recommendation service, the host ensures to create a playlist containing new music previously not listened to by the attendees. To the *music player service provider*, she sends the (post-filtered) group music playlist she received from the *music playlist recommendation service provider*. The meeting guest receiver is an attending guest in the role of a listener of the played music. By hearing the music, she can know the generated playlist.

In the introduction, we defined privacy as the ability of the individual to choose what information she wants to share with other users or with service providers. One part of privacy is the data a user actively shares with another user or service provider. Another aspect concerning privacy is when individual users are (unintentionally) identifiable (or traceable) through data they share. To encompass all potential aspects and data flows, our role model will help with the evaluation of the privacy of our system.

### C. Mobile Component

Figure 2 depicts the components of the system. On the top, there are multiple users with their smartphones. The installed meeting application contains one main component – the *Data Collection Engine*. It is responsible for the unobtrusive and continuous collection of music information of played songs. Additionally, in the background, it accesses different sensors and webservice to obtain and store context data.

There are two types of collected data: music data and context data. Music data comprises the artist, track, and album. This data is generally enough to uniquely identify a song. Other music meta-data like genre, tempo, etc. can be acquired later. The other collected data describes the context in which a user listened to a song. Using the taxonomy of [11], we consider the *device context* somewhat uninteresting, as net connectivity and communication cost might not significantly influence the choice of music. Regarding the *user context*, the geographic position is tracked. Neighbors or social situations could later be inferred when the data is sent to the server

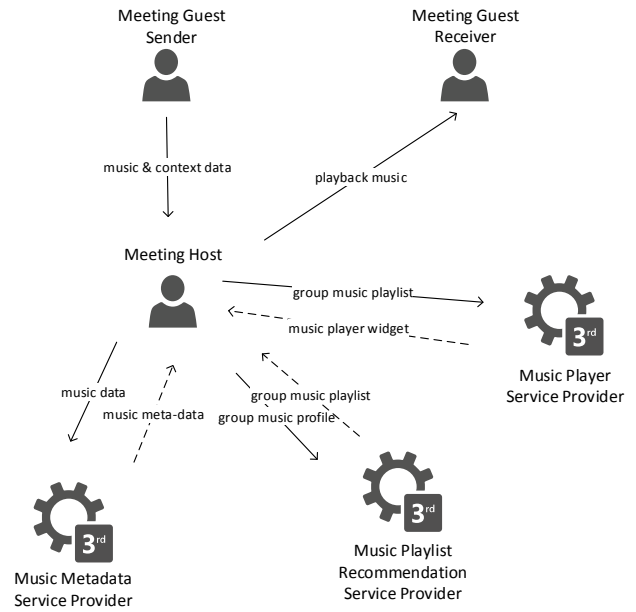


Fig. 1. Role model of the system

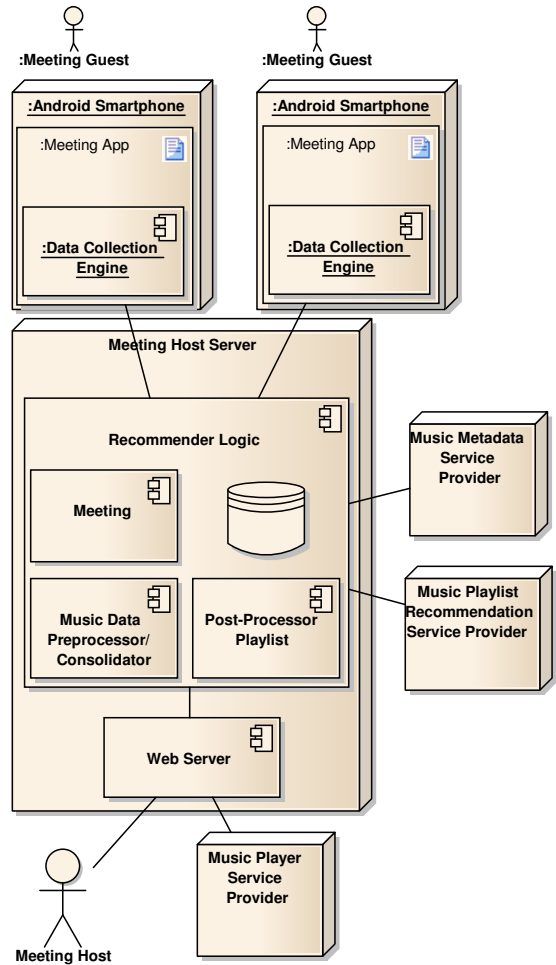


Fig. 2. Components of the architecture

component. Regarding the *physical context*, we prototypically implemented the collection of current activity<sup>1</sup> and weather<sup>2</sup>, data that related work also considered of importance with respect to the choice of music [2], [3]. For the physical activity, the most likely state returned by the SDK is stored. The possible states are: sedentary, walking, running, biking, in car, random, none. As for the *temporal context*, the date and time a song was played is tracked.

#### D. Server Component

In the center of Figure 2, there is the server component. This component is used to create meetings, automatically generate group music playlists and to offer the possibility to play back music. A web server provides a website to the host. The website is the interface to the meeting host for creating meetings and playing back music.

The *Recommender Logic* module is responsible for creating a group music playlist out of the attendees' data. We view the process of generating a group music playlist as a recommender system task. In general, for such a group recommendation task, there are generally two approaches [16]:

- consolidate profiles into one group profile
- consolidate individual recommendations to one group recommendation

Using an external music recommendation system, the first option offers the individual guest more privacy: when the individual profile data is preprocessed and consolidated in order to generate a group music profile, only the created group music profile has to be given to the music playlist recommendation service provider. The individual user is thus less identifiable or traceable.

#### E. External Services

In our prototypical implementation, we used three types of external service providers. As a *Music Metadata Service Provider*, returning, e.g., genre of a music track, we used Gracenote<sup>3</sup>. As a *Music Playlist Recommendation Service Provider*, for generating a playlist from a group music profile, we used The Echo Nest<sup>4</sup>. The music player is integrated in our website via JavaScript, in our prototype we used Deezer<sup>5</sup>.

#### F. Processes

This section explains the three main processes of the system: the meeting creation, the registration, and the meeting itself. Furthermore, the straightforward process of collecting music and context data on the mobile device is the foundation for the generation of the collaborative playlist. An extension to the current implementation could be to have an on/off switch for tracking and to offer some filtering that disables tracking for certain times, locations, contexts, or genres.

<sup>1</sup>Via the Intel Context Sensing SDK, <https://software.intel.com/en-us/context-sensing-sdk>

<sup>2</sup>Via OpenWeatherMap, <http://openweathermap.org/>

<sup>3</sup><http://www.gracenote.com/>

<sup>4</sup><http://the.echonest.com/>

<sup>5</sup><http://www.deezer.com/>

For the meeting creation, the meeting host installs the software on her own home server. In the web front-end, the host can specify date, location and (optionally) a music genre of the meeting. A website is generated for the meeting, which contains a music player widget from an external provider, which at the time of the meeting will include the continuously updated group playlist. Furthermore, the website contains a QR code for distributing to potential meeting guests.

In the registration process, the meeting guest scans the QR code which is a URL of the REST interface of the meeting host server. The application uses this URL to request some information from the server, like date, location, and genre of the meeting. The user then decides if she is interested in the meeting. She can decide to not inform the meeting host and not send any data. This would be the most private setting, but the attendee would not be able to participate in the creation of the group playlist. If the guest decides to attend and inform the meeting host, she can decide what data – if any at all – she later wants to send to the meeting host. The implementation so far allows for a manual song based filtering that allows the user to remove tracks that she does not want to send to the host. An extension to that would be to enable the user to filter music and context data by location, context, genre, or time to remove unwanted entries more easily. When deciding to send any data to the meeting host, the guest's device receives a unique user id (unique for the event) in order to later avoid guests sending their data multiple times.

Afterwards, when being registered for a meeting, a geofence is activated in the system's location API, so the application finds out automatically if the guest is at the meeting location. Geofences are virtual boundaries for geographical areas and are used to check whether a device is entering, dwelling inside, or exiting such an area [17], [18]. The geofence is derived from the location of the meeting and a radius the host indicated when setting up the meeting. Henceforth, the system continuously checks if the user has entered the geofence. The general process of a meeting is illustrated in the BPMN notation [19] in Figure 3. Originally created for business processes, we believe BPMN is also a useful generic tool to describe software processes and interactions between users and service providers. This geofencing process is shown as the expanded sub-process "Meeting-entering scanning" in the top left of Figure 3. If the user enters the geofence of a meeting, the application verifies that the time and date are correct. If not, the process jumps back and continues checking for the geofence. If the location and date correspond to the meeting information, the application notifies the meeting host server and sends the filtered music and context data. On the user side, the process continues with scanning, in order to check if the guest leaves the meeting. If this happens, the application notifies the server.

When the meeting host server receives attendees' data, the intermediate message event "Music&context data" attached to the boundary of the data reception activity is called and leads to the process for creating the collaborative playlist. The process starts with the preprocessing of the data. It is depicted

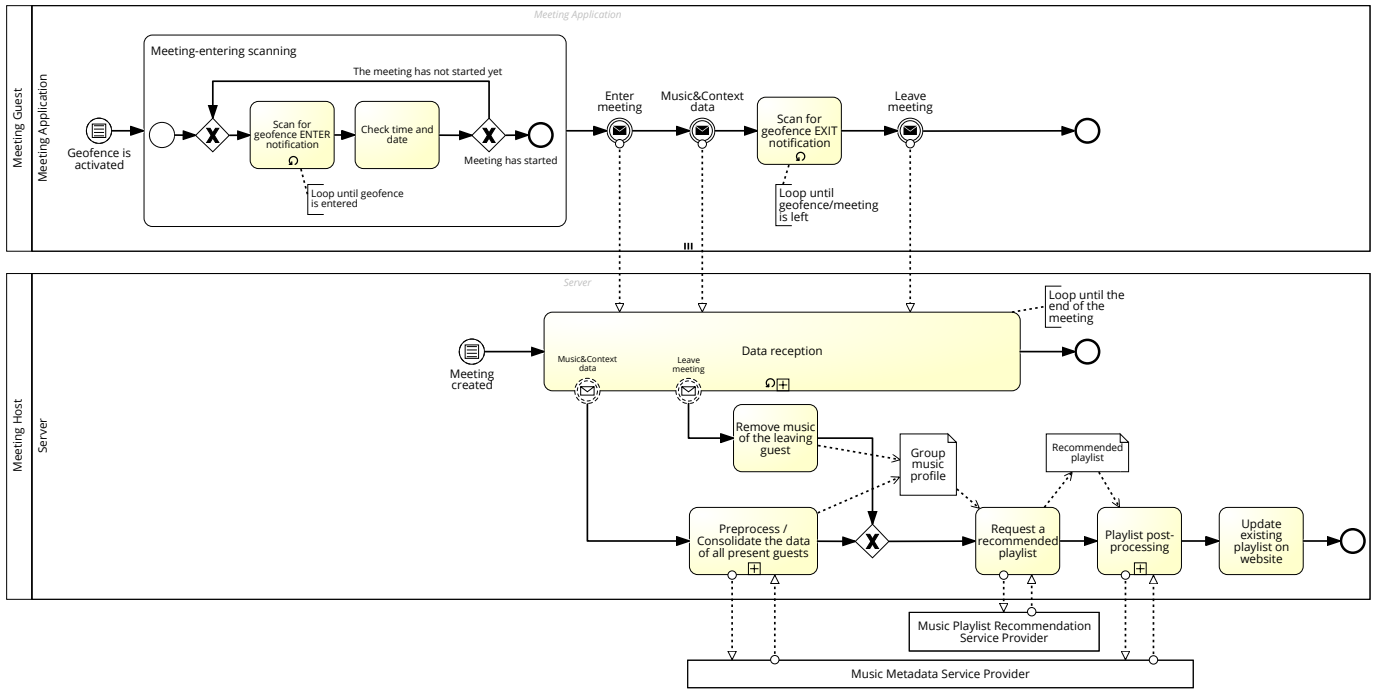


Fig. 3. The general process of an event.

as a collapsed sub-process, observable by the plus sign inside the small square at the bottom center of the activity. Multiple steps are executed:

- *Augmentation* of the music data with useful meta data (e.g. genre).
- *Normalizing* the music data of one guest over all attendees: the server has to limit the influence of users with a very high amount of played tracks and a high track count in relation to a user with lower numbers.
- *Filtering/Weighting* of the music data. Here, the context data and the music metadata is used to filter and weigh music to, e.g., discard music listened to early in the morning or while exercising.
- *Consolidation* of the music data of every guest to a build a *group music profile*. The result is a list that represents the preprocessed music data of all meeting attendees.

The server uses the resulting *group music profile* in the next step and hands it over to an external playlist recommendation service provider, which returns a *recommended playlist*. To avoid playing the same song or artist twice in a short amount of time, this playlist is post-processed. When this activity is finished, the website is updated, so the host is able to play the recommended songs based on the taste of the current attendees.

Looking at the "Data reception" activity in Figure 3, we can see that the process for creating the recommended playlist is permanently repeated for new guests that enter the meeting. The attached "Music&Context data" message event is non-interrupting (denoted by the dashed circles around the event). It creates a new parallel process which results in the generation

of the playlist. The "Data reception" process is continued until the loop ends, i.e., when the meeting is over.

When a guest leaves the meeting, the message event "Leave meeting" (attached to "Data reception") is triggered. The started process flow removes the guest's music data from the *group music profile*. Afterwards, the process flow is merged with the process for generating the recommended playlist.

#### IV. EVALUATION

In this section, we evaluate our system according to the requirements we established.

##### A. Requirement 1: Privacy

Starting from our definition of privacy, in designing our system, we made sure that the user can choose what data about her is shared with other users and service providers. Looking at the meeting guest and her mobile application, we first examine the music and context data collection: when collecting context data, a location or weather provider will most likely not be able to distinguish between regular querying of location and weather data by other applications – especially because location and weather is highly typical data that is frequently queried. The collection of data of physical sensors and music data (assuming playing locally available music files) is completely done offline.

Looking at Table I, the "Guest Sender" role as the attending guest that sends data to the meeting host has all music data and context data. She can generate filtered lists of both those data sets before sending.

We designed our system so that the meeting host can be anybody that wants to host a meeting. She installs the server

component software and can start to create meetings. By this design, we generally assume a high degree of trust between meeting host and meeting guests. The meeting host receives the attendee’s music and context data. She communicates with three external service providers. The context data remains with the host and are not send to any service providers.

The first service provider contacted is the music metadata service provider (MMSP in Table I). The filtered music data is sent to the server in order to receive additional information about the music, e.g., genre. By receiving the data, the service provider can infer something about the musical taste of the attending guests. To improve the privacy of host and guests, a local database containing music information could be introduced to remove the need to query an external service. The second service provider contacted is the music playlist recommendation service provider (MPRSP in Table I). Here, the meeting host sends the locally created group music profile. By sending a group music profile, for the service provider, it is indistinguishable if the request originated for one user or for a group. Individual users will most likely not be traceable in such group music profiles. The service provider can (and is supposed to by the logic of our system) infer the musical taste by this profile and returns a recommended playlist. To have more privacy for host and guests, a local recommender system could be implemented, although it might be hard to compete with services like The Echo Nest. The third service provider contacted is the music player service provider (MPSP in Table I). The meeting host sends the post-processed playlist. By offering music streaming for the requested songs, the music player service provider knows the musical taste of the group. In order to increase the level of privacy for host and guests here, only locally available music files could be played.

From the perspective of the "Guest Sender" role, by looking at the checked boxes in Table I, the privacy implications are indicated. The host has to be trusted with the filtered music and context data. The privacy implications of the three service providers are described above. The last step in the process of hosting a meeting is the actual playback of the music. At this step, the "Guest Receiver" role "receives" the playlist by listening to it. This is another privacy implication to be considered by the Guest Sender, as she might want to hide her musical taste. In the general case of a meeting with several attendees, it will be virtually impossible to make out a correlation between a particular song or taste in music and an individual guest. The higher the number of attendees, the more unlikely it is to make out such correlations. For very low numbers of attending guests, there are some edge cases. If very few attendees are present and a guest is arriving or leaving, the playlist should not change immediately in order to not give an indication of the individual’s taste in music that could be considered private. To avoid this problem, the process of updating the playlist can be done in bulk after several people entered/left, in order to hide an individual’s taste in a group of people. If there are zero (or one, if the host is also a guest) attendees, the genre setting of the meeting can still make it possible to play back music.

Overall, the employed mechanisms of filtering data on the smartphone before sending, sending limited data to service providers and considering edge cases with few attendees leave the user in control of her data and avoid unintentional sharing of private data. In future work, the proper way of presenting the user the information given in Table I will be researched. To lower the level of trust needed towards the meeting host, some parts of the preprocessing of the data could be done on the smartphone. This way, the meeting guest’s privacy could be improved because she does not send the list of music tracks listened to with associated context data, but some form of music profile that models her taste.

TABLE I  
OVERVIEW OF ROLES AND THEIR DATA ACCESS

Data \ Role	music data	filtered music data	context data	filtered context data	group profile	playlist
Guest Sender	x	x	x	x		
Guest Receiver						x
Host		x		x	x	x
MMSP		x				
MPRSP					x	
MPSP						x

MMSP: Music Metadata Service Provider  
MPRSP: Music Playlist Recommendation Service Provider  
MPSP: Music Player Service Provider

### B. Requirement 2: Automation

Looking at the mobile component, the collection of music and context data is done without any user interaction. The user just uses her music player application.

The meeting host just has to create a meeting and can use the automatically created QR code to invite meeting guests. At the meeting itself, she just needs to start the music player widget on the automatically generated meeting website.

For attending a meeting, the meeting guest has to trigger the QR code scan. After receiving the meeting information from the server, she can decide whether she wants to attend and create a filtered list of music and context data to send. Choosing what data to send is a necessary manual step in order to assure the guest’s privacy. Without considering privacy, this step could be automated and every user would send their data.

The meeting itself can then run fully automatically. Coming guests are automatically considered and leaving guest disregarded in the group music playlist.

### C. Requirement 3: Battery consumption

The mobile application is used in two processes: tracking music and context and sending data to the meeting host. The latter process is executed only occasionally. The tracking of music and context is run permanently when listening to music, so we focus on this process in this evaluation. We used an LG Nexus 5, running stock Android 5.1.1 with only the stock

Google applications and an additional file explorer installed. We collected two sets of data. In both cases, during the whole run, we turned off the display and let music play continuously. The first dataset tracked the battery life without using our application: 19.5 hours. For the second dataset, we let our mobile application run. The battery lasted 19.3 hours. Our evaluation shows that the usage of our application made the battery drain around 1 percent faster. Our interpretation of this result is that the power consumption by playing back music is already so high that the tracking of music and context data does not carry a lot of additional weight. Furthermore, in real world scenarios, additional applications that cause traffic and notifications, like email, messaging, or online social network applications, will most likely make the additional battery drain by our application unnoticeable.

## V. CONCLUSION AND OUTLOOK

In our paper, we combined the ideas of the smartphone being both a highly individualized device, as well as a device for social interaction. We designed and implemented a group music playlist generator that continuously creates group music playlists based on group music profiles for guests attending a meeting. We use the smartphone to track individual musical tastes. We evaluated the privacy, automation and battery consumption of our system. When designing our system, we took into consideration current discussions about privacy, data ownership, and data control. We keep the data collected to a minimum with respect to the application it is used for. We also detailed further possible improvements to advance the level of privacy even more. The level of automation is already very high and further automation might only be possible by compromising privacy. For the mobile application, we evaluated that the additional battery consumption is negligible. The design of our system might give hints on how to develop other privacy-aware applications or group recommender systems.

Future work includes evaluating the pre-processing and consolidation step for generating the group music profile to quantify the quality of the generated playlists. Furthermore, the collected data could be used for other features like displaying who is attending the party or for visualizing a map which displays where which music is popular. Current technologies also allow for even more automation: Bluetooth Low Energy beacons could be used for accurate indoor positioning of meeting guests in order to automatically infer the atmosphere of the party. The smartphone accelerometer could be utilized for recognizing dancing moves to additionally impact the social music playlist creation. Further future work is to enhance the concept of current online social networks by, e.g., automatically connecting people with similar tastes in music.

## ACKNOWLEDGMENT

This work has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 645342, project reTHINK and from project DYNAMIC<sup>6</sup> (grant No 01IS12056), which is funded as part of





the Software Campus initiative by the German Federal Ministry of Education and Research (BMBF). The authors would like to thank Jonas Düver, Paul Mälzer, Aleksey Perevosnikov, and Raed Ben Younes for their substantial work regarding the implementation of the prototype.

## REFERENCES

- [1] A. Smith, "U.S. Smartphone Use in 2015," <http://www.pewinternet.org/2015/04/01/us-smartphone-use-in-2015/>, Accessed 2015-10-07.
- [2] L. Baltrunas, L. Rokach, M. Kaminskas, B. Shapira, F. Ricci, and K.-H. Luke, "Best Usage Context Prediction for Music Tracks," in *in Proceedings of the 2nd Workshop on Context Aware Recommender Systems*, 2010.
- [3] Y.-C. Teng, Y.-S. Kuo, and Y.-H. Yang, "A large in-situ dataset for context-aware music recommendation on smartphones," in *2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*. IEEE, Jul. 2013, pp. 1–4.
- [4] Merriam-Webster, "privacy," <http://www.merriam-webster.com/dictionary/privacy>, Accessed: 2015-10-07.
- [5] P. Bosesky, P. H. Deussen, A. Quandt, S. E. Schulz, and L. Strick, "Datenhoheit in der Cloud," Fraunhofer Fokus, Berlin, Tech. Rep., 2013. [Online]. Available: <http://publica.fraunhofer.de/dokumente/N-281950.html>
- [6] M. Falch, A. Henten, R. Tadayoni, and I. Windekilde, "Business models in social networking," in *CMI Int. Conf. on Social Networking and Communities*, 2009.
- [7] A. Datta, S. Buchegger, L.-H. Vu, T. Strufe, and K. Rzadca, "Decentralized Online Social Networks," in *Handbook of Social Network Technologies and Applications*. Springer, 2010, pp. 349–378.
- [8] A. Bleicher, "The anti-Facebook," *IEEE Spectrum*, vol. 48, no. 6, pp. 54–82, 2011.
- [9] C. Wilson, T. Steinbauer, G. Wang, A. Sala, H. Zheng, and B. Y. Zhao, "Privacy, Availability and Economics in the Polaris Mobile Social Network," in *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications*, ser. HotMobile '11. New York, NY, USA: ACM, 2011, pp. 42–47.
- [10] A. K. Dey, "Understanding and Using Context," *Personal Ubiquitous Comput.*, vol. 5, no. 1, pp. 4–7, Jan. 2001.
- [11] O. Yurur, C. Liu, Z. Sheng, V. Leung, W. Moreno, and K. Leung, "Context-Awareness for Mobile Sensing: A Survey and Future Directions," *IEEE Communications Surveys Tutorials*, vol. PP, no. 99, pp. 1–28, 2014.
- [12] A. Crossen, J. Budzik, and K. J. Hammond, "Flytrap: intelligent group music recommendation," in *Proceedings of the 7th international conference on Intelligent user interfaces*. ACM, 2002, pp. 184–185.
- [13] A. Beach, M. Gartrell, X. Xing, R. Han, Q. Lv, S. Mishra, and K. Seada, "Fusing mobile, sensor, and social data to fully enable context-aware computing," in *Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications*. ACM, 2010, pp. 60–65.
- [14] M. Schedl, G. Breitschopf, and B. Ionescu, "Mobile Music Genius: Reggae at the Beach, Metal on a Friday Night?" in *Proceedings of International Conference on Multimedia Retrieval*. ACM, 2014, pp. 507–510.
- [15] F. Beierle, S. Göndör, and A. Küpper, "Towards a Three-tiered Social Graph in Decentralized Online Social Networks," in *Proceedings of the 7th International Workshop on Hot Topics in Planet-scale mObile computing and online Social neTworking*, ser. HotPOST '15. ACM, 2015, pp. 1–6.
- [16] S. Berkovsky and J. Freyne, "Group-based Recipe Recommendations: Analysis of Data Aggregation Strategies," in *Proceedings of the Fourth ACM Conference on Recommender Systems*, ser. RecSys '10. ACM, 2010, pp. 111–118.
- [17] U. Bareth, A. Küpper, and B. Freese, "Geofencing and Background Tracking - The Next Features in LBS," in *Proceedings of the 41th Annual Conference of the Gesellschaft für Informatik e.V. (INFORMATIK 2011)*, vol. 192. Berlin, Germany: Köllen Druck + Verlag GmbH, Oct 2011.
- [18] S. Rodriguez Garzon and B. Deva, "Geofencing 2.0: Taking Location-based Notifications to the Next Level," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '14. ACM, 2014, pp. 921–932.
- [19] O. M. Group, "Business process model and notation (bpmn), version 2.0.2," Available at: <http://www.omg.org/spec/BPMN/2.0.2/>, 2013.

<sup>6</sup><http://www.dynamic-project.de>

## Additional Information

<b>Bibliographic Data</b>	F. Beierle, K. Grunert, S. Göndör, and A. Küpper, "Privacy-aware Social Music Playlist Generation," in <i>Proc. 2016 IEEE International Conference on Communications (ICC)</i> . IEEE, 2016, p. 5650–5656.
<b>Pre-print from</b>	<a href="https://beierle.de">https://beierle.de</a>
<b>Online at</b>	<a href="http://dx.doi.org/10.1109/ICC.2016.7511602">http://dx.doi.org/10.1109/ICC.2016.7511602</a>
<b>Authors</b>	Felix Beierle    Kai Grunert  Sebastian Göndör    Axel Küpper
<b>BibTeX</b>	@inproceedings{Beierle2016ICC, title = {{Privacy-aware Social Music Playlist Generation}}, author = {Beierle, Felix and Grunert, Kai and G\"ond\"or, Sebastian and K\"upper, Axel}, booktitle = {{Proc. 2016 IEEE International Conference on Communications (ICC)}}, publisher = {{IEEE}}, year = {2016}, pages = {5650—5656}, doi = {10.1109/ICC.2016.7511602} }
<b>Copyright Note</b>	IEEE Copyright Notice Copyright (c) 2016 IEEE  Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.