Cupeer: An Offline Socializing Application Using Mobile Peer-to-Peer Connectivity

Jaeseok Huh*
KAIST School of Computing
jshuh@kaist.ac.kr

Adrian Steffan*
Technische Universität München
adrian.steffan@tum.de

ABSTRACT

In the digital era, over three billion people use social network services (SNS) to cultivate relationships with others every day [1]. However, most services have been criticized for ignoring the well-being of users while maximizing the profits within the platform. In this paper, we propose *Cupeer*, an offline socializing application that uses peer-to-peer (P2P) connectivity and a decentralized, user-driven matching algorithm. We discuss its design choices and prototype implementation and the result for our user survey. Our evaluation shows a high level of user satisfaction with *Cupeer* and a promising future for other directions including online dating. We believe that users' growing awareness of privacy will give rise to services like *Cupeer*. We made source code available at https://github.com/iriszero/Cupeer [23].

CCS CONCEPTS

• Network > Network Services

KEYWORDS

P2P, Google Nearby Connection, Android, Online Dating

1 INTRODUCTION

Social interaction has been an essential part of homo sapiens' life. In the digital era, the proliferation of mobile devices has given rise to social networking services (SNS) to provide them with better ways of maintaining relationships and socializing with others. However, most of their existing services fail to meet their ultimate goal—to offer a quality user experience. In the capitalist world, the service providers have rather prioritized the needs of advertising parties over the healthy social life of the users and taken the opportunity to collect user data on a massive scale. As such, it is no surprise that people are struggling with interacting well with others. A recent survey by Evite [2] revealed that, in the United States, 45% of adults have difficulty in finding new friends, citing introversion or shyness as the main reasons. Similarly, the online dating services are incentivized to hold their users attached to the platform as long as possible, rather than providing a service their nominal goal—matching the users—which inevitably leads to the loss of membership.

Given their conflict of interest, human society cannot rely solely upon the existing companies and their products in order to facilitate and pursue genuine social interactions. To make matters worse, they often use a black-box, centralized algorithm for "recommendation" or "matching", of which even designers cannot understand how it behaves [3]. Also, the collection of sensitive data for the algorithms has raised another concern on privacy [4].

We propose an offline socializing application, *Cupeer*, to engage with people in one's vicinity by utilizing peer-to-peer (P2P) networking capabilities of modern smartphones, with no reliance on a central server (e.g. for authentication) or algorithm (e.g. for matching). Instead, we explore an interactive, decentralized approach to allow users to select their peers by themselves. *Cupeer* aims at delivering real-time, location-based, user-driven socializing experience while abstaining from collecting any sensitive data into a centralized server and encouraging direct human interaction.

The rest of this paper is composed as follows. Section 2 explains the design goals of *Cupeer* and its compartments. Section 3 explains the implementation detail and interaction between each part. Section 4 evaluates *Cupeer* in the technical aspect and user experience. Section 5 discusses the limitations of *Cupeer* and propose directions for future research. In Section 6, we conclude.

2 DESIGN

In designing *Cupeer*, we establish three main goals:

G1. Enabling the discovery of nearby users with connectivity offered by modern smartphones and a secure way to engage in (or disengage from) data transmission with other users
G2. A P2P matching algorithm based on user information gradually available to each other

* equally contributed

G3. A user-friendly interface

In the following subsections, we explain how we accomplish these goals.

2.1 Connection Method

Modern smartphones are equipped with dozens of sensors. Among them, for the P2P connection, we identified three possible candidates: Wi-Fi, Bluetooth, and microphone and speaker echoing inaudible sound. We initially planned to use Wi-Fi Direct [5], a Wi-Fi standard protocol using single-hop communication without an access point (AP), expecting to piggyback on the incredible success of the Wi-Fi. However, despite our efforts, we found it difficult to employ because of instability issues and a lack of practical guidelines for development.

Instead, we adopt the Google Nearby Connections API [6], which operates in a fully-offline P2P fashion by using all of the three methods depending on the circumstances of devices. By utilizing the strength of them and avoiding their respective weaknesses, it provides a high-speed, reliable connection. Moreover, it supports various types of topology—P2P_CLUSTER, P2P_STAR, and P2P_POINT_TO_POINT, enabling a wider range of communication.

The Nearby API [6] involves two stages: pre-connection and post-connection. In the pre-connection stage, there are Advertisers advertising themselves and Discoverers discovering nearby Advertisers and sending a connection request [6]. Once the connection is requested, they initiate symmetric authentication and subsequently accept or reject the request independently. After the connection is accepted by both parties, it phases into post-connection, in which both sides can exchange three types of data: BYTES, FILE, and STREAM. In this work, we only use BYTES while noting that FILE can be used for further development. As with other connection methods, either side can terminate the established connection to the endpoints.

2.2 Matching Algorithm

The paramount goal of matching algorithms is to solve the problem of adequately matching users given information related to their characteristics. As not every person is a good match for everyone, socializing applications, therefore, need to provide a mechanism for users to gauge their compatibility before they engage in interaction. Traditionally, it is achieved by a two-step process. Initially, the algorithms prefilter profiles and present some of them to the user using centralized algorithms for recommendation [7,8] or matching [9-11]. Then, users can choose whom to interact with by reviewing the information presented. However, this process does not align with our goals: such filters typically exploit pre-collected user data and behavior. Yet *Cupeer* refrains from any data collection and does not require even an Internet connection.

As an alternative, we propose a novel interactive approach by which users are enabled to choose their partners on their own. After the connection is set up, *Cupeer* presents a set of randomly chosen multiple-choice questions to both users, as shown in Table 1. Then, each user compares her own answers with her partner's. Based on them, she decides if she wants to continue the interaction. This allows users to gradually express their personality and preference. It can also serve as potential ice-breaking topics after the users get to chat with or meet each other. We argue that this interactive particular part of *Cupeer* entertains the users in a unique way. To test this assumption, we explicitly asked them in the user survey (Section 3.2).

For the question set, we adopt questions from a personality quiz of Pokémon's Red and Blue Rescue Team game [12] (Table 1). For future work, one can coordinate the question set to better fit the target users and/or a context in which they interact. (e.g. ice-breaking)

Q. A test is coming up. How do you study for it?
- Study hard.
- At the last second.
- Ignore it and play.

Table 1. A sample question from *Cupeer*.

2.3 UI/UX

As *Cupeer* centers around the human experience, providing an appealing and easily understandable UI was of significant importance in our work. As our service aims to motivate delightful in-person interactions, the presentation of the application itself should also strive to be a source of delight.

In part for the user survey (Section 4.2), we implement an in-app instructional menu, which guides users by explaining each stage of *Cupeer* as well as the UI elements. This not only enabled us to combine an explanation with a hands-on demonstration and but also unfettered us from a requirement that other user(s) be co-present for P2P connection.

In order to achieve a familiar look of modern applications and a quality user interface, we adhere to Material Design's principles regarding the color scheme and shape [13]

To further enhance the feeling of familiarity, we take inspiration from popular messaging apps like KakaoTalk [14] and introduced a set of relatable characters to the application. Because we want to keep the anonymity of users before matching, we share no profile images during the discovery process. However, instead of only using their name as a representation for a matching partner, users can also choose one of the avatars to represent themselves. This makes the process of discovery overall more delightful and allows for a small amount of expression in the personal choice of their own character. Lastly, it helps in giving *Cupeer* a memorable application identity. We purchased the rights to use these characters from an artist on Shutterstock, a well-known image sharing platform.

Due to the space limit, we will leave the implementation details of the UI to our GitHub repository [23]. To name a few, we use the CardViewStack library [15] to create the swiping animation for the quiz question; RecyclerViews with the Adapter-pattern [16] in the question, result, and chat activities so as to render multiple items of a similar structure in an effective manner.

3 IMPLEMENTATION

Following our design choices, we implement an Android application as a prototype (later used in the evaluation part, Section 4). For the architecture pattern, we adopt Model-View-Controller (MVC). In the MVC pattern [17] (Figure 1), Model simply represents the internal data structure and can be shared across applications. View interacts with the user by accessing the Model's data but does not know how to manipulate them. Between Model and View does the Controller reside. It handles events trigger from View and calls appropriate method(s) on the Model. View and Controller are application-specific, thus it needs to change across applications. In our design, each view has a corresponding controller. Through the rest of this section, we explain our implementation in a top-down approach: from View to Controller and Nearby Client.

We leave readers interested in the Model part to our GitHub repository [23].

For development, we used Samsung Galaxy S9 and LG V40 ThinQ both with Android 9.0 and Play Store 17.9 and Kotlin version of 1.3.61.

3.1 Views

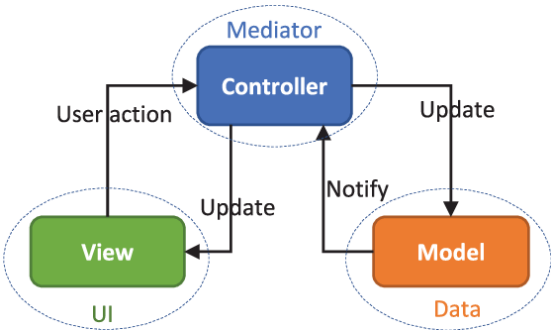

Figure 1. Model-View-Controller design pattern [15]

The View part has four key views: *Main*, *QuizQuestions*, *QuizResults*, and Chat. The flow is described in an activity diagram in Figure 2 and explained below.

Main

The users are shown the *Main* view. First, they can set their own name and avatar that will appear on the screen of others. After the setting, they can start searching other application users who have started searching, by pressing the "DISCOVER" button. Here, "DISCOVER" is simply meant to be a layperson's term for the users, not the one used along with "advertising" in the Nearby Connections API. It calls its corresponding controller, which then interacts with a Nearby client. When the button is clicked again, the searching stops accordingly.

After a connection is established (see how in the subsequent Network IO part), it is prompted the avatar and name of the other user, with the "CONNECT" button to proceed to the next stage, the quiz part, and the exit button to decline the incoming connection and go back to *Main*. If both users press the "CONNECT" button, the corresponding controller tells the view to move on to the *QuizQuestions* view. Otherwise, the rejected user is prompted a rejection message.

QuizQuestions* & *QuizResults

In the *QuizQuestions* view, both users choose answers for multiple-choice questions whose goal is to indirectly express their personality and preference through their choices. After a user finishes all the questions, she waits for the other to do so. If both have completed and received the choices from the other, the application proceeds to the next screen (via controllers).

In the *QuizResult* view, both users are presented the answers of both in comparison. They decide whether to proceed to the next stage given the information.

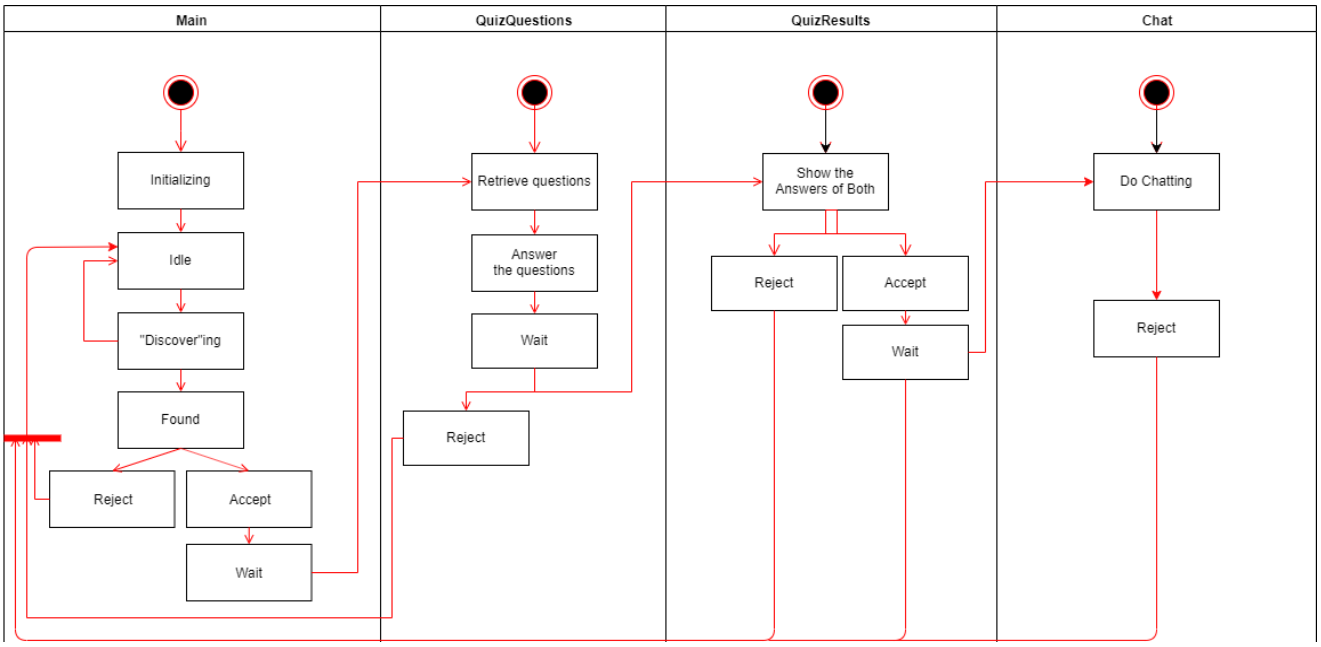

Figure 2. An activity diagram of *Cupeer*.

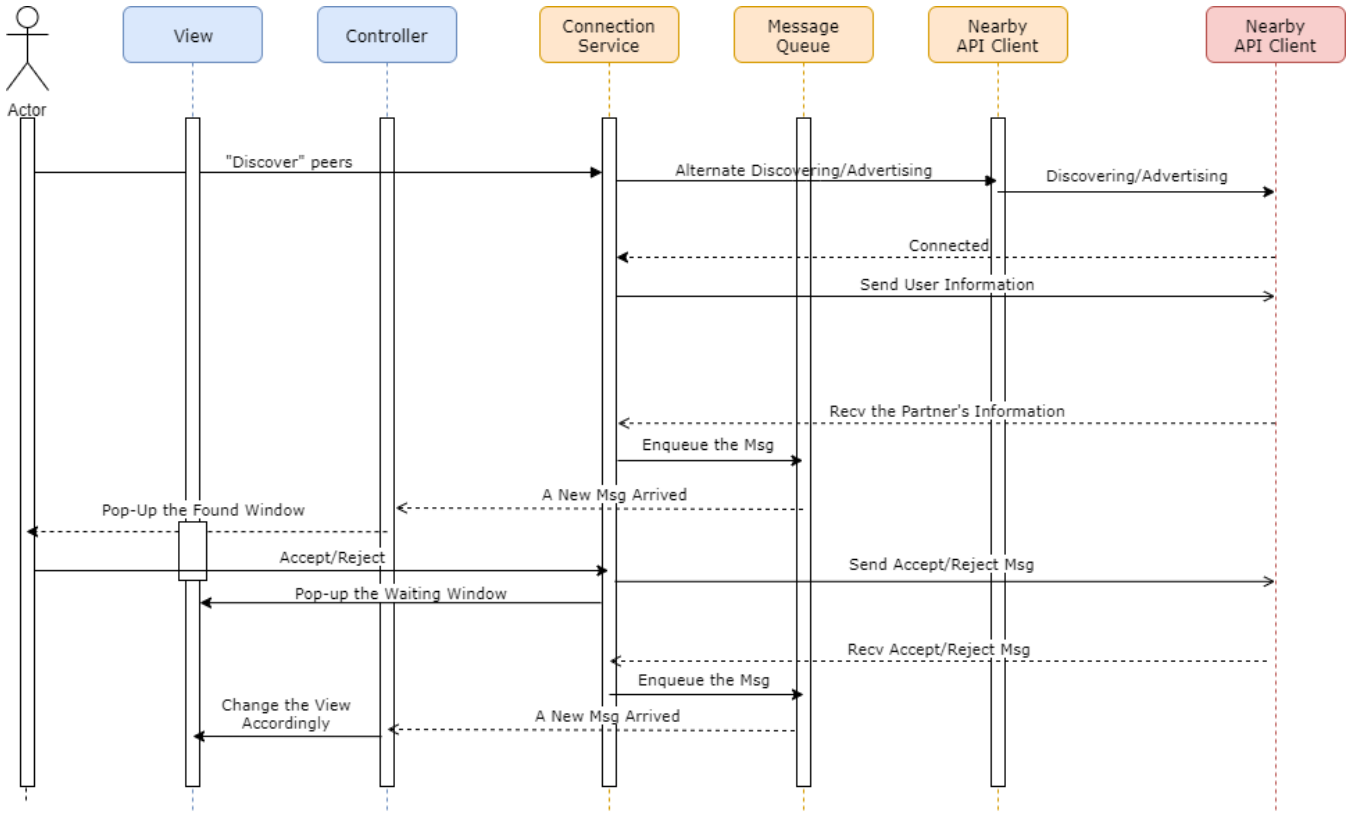

Figure 3. A sequence diagram of *Cupeer*. The View includes the four aforementioned views; the Controller includes their dedicated controllers. *ConnectionService*, *MessageQueue*, and Nearby API Client in yellow are unique across the application thus shall be initialized only once in the controller for Main. The Nearby API Client in red refers to the one running in other device(s).

Chat

Now that the users agree to engage in further, both sides are shown the Chat view. They can freely chat with each other about any topics including the quizzes. Even in this stage, either party can terminate the communication whensoever by pressing the exit button, which prompts a rejection pop-up to the other.

Finally, or skipping the casual chat, if the users want to meet each other face-to-face, they can talk about their location and appearance in the view. In this case, they can press the "FOUND YOU" button to return to the home screen without sending a rejection pop-up.

Instruction Mode

Using the Instruction Mode, novice users can learn how to use the application even without any people nearby. The helper agent mimics the virtual person and guides them along the way. It can be started in the *Main* view.

3.2 Controllers

The controller part has four dedicated controllers bound to their corresponding view and one special controller for network I/O. One challenge in implementing the controllers is that the Nearby client must be shared across the controllers seamlessly. When a user moves across views, the controller of the running view should operate as the listener to the Nearby API client, replacing the old one.

As a solution, we implement a dedicated controller named *ConnectionService* (Figure 3) to handle all interactions between the controllers that are bound to the views and the Nearby client. Internally, *ConnectionService* has an instance of the Nearby client and (its) *MessageQueue* for received data. One of the dedicated controllers at a time registers itself as a listener for the *MessageQueue* with its own pulling condition specified when it acts as an active controller. If any, the first element that satisfies the condition shall be delivered to the active controller. This conditional pulling is necessary to ensure the behavioral correctness of *Cupeer* as the packets do not necessarily arrive in order.

Additionally, the *ConnectionService* has an instance of lock in order to avoid any race condition. Whenever the internal states of *ConnectionService* including *MessageQueue* are accessed, the lock should be held. Also, we also leverage the lock when switching controllers similar to context switch in the operating systems:

```
In the switched-out view fromA,
connectionService.lock.acquire()
changeView(fromA, toB)

In the switched-in view toB,
connectionService.registerListener(this)
connectionService.lock.release()
```

3.3 Nearby Client

When a pair of users in the stage of "searching" is within the range of each other, e.g. sitting in the same cafe or in the lobby of the library. During "searching", a client alternates between "advertising" and "discovering" as one of them should operate as an advertiser and the other as a discover to get connected. The duration of each stage is picked randomly, not fixed, uniformly from 5 to 10 seconds to avoid cyclic failure. Nearby API requires Google Playstore services no earlier than 7.8. The payload must be *Parcelable*[1].

4 EVALUATION

In this section, we share a technical experiment with Nearby API [6] and evaluate our prototype in three user-oriented aspects.

4.1 Technical Experiment on Nearby Availability

We installed our application on the same pair of devices used for development (Section 3), with Wi-Fi and Bluetooth on. The devices are placed distantly in order to see the range of Nearby connection. We considered the devices are reachable to each other if they succeeded in finding each other three times out of four trials.

As an indoor test, our pair of devices were reachable to each other within 35 m (approximately measured with a smaller ruler; error < 2 m) in the IT complex building at our campus. We also tested them outdoors and found up to 38 m of reachability (using GPS; error < 3 m) Our measurement is in proximity to the specification of 802.1g standard [2], its indoor range being 125 ft (=38.1 m). On the other note, a group of more than two devices can support longer communication with the P2P_CLUSTER or P2P_STAR strategy. We leave its scalability for future work.

[1] https://developer.android.com/reference/android/os/Parcelable

4.2 User Survey with the Prototype

To evaluate the viability of *Cupeer*, we devised and conducted a user study (N=21) to get feedback on i) the idea of P2P socializing and ii) the application prototype.

Methodology

Over one week, we recruited 21 participants, consisting mainly of students in the S.T.E.M. fields. Of the participants, 7 of them identified themselves as female and 13 as male, with one participant preferring not to disclose one's gender. The age ranged from 20 to 32 (mean=23.7, sd=2.7). The participants were orally given a short introduction to the core idea of the application. With the built-in tutorial mode enabled, they ran through a simulated interaction in *Cupeer*. After using the application, they were asked to answer a survey aimed at collecting their experience. The survey is composed of 24 questions: i) open questions where participants could share their thoughts on specific areas and ii) Likert-scale (from 1 to 5) questions to rate different aspects of the application. In detail, the participants were asked about i) their opinions on the core idea of the service, ii) their impressions on the prototype implementation, and iii) their feelings towards the adaptation of *Cupeer* as a dating service.

The Core Ideas of *Cupeer*

In order to see the viability of the idea, P2P discovery, we first asked when and how often users could use such a service. Figure Q1. shows that 81.0% responded that at least one point in their lifetime, they would have founded a service like *Cupeer* convenient. The most mentioned examples included the phases of boredom and loneliness, using *Cupeer* to interact with others while traveling or the start of student exchanges. When asked how many times they could be in a position to use the application per week, 76.2% chose 1-3 times per week, followed by Never (14.3%), 4-6 times (4.8%), and 7+ times (4.8%). We suspect that one major reason for this level of expected usage was the hectic schedule of the students (mostly in engineering) during the semester. It could be worth looking into how differently a more diverse cohort would answer the question.

Next, in order to gauge *Cupeer*'s effectiveness in galvanizing an interaction between strangers, we asked: "app could make it easier for you to approach/meet new people?" on a Likert scale, where 1 is denoted as strongly disagree and 5 as strongly agree. The average was 3.95 (SD=0.67), implying that they agreed to some extent.

Lastly, we asked if they will be using *Cupeer* on a daily basis. Surprisingly, 12 (57.1%) responded "yes", while of 9 (42.8%) "no"s 4 participants state the reason being simply their unwillingness to meet new people all the time. This uprise implies that the participants are willing to put themselves more in a situation in which they have more chances to use *Cupeer* than before—only 1-3 times per week for 76.2%.

Prototype: the application

As we regard the UI appearance and usability important to the overall experiment (Section 2.3), we first ask about the visual appealingness of the prototype. On a Likert scale of from 1 (Hard to look at) to 5 (Very appealing), *Cupeer* received an average of 4.67 (SD=0.57) with 71.4% responding "Very Appealing." For usability, it scored an average of 4.19 (SD=0.58) where 1 is "very difficult [to use]" and 5 is "very easy [to use]." From our observation during the experiment, the "FOUND YOU" and the exit button on the Chat view seemed to convey their purpose not explicitly, hinting room for improvement in UI. However, given the high scores, we argue that our implementation of UI was more than enough to showcase the flow and core concept and of *Cupeer* and to receive feedback on them.

Next, we verify the effectiveness of the quiz as a matching algorithm. Out of the 21 participants, 18 (85.7%) responded it will help to see if both have similar personalities and/or their subsequent conversation. In open questions regarding the overall experiment, four participants specifically mentioned the question game as their favorite part of *Cupeer*, whereas another seven participants answered that they expect more elaborate quiz questions and a feature to input customized answers. We argue that the quiz concept proved its effectiveness while providing users with full authority to choose their partner and that it will be more effective in aiding the process with the suggested enhancements.

As a dating service

Finally, we fathom the possibility of transiting toward dating service. We asked if a user "can imagine using an enhanced version of *Cupeer* as a dating application." To that, 71.4% answered "yes," and no otherwise. For the enhancement, eight users answered that they wanted to know more information about the engaging partner after matching, such as a profile picture or more in-depth questions. It is admittedly necessary to include these additional features in a dating service, provided that people would regard their intimate relationships more seriously than general socialization and that sexual attractiveness plays a central role in those relationships [18].

Also, we asked about the perception of the matching algorithm and data collection in dating services. There were 8 participants who had experience in using any kind of dating applications, and they were asked to express the degree of agreement to two statements on a Likert scale from 1 (Strongly disagree) to 5 (Strongly agree). For the first statement, "the matching algorithms in dating applications are biased and/or affect the overall experience negatively," the average is 3.50 (SD=1.07); for the second statement, "worried about traditional dating apps having full access to your data," 3.25 (SD=1.28). At the moment, the participants seemed attracted to *Cupeer* for its creative way of matching and visually appealing interface, rather than its resolving the burgeoning societal concerns. Yet they expressed only a little concern over them, we believe the awareness of possible algorithmic bias and privacy invasion will grow as the voices around them keep resonating.

Limitations

In general, the results show that the users are open towards trying out a service like *Cupeer*. However, our experiment had some limitations. First, because we simulated the scenario without any real people co-present to meet nearby, the participants may not have felt in the same way as supposed to have on the spot. Second, the cohort has limited representativeness due to the recruitment process: mainly, promotion within computer science classes and requests to our acquaintances. Third, the experiment was a short-term, laboratory setting. In order to orchestrate more realistic situation, one can have a group of strangers to perform other tasks using *Cupeer* without having informed the real purpose of the experiment.

5 RELATED WORK

Most of the existing services, like Tinder, Match, and Bumble, keep their centralized matching algorithms secret, which raises concerns over fairness, accountability, and transparency. While relying on such centralized algorithms, Happn, Sumtu, and Spotted use geographical information to match users with "who you've crossed path". There are some experimental services that adopt the P2P scheme. Mingleton [19] uses iBeacon, or Apple's BLE technology, to match nearby people using Facebook data such as friends and hashtags. Bluedating (2005) [20], based on Serendipity from MIT [21], enables spontaneous connection via Bluetooth. Our work differs from them in that i) it does not rely on centralized server and algorithm, ii) real-time, location-based, user-oriented approach, and iii) gradual exposure of user data via P2P communication.

6 CONCLUSION

In this paper, we have proposed *Cupeer* as an offline socializing platform. We argue that our design and implementation fulfill its design goals—i) secure offline P2P discovery, ii) user-driven matching algorithm and gradual exposure of user data, and iii) a friendly-user interface. While each part leaves room for improvement and future work, we believe more users will shed light on user-centered but decentralized services like *Cupeer* to cultivate a healthy social life.

For this type of networking service to tip, many people need to have been using the service to attract new users. We think that one can begin by promoting it to conventions, schools, or cafes, where people are open to making new acquaintances.

A CONTRIBUTION

Disclaimer — the order in which the names appear is irrelevant to the degree of contribution.

During the project, we divided the work with an emphasis on the individual strengths of each teammate: Jaeseok used his extensive experiences in programming and system design to lead the technical part of the project and provided guidance for the writings. Adrian used his experience in designing user-centric applications to lead the UI development and shape the overall user experience by swiftly receiving feedback on it.

Idea Development - Jaeseok proposed several research-oriented topics and Adrian proposed several application-oriented topics. Among them, four ideas—one by Jaeseok and three by Adrian—were selected for further review. Both split the four into two by each and conducted further research on them separately. Both agreed to move on with P2P dating, which was originally proposed by Adrian and later evolved to *Cupeer*.

Elevator Pitch - Adrian made the outline and slides, which were reviewed together, and also delivered the elevator pitch itself.

Project Proposal - Jaeseok wrote the document, which was reviewed together.

Half Presentation - Jaeseok crafted the outline and slides (the design and some of the contents were borrowed from Adrian's elevator pitch), which were reviewed together, and delivered the presentation except for the demo part. Adrian developed a minimal demo program

(connecting via Wi-Fi Direct, not Nearby API) and presented it.

Research - Both conducted research on Bluetooth and local wireless media. The data flow was designed by both in its entirety Adrian devised the quiz game as the central matching algorithm

Development - As explained, we decided on using the MVC (Model-View-Controller) design pattern. The Model part was designed in tandem and was written into code by Jaeseok. At large, Adrian led and implemented most of the View part including the UI and program flow, focusing on the overall UX and Jaeseok led and implemented most of the Controller part, enabling the core technology of P2P connectivity with Nearby Connections API. Thereafter, Adrian additionally implemented tutorial mode instruction.

Final Demo - Adrian made the outline and slides. The presentation was delivered evenly by both.

Evaluation - Adrian led the user study and devised the questionnaire for the survey. The user study was prepared by Adrian and then conducted by both right after the last class and further by each posterior to that.

Write-up - Jaeseok used his experience in academic writing and led the final project report. He stitched together all the materials crafted by both. After that, the entire writing was reviewed and refined by both. The contribution to the writing materials per se, excluding Jaeseok's work for editing and (re-)formatting, is, at our best estimation, quantified by each section:

%/#	1	2	3	4	5	6	A	B	Ref
J	100	65	40	60	25	100	65	90	18
A		35	60	40	75		35	10	5

We acknowledge that Jaeseok had more editing discretion over what to include in the paper as per our agreed timeline. We publish what is best out of what we've had.

Etc. - Switching from Wi-Fi Direct to Nearby Connections was Adrian's idea. GitHub repository contribution – Jaeseok, 7,593 LoC; Adrian, 6,095 LoC.

B WHAT WE HAVE LEARNED

From developing this idea from scratch to tackling the challenges along the way, we learned and practiced: i) structuring mid-sized Android application in a collaborative way, ii) thinking from users' viewpoint and sticking to our design choices, iii) handling UI objects delicately and stitching everything together seamlessly with the P2P libraries, iv) developing and executing a user study for feedback cycle, and v) cooperation between a KAIST native and TUM exchange.

ACKNOWLEDGMENTS

We thank the Networking & Mobile Systems Lab for lending the smartphones and course staff and fellow students for giving valuable feedback on our project.

References

[1] Hootsuite, The global state of digital in 2019. Retrieved from https://hootsuite.com/pages/digital-in-2019

[2] Chopik, William J. "Associations among relational values, support, health, and well-being across the adult lifespan." Personal relationships 24.2 (2017): 408-422.

[3] Yampolskiy, Roman. "Unexplainability and Incomprehensibility of Artificial Intelligence." (2019).

[4] Lutz, Christoph, and Giulia Ranzini. "Where dating meets data: investigating social and institutional privacy concerns on tinder." Social Media+ Society 3.1 (2017): 2056305117697735.

[5] Wi-Fi Alliance, P2P Technical Group, Wi-Fi Peer-to-Peer(P2P) Technical Specification v1.0, December 2009

[6] Google, 2019. Overview | Nearby Connections API. Retrieved from https://developers.google.com/nearby/connections/overview

[7] Yang, Xiwang, Yang Guo, and Yong Liu. "Bayesian-inference-based recommendation in online social networks." IEEE Transactions on Parallel and Distributed Systems 24.4 (2012): 642-651.

[8] Kazienko, Przemysław, and Katarzyna Musiał. "Recommendation framework for online social networks." Advances in Web Intelligence and Data Mining. Springer, Berlin, Heidelberg, 2006. 111-120.

[9] Wang, Zhibo, et al. "Friendbook: a semantic-based friend recommendation system for social networks." IEEE transactions on mobile computing 14.3 (2014): 538-551.

[10] Silva, Nitai B., et al. "A graph-based friend recommendation system using genetic algorithm." IEEE congress on evolutionary computation. IEEE, 2010.

[11] McCaffrey, Daniel, et al. "Friend recommendation system." U.S. Patent No. 10,315,106. 11 Jun. 2019.

[12] FANDOM Games Community, Red and Blue Rescue Team Quiz Guide. Retrieved from https://pokemon.fandom.com/wiki/Red_and_Blue_Rescue_Team_Quiz_Guide

[13] Google, Design - Material Design. Retrieved from https://material.io/design/

[14] Kakao Corp, KakaoTalk | KAKAO. Retrieved from https://www.kakaocorp.com/service/KakaoTalk?lang=en

[15] yuyakaido, yuyakaido/CardStackView: Tinder like swipeable card view for Android. Retrieved from https://github.com/yuyakaido/CardStackView

[16] Freeman, Eric, et al. Head first design patterns. " O'Reilly Media, Inc.", 2008.

[17] Leff, Avraham, and James T. Rayfield. "Web-application development using the model/view/controller design pattern." Proceedings fifth ieee international enterprise distributed object computing conference. IEEE, 2001.

[18] Margolin, Leslie, and Lynn White. "The continuing role of physical attractiveness in marriage." Journal of Marriage and the Family (1987): 21-27.

[19] Business Insider. Mingleton - Business Insider. Retrieved from https://www.businessinsider.com/mingleton-2014-4.

[20] Tis Veugen. Blue Date - App Store on Google Play. Retrieved from https://play.google.com/store/apps/details?id=nl.tistis.bluedate&hl=en.

[21] Eagle, Nathan, and Alex Pentland. "Social serendipity: proximity sensing and cueing." submitted to Ubicomp (2004).

[22] Vassis, Dimitris, et al. "The IEEE 802.11 g standard for high data rate WLANs." IEEE network 19.3 (2005): 21-26.

[23] Jaeseok Huh, iriszero/Cupeer. Retrieved from https://github.com/iriszero/Cupeer