

Wi-Friend: Identifying Potential Real Life Friends Nearby

Jia Wang, Xuefeng Liu, Jiannong Cao

The Hong Kong Polytechnic University, Kowloon, Hong Kong, China

Email: {csjiawang, csxfliu, csjcao}@comp.polyu.edu.hk

Abstract—Nowadays, with the help of various on-line social networking applications, one can freely interact with any person in their friends list. However, in most conditions, people in your friends list are those you know a priori, either via face-to-face talking or introducing by a common friend. This way of establishing ones friends list can miss many potential friends nearby. These ‘potential real life friends nearby are those who are invisible to you (you do not know before), physically close to you (e.g. taking the same subway for commuting, doing exercises at the same time slot), and share the same interest with you. Correspondingly, we developed Wi-friend. Wi-friend, once installed in your mobile phone, can help to find these ‘potential friends nearby’. Using Wi-friend, one only needs to briefly specify his/her interest. Then the smartphone starts to look for nearby people with the same interest and add each other into ones friends list automatically. One special feature of Wi-friend is that it does not rely on Internet connection. We leverage the Wi-Fi tethering technique, and let ones smartphone to switch between Wi-Fi hot-spot mode and Wi-Fi client mode to exchange information with nearby people. In addition, Wi-friend can dynamically determine how long to stay in each mode to maximize the number of people who can exchange information. Extensive experiments and simulations demonstrated the technical feasibility and the effectiveness of our Wi-friend system.

I. INTRODUCTION

With the help of various online social networking applications such as Facebook, Twitter and Wechat, one can easily communicate with their friends in the friend list[1]. Generally speaking, people in your friends list are those who were introduced by mutual friends, those joined the same online interest group, or those who have had a face-to-face talking with you. However, these ways of establishing the friend list can potentially miss many potential real life friends nearby. These friends are those who you do not know yet, but have similar interests and physically close to you. They can be those who often take the same subway with you for commuting every day, those who regularly doing exercises to you in the gym, or those who usually sit near to you in the library.

Existing popular online social networking applications, however, are not efficient to identify these potential real life friends nearby. For example, Facebook has launched a new feature called ‘Nearby Friend’ which can track location of people [2]. However, the friends identified are limited to those who are already in your friends list. Wechat contains a similar feature [3] which allows you to find people nearby to you. However, using Wechat you have to manually add people

in your friend list first and then talk to them to find out whether they are ones you are interested. Moreover, these social networking applications rely on Internet connection to locate these people and to communicate with each other. Therefore, people cannot use these applications without Internet connection.

To address these problems, we develop Wi-friend, an Internet-independent social application targeting on finding potential real life friends. Using Wi-friend, one only needs to briefly specify his/her own interest and the characteristics of expected friends. Then the Wi-friend will automatically look for nearby friends and add each other into one’s friend list if there is a match.

An important feature of Wi-friend is that it is designed to work in an offline mode without Internet connection. To achieve this objective, we leverage the Wi-Fi tethering technique to turn smartphones either in the Wi-Fi hot-spot (AP) mode or in the client mode. In addition, we let smartphones change their modes to realize two-way communications. Furthermore, considering frequently adjusting the working mode can incur a large delay, Wi-friend also adjusts the changing frequency dynamically to maximize the overall performance.

We have experimented Wi-friend in the campus of Hong Kong Polytechnic University. The experiment results show with the help of Wi-friend, it takes about 20 seconds for a mobile phone to exchange information required for matching friends with 8 mobile phones nearby. In addition, the extensive simulations are conducted to evaluate the performance of Wi-friend in larger scale scenarios.

The main contributions of this paper are as follows: (1) We develop an application which can help people automatically to find potential real life friends nearby. (2) We design mode assignment mechanisms which dynamically determine a smart phone how long to stay in each mode to maximize the number of people who can exchange information. (3) Field experiments and simulation have demonstrated the technical feasibility of our proposed system.

The remainder of this paper is organized as follows. In Section II, we reviewed related existing works. In Section III, we elaborated the system design and architecture. The details for the operation mode selection algorithms are presented in Section IV. In Section V, we explained the system implementation details and experiment results. Finally, the simulation results on the user density scenario are given in Section VI.

II. RELATED WORKS

Despite of the popularity of many popular online social networking applications like Facebook and Wechat, they are not able to efficiently identify one's potential real life friends. For example, Facebook's 'Nearby Friend' feature can track the location of people [2], but limited to those who are already in your friends list. Similarly, Wechat contains a similar feature called 'Look Around' [3]. With this feature, one can find nearby people using this feature easily. However, using Wechat you have to manually add people in your friend list first and then chat with them to find out whether they are really the ones you are interested. Different from these applications, our proposed Wi-friend can help to identify potential real life friends. Using Wi-friend, one only needs to briefly specify his/her own interest and the characteristics of expected friends. Then the Wi-friend will automatically look for nearby friends and establish the connection if there is a match.

Another drawback of all the online social applications like Facebook and Wechat is that they rely on Internet connection to locate these people and to communicate with each other. Smartphones running these applications act as simple clients and send out user locations to the third-party servers via Internet. These servers then collect location data and provide the service to smartphones. As a result, people cannot use these applications without Internet connection. In contrast, Wi-friend is designed to work in an offline mode without Internet connection.

Similar to Wi-friend, there are many existing works for opportunistic communication without relying on the Internet connection. A remarkable progress is the work by Jung and Woo-Sung [10], they presents the light weighted solution by inserting message into SSID for delay-tolerant communications in public space. However, the message length is limited by the content capacity of SSID. The serval project [9] is another example implemented a communication system to provide free phone-to-phone local message sending via their Wi-Fi interfaces, but the root access of smartphones is required. In addition, Beacon Friend Finder [11] works by exchanging identifiable user messages via classic Bluetooth signals and the [12] provides the ad-hoc network communication by utilising the 'Wi-Fi Direct' function on smartphones. However, the network initiation processing is time-consuming which requires lots of user interventions for paring devices.

To eliminate the manual operation, WiFi-Opp [13] provides opportunistic communication service on smartphone with the constant switching interval between client mode and access point mode, which is similar to the mechanism proposed in Wi-friend. Later on, FireChat [14] enables the opportunistic communication based on the Wi-Fi direct and bluetooth low energy technique. However, after testing these applications, we find that (1) both the WiFi-Opp and FireChat perform slowly when paring with new device in the mobile environments. (2) The applications can provide enduring communication only when users are close no more than 30 meters. By contrast, Wi-friend is a more efficient and feasible solution to connect

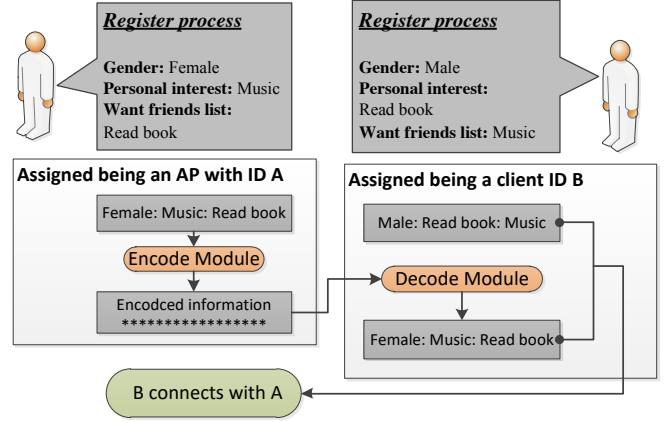


Fig. 1: The function and working process of the system.

nearby devices without creating any heavy user intervention in a larger space. Moreover, Wi-friend can be readily installed on any type of smart mobile devices without requiring root access privileges.

III. SYSTEM OVERVIEW

The working process of the our system is illustrated in Fig. 1. In the first step, the user fills his personal information (e.g., gender and personal interest) into the profile. Then, the user inputs the set of key words (e.g., sports, books) in the friend finding list representing features with which he expects to find such friends. After that, the user configures a message to be sent. In the second step, the application dynamically changes the mode of the smartphone to detect whether there are eligible users nearby. Once the smartphone turning into the AP mode, the application encodes user's profile and sets the encoded text as its SSID, then it starts the hot-spot function. Whereas if the smartphone is in the client mode, the application first scans all reachable SSIDs, then decodes contexts of those collected SSIDs through the decode module. If the extracted context matches the friend list, the client smartphone connects to that Wi-Fi point. After setting up the network connection, configured messages is sent from smartphone in the AP mode to smartphone in the client mode via the TCP socket.

There are numbers of candidate techniques for building up the underlying network. We utilise Wi-Fi tethering technique in our system for couple of reasons. First, when searching nearby devices and establishing connections, Wi-Fi direct and Bluetooth techniques require user interventions for authorization. Second, the communication range by Bluetooth is limited. Therefore, we select the Wi-Fi tethering technique in this system by blocking the Internet sharing function to prevent the extra charges for the 3G data usage. However, utilising this technique, a smartphone in the client mode cannot initialize the communication with smartphones in the AP mode, and two smartphones with the same roles cannot communicate with each other. Thus, to improve the efficiency for finding the nearby friends, it is preferable to have smartphones change their mode. Considering the time cost on the mode switching

lowers down the efficiency for finding friends, we design the strategy dynamically determining how long to stay in each mode, which is presented in the following sections.

The flow chart of state transition processes is illustrated in Fig. 2. Our system makes smartphones switch between three different modes: preparing mode, client mode, and AP mode.

To be more specific, before switching the operation mode, the system always moves into the preparing mode for turning on and turning off the networkboard. When the smartphone is in the AP mode, the logic flow can be divided into three steps as follows.

- **Preparing:** Turn off the Internet sharing function, and move into the AP mode.
- **Friends initiator:** It encodes features of user's interests and sets the cipher-text as the SSID configuration. Then it starts up the Wi-Fi tethering function.
- **Set up a TCP connection:** Once having set up a Wi-Fi connection with a potential friend. This step sends messages to the connected smartphone.

Meanwhile, for smartphones in the client mode, the system operates the following steps.

- **Preparing:** Disable the Internet sharing function, and move into the client mode.
- **Friends discover:** Scan all nearby SSIDs, decode the context of detected SSIDs checking if it matches the friend finding list. Once discovered, the smartphone connects to that AP.
- **Set up a TCP connection:** Once setting up a Wi-Fi connection ready, the client sends prepared messages to the connected smartphone.

IV. OPERATING MODE SELECTION

In order to efficiently find nearby friends, this section discusses solutions to adjust the mode switching interval thereby optimizing the trade off between the switching overhead and the overall performance.

In our system, a communication session is achieved through a single-hop radio transmission. The topology of such a network can be modeled as a unit-disk graph $G_W = (V, E)$ [13], in which users with the same interest are presented as vertexes in the network. And the graph has an edge e_{ij} , if the distance between two nodes v_i and v_j is within the Wi-Fi communication range R and the interest point of the user v_i is within the interest set of the user v_j . The definition for interest point and interest set are as follows:

Definition 1. Interest point. The features (e.g., 'pets', 'piano') which are in the friend finding list.

Definition 2. Interest set. The series of keywords (e.g. 'music', 'instrument', 'piano') represents user's own interest.

Our target is to minimize the iteration when setting up all edges in the graph by iteratively inserting each node $v_i \in V$ into subsets S_{ap} and S_{ct} . Considering the limitation of the tethering technique, a node $v_c \in S_{ct}$ can only connect to one $v_a \in S_{ap}$ in each round. The problem formulation is given as below:

Given:

- a graph $G_W = (V, E)$ has all nodes $v_i \in V$ in each other's friend finding list.
- S_{ap} has nodes $v_a \in S_{ap}$ working in the AP mode.
- S_{ct} has nodes $v_c \in S_{ct}$ working in the client mode.
- the edge e_{ij} is activated if $v_i \in S_{ap}$ and $v_j \in S_{ct}$.

The objective is to find the AP set $(S_{ap}^j \subset V, j = 1, \dots, n)$ and the client set $(S_{ct}^j \subset V, j = 1, \dots, n)$ for the corresponding rounds $r_1, \dots, r_n$, such that the totally number of iteration $r_1 + r_2 + \dots + r_n$ is minimized when all e_{ij} are activated, which subjects to the following constraints:

- In each round, each v_c in the client mode, only one edge e_{ac} can be activated by connecting with a node v_a , within the communication range $d(v_c, v_a) \leq R$.
- Add the counter C one, once there is an edge being activated. C should be equal to the number of edges e_{ij} in the given graph.

Briefly speaking, the first item presents the constraint that in each round, users in client mode can only connect to one user in the AP. The second constraint ensures that all edges have been connected at least once at the end of the iteration, such that guaranteeing finding all potential friends nearby.

For minimizing the total number of the iteration, in each round r_i , we should find the optimal AP set S_{ap}^* that maximizes the number of edges connected. Therefore, the objective function is transformed to the *bounded multi-dimensional knapsack problem (BMKP)* in each round, which is stated as:

$$\begin{aligned} & \text{Maximize } \sum_{i=1}^n v_i x_i, x_i = 0, 1, \dots, l_i, \\ & \text{Subject to } \sum_{i=1}^n w_{ij} x_i \leq c_i, j = 1, \dots, d, \end{aligned} \quad (1)$$

Where x_j represents the number of instances of item j have included in the knapsack $x_j = 1, \dots, l_j$. n is the number of instance types, d is the number of dimensions that restraint the knapsack, v_j represents the weight of the instance j in the knapsack, c_i represents the i th dimension's capacity of the knapsack, and w_{ij} represents the entries in the constraint matrix of the knapsack. After transforming, our problem is written in the form of the BMKP as below:

$$\begin{aligned} & \text{Maximize } \frac{1}{2} \sum_{i < j} 1 - x_i x_j, \\ & \text{Subject to } \sum_{i=1}^m i < \sum_{j=1}^n j, \\ & \text{And } v_i \in N(v_j), \end{aligned} \quad (2)$$

where x_i and x_j are the weights of edges, and $x_i = 1$, if one of the end node $v_i \in S_{ap}$ and the other node $v_j \in S_{client}$, while $x_i = -1$, if both of the end nodes v_i and v_j are in the same set. Notice that for any i and j , $1 - x_i x_j$ is 0 if the end

nodes are in the same set, and 2 otherwise. This means that $\sum_{i < j} (1 - x_i x_j)$ is twice as much as the number of the edges. Dividing this value by 2 gives the number of the edges. $N(x_i)$ is the neighbor set of the node x_i , and neighbor is defined as the nodes with the one hop distance to the node x_i .

To maximize the object function, the two ends (i.e. v_i and v_j) of the edge e_{ij} should in the different set. we apply the dominating set algorithm to tackle our problem [16] [17]. The minimum dominating set problem concerns to find the minimum subset of the graph which contains vertexes enabling every vertex not in the subset is adjacent to at least one vertex in this subset. This is a classical NP-complete decision problem in computational complexity theory [18].

To solve this, brute force method is applied to find the benchmark for the given object function. Then we study the centralized algorithm [16] and distributed algorithm [17] respectively, to solve this problem.

A. Brute Force Method for The Mode Selection

In each round $r_i \in R^*$, we enumerate all possible set partitions that optimize the objective function above. Theoretically, the number of possible solutions increased exponentially with respect to 2^M , where M is the number of the nodes in the graph. The complexity of this method is $R \cdot 2^N$. A collection of enumerated mode shape vector MS_i^k at the round r_i denoted as MS_i^k is a 1-by-m row vector. The nodes assigned in the AP mode is represented by the '1'. For example, $[1, 0, 0, 1, \dots, 0]$ means the first and third nodes are in the AP mode. The range of MS_i^k are from $[0, 0, \dots, 0]$ to $[1, 1, \dots, 1]$.

In each round r_i , we find the best setting which maximizes the number of connected edge e_{ij} . Once e_{ij} is activated, the counter C increases one. the iteration is completed when $C \geq M$, where M is the number of edges in the graph.

B. Centralized Algorithm for The Mode Selection

On this centralized strategy, each mobile device is initialized to connect with the base-station, reporting the current location. Hence, the global geometric information is known.

Mode Selection Process. The network is represented as a graph $G_W = (V, E)$ where V is the set that contains all vertexes in the graph and E is the edges collection. C is the connected edge set which means edges in this set are activated. If the edge connected set C is not empty, in each turn, the nodes will be reassigned state based on the following centralized dominating set algorithm. First, all the nodes will be colored to white belonging to the white set. Then as shown in Algorithm 1 from line 6 to line 12, the degrees for all nodes in the white set is calculated. The nodes with the maximum degree are colored black, and its neighbor nodes are colored gray. This process will be repeated until all nodes have been colored. And if nodes have completed the communication with all nearby nodes, the nodes will be removed from C . Nodes with black colors are assigned into the AP set, others are assigned into the client set.

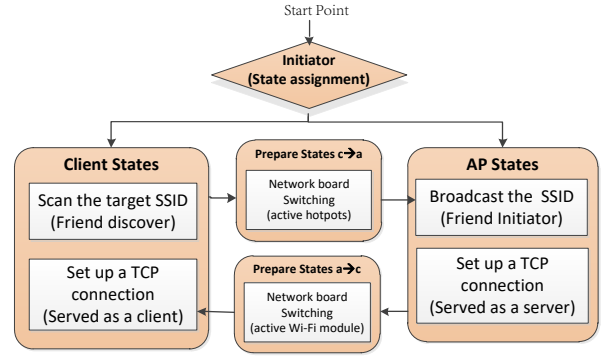


Fig. 2: State transition diagram.

Algorithm 1 Centralized algorithm for the mode selection.

```

1: Set the color {black, gray, white}
2: Initially assign white to each  $v_i \in V$ 
3: while  $C \neq \phi$  do
4:   if  $Set(white) \neq \phi$  then
5:     Calculate the degree  $\delta_i$  for each  $v_i \in Set(white)$ 
6:     for Each  $v_i \in V$  do
7:       if  $\delta_i = \argmax \{d(v_i) | v_i \in V\}$  then
8:         Mark  $v_i$  black
9:         Mark each  $v_j \in N(v_i)$  gray
10:        Remove  $v_i$  and  $N(v_i)$  from the  $Set(white)$ 
11:      end if
12:    end for
13:  end if
14:  for Each  $v_i \in V$  do
15:    if  $N(v_i) \cap Set(white) = \phi$  then
16:      Remove  $v_i$  from  $C$ 
17:    end if
18:  end for
19: end while

```

C. Distributed algorithm for the mode selection

For the distributed mechanism, each smartphone is able to switch the mode based on the following algorithm.

Mode Selection Process. We propose a marking process that marks every node in the graph. $m(v)$ is a marker for node $v \in V$, which is either in black or white. We assume that all nodes are marked white initially. $N(v) = \{u | v, u \in E\}$ represents the open neighbor set of the node v . On the line 2, each node in the graph exchanges its open neighbor set N_i with one hop neighbors. The node will be marked black if any node in its neighbor are not connected. In the following, two rules are defined to reduce the redundant nodes in the black set. As shown from the line 9 to line 18, if the node v_i has neighbor nodes n_i and n_j that are in the black set. Once the neighbor set of n_i belongs to the neighbor set of n_j , the node with the small degree is removed from the black set. Moreover, as described from line 17 to line 20, second rule considers if

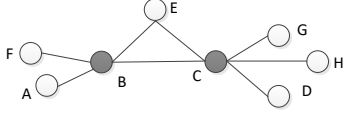


Fig. 3: The example of distributed algorithm for the mode selection. Smartphones working in the AP mode are marked black.

the neighbor nodes of v_i are within the intersection of the n_i and n_j 's neighbor set, i.e. $N(v_i) \subset N(n_i) \cup N(n_j)$, then the node with the small degree is removed from the black set.

After this, nodes with black colors are assigned into the AP set, others are assigned into the client set.

The corresponding mode selection process is presented as below.

Algorithm 2 Distributed algorithm for the mode selection.

```

1: Initially assign white to each  $v_i \in V$ 
2: Exchange open neighbor set  $N_i$  with one hop neighbors
   to get neighbor set  $N_i$  for the node  $v_i$ 
3: while  $N_i \neq \phi$  do
4:   for Each pair of neighbor nodes  $(n_i, n_j) \in N_i$  do
5:     if  $e(n_i, n_j) = \phi$  then
6:       Mark  $v_i$  black
7:     end if
8:   end for
9:   Exchange information with one hop neighbors  $N_i$ 
10:  if The neighbor  $n_i \in \text{black}$  and  $N[n_i] \subseteq N_i$  then
11:    Exchange the degree information with neighbors
12:    if  $\text{degree}(v_i) < \text{degree}(u_i)$  then
13:      Mark  $v_i$  white
14:    end if
15:    if  $N(v_i) \subset N(n_i) \cup N(n_j)$  and  $\text{degree}(v_i) =$ 
        $\min \{ \text{degree}(v_i), \text{degree}(u_i), \text{degree}(u_j) \}$  then
16:      Mark  $v_i$  white
17:    end if
18:  end if
19:  for Neighbor node  $n_i \in \text{White}$  do
20:    Remove  $n_i$  from  $N_i$ ;
21:  end for
22: end while

```

Example. In the example of Fig. 3, after exchange the information with neighbor nodes, the node A knows it has the neighbor node B, and the B's neighbor contains node A, F, and E, the C has the node B, D, E and G, the D has C and the E has neighbor B and C. After processing the algorithm 2, the B has unconnected neighbor A and E, then node B is marked black being in the AP state. Similarly, after the marking process, the node C is assigned in the AP mode.

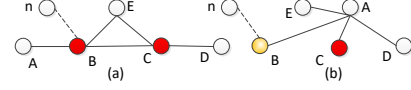


Fig. 4: (a) Case I as the system cooperates with a new device n joining the network. (b) Case II as the system cooperates with a new device n joining the network. Red nodes are smartphones working in the AP mode. Yellow nodes are smartphones switching from the client mode into the AP mode.

D. Update The Mode Selection in The Dynamic Scenario

The topological of an opportunistic wireless network is changed on the two different types: some new devices join the network, and some devices leave the network. In the rest part of this section, we will discuss how to update the mode selection in the dynamic scenario.

New device joins the network. Based on the updating algorithm proposed in [18][19], When a new device coming, the neighbors of this device in the client mode updates their state. Meanwhile, the devices in the AP mode keep their states. For example, in the Fig. 4(a), when the device n joins the network, the AP neighbor B is not affected because it already has connected neighbor A, C, E. However, in the Fig. 4(b), the new coming of the device n affects the state of the node in the client mode (i.e., B), forcing the device B to switch from the client mode to the AP mode. The corresponding marking process for the new coming device n is presented as follows:

Algorithm 3 The system cooperates with a new device coming to the network.

```

1: Initially turn the new device  $n$  into the sniffer mode getting
   the open neighbor information.
2: if the  $n$  has at least two unconnected neighbors then
3:   Switch the  $n$  into the AP mode.
4: end if
5: for Each neighbor node  $n_i \in N(n)$  do
6:   if  $n_i$  is in the client mode and the  $n_i$  has at least two
       unconnected neighbors then
7:     Switch the  $n_i$  into the AP mode.
8:   end if
9: end for

```

Device leaves the network. As illustrated in the Fig.5 (a), leaving device n originally in the AP mode affects all its neighbors to update the mode. On the other hand, in Fig. 5 (b), device n in the client state leaving the network does not affect any of its neighbor. The corresponding mode selection process for cooperating the new leaving device n is described as follows:

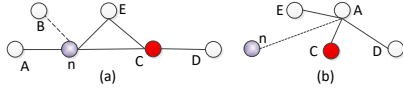


Fig. 5: a) Case I as the system cooperates with a new device leaving the network. (b) Case II as the system cooperates with a new device leaving the network. Red nodes are smartphones working in the AP mode.

Algorithm 4 The system cooperates with a device leaving the network.

```

1: if the leaving device  $n$  is in the client mode then
2:   No updating
3: else
4:   Turn the new device  $n$  into the sniffer mode getting
     the open neighbor information.
5:   for Each neighbor device  $n_i \in N(n)$  do
6:     if  $n_i$  has at least two unconnected neighbors then
7:       Switch the  $n_i$  into the AP mode
8:     end if
9:   end for
10: end if

```

V. IMPLEMENTATION AND DEMONSTRATION

A. Implementation Details

To evaluate the performance of the proposed mode selection algorithm, several testings have been conducted. In the field testing, 8 students with the similar interest (i.e., playing as potential friends to each other), carried android smartphones (with Quad-core 1.3 GHz CPU, 8GB RAM memory and Wi-Fi 802.11 b/g/n), walking or standing still in the open space of the Hongkong polytechnic campus, as illustrated in the Fig. 7. The red points A_i refer to users currently being assigned in the AP mode. The black points C_i represent users currently being assign in the client mode. Sub-figures (a) and (b) of Fig. 7 in the first line represent the static scenario where users do not move. While, sub-figures (c) and (d) in the second line represent the dynamic scenario where users move within the red circle with the walking speed. The PC (with Intel 2.70GHz i7-2620M CPU, 4GB RAM, and Windows OS) running the Matlab (2009a) takes the role of the central controller, calculating and optimal modes for each smartphone in each round. The time cost for a user finding all ‘potential friends nearby’ (i.e., completed exchanged information) is recorded by the application.

B. Evaluate Criteria

In the experiments as illustrated in Fig. 8, we concern the connection time (CT) which consists of the time costing on the state selection, network board switching and Wi-Fi network connection setting up. Also, we measure the round transmission time (RTT) for a message sending between a client device and an AP device.

TABLE I: THE NETWORK BOARD SWITCHING TIME ON THE DIFFERENT SMART-PHONE BRAND.

Smart-phone brand	The network board switching time(s)
Huawei Honor 6	8.543
Xiaomi redmi	7.091
Samsung Note3	8.991
Nexus 6	8.232

C. Performance Analysis

The data captured from the experiment are plotted in Fig. 9. For field testing, we considered both static (a) (b) and dynamic (c) (d) scenario, measuring the time for find all potential friends nearby. In the completely connected scenario, as illustrated in the Fig. 6(a) and (c), all users are within the communication range. Thus, one user is selected as the AP (i.e., A_1), and others in the client modes. While, the (b) and (c) have some users which are out of the communication range, so that two of them are assigned in the AP mode (i.e., A_1, A_2).

When starting the experiment, the user first starts the mode switching algorithm. If utilizing the centralized algorithm, each smartphone will first uploads information (i.e., ID and location) to the center station. Then, the center station decides which mode the smartphone should turn to. A computer running the Matlab is played as the center station in this experiment. Meanwhile, when we evaluate the performance of distributed algorithm, each smartphone first scans the network and collects all SSIDs of its neighbors. This neighbor information then is reset to its SSID. After exchanging its own neighbor information (i.e., SSID) with all neighbors. The operation mode is determined by the smartphone itself by following the distributed algorithm.

The measurement plotted in the Fig. 8 shows that the network connection time (CT) is much longer than the message exchange time (RTT). This suggests the system latency is mainly caused by the mode switching operation, rather than the time cost on the message transmission delay. Also, from the statistic result of the RTT, it is shown that the average RTT is influenced by the distance. The RTT ranges from 11.83s to 12.53s for smartphones staying 25m away from the AP. Meanwhile, for smartphones staying 50m away, the average RTT time ranges from 16.95s to 17.83s, and for smartphones staying 100m away from the AP, the time ranges from 17.88s to 18.65s.

As discussed above, when switching the smartphone from one mode to another mode, the switching time cost by the network board is the main reason for delay. Considering the different type of smartphones, the testing result of switching time for different network boards is presented in Table I. From these data, we find the switching time for different type of network board are similar, which are all around 8s.

The relation of the network connection time CT with the CPU occupation rate is presented in Table II. When conducting the testing, we control smart-phone applications running on the back-end. These applications do not disturb the Wi-Fi con-

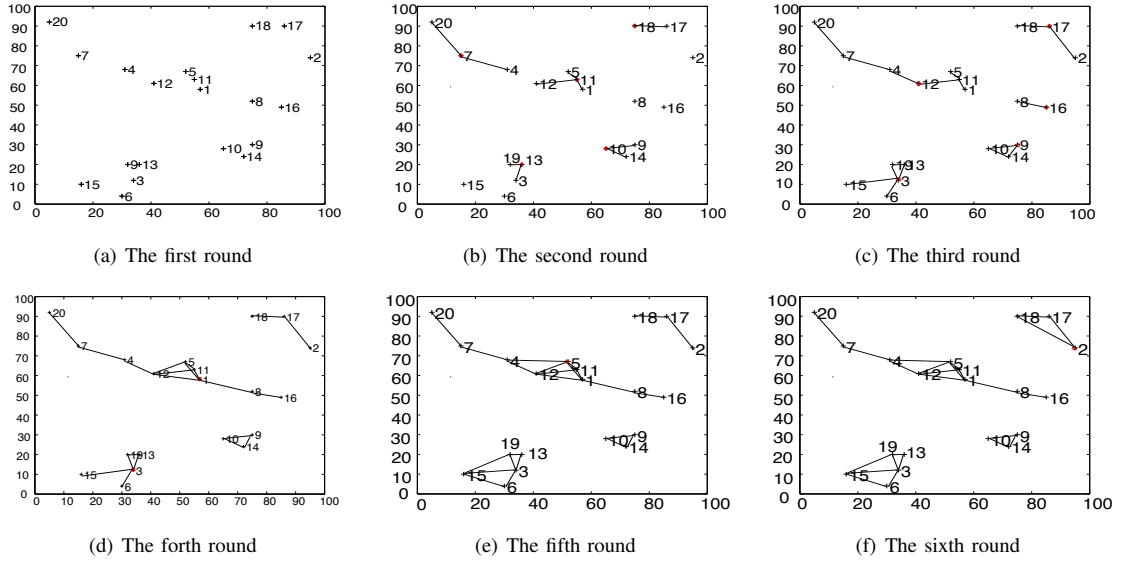


Fig. 6: The links get connected in each round. The red points represent smartphones working in the AP mode. The line represents that its end nodes have get connected in this round.

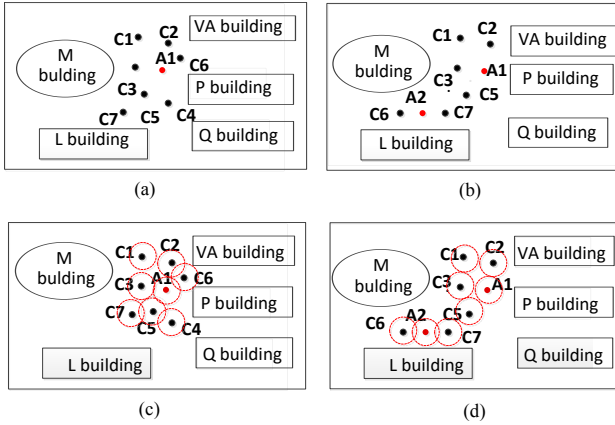


Fig. 7: The experiment environment.

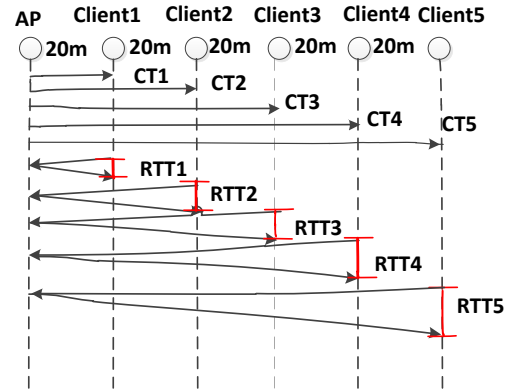


Fig. 8: The evaluate criteria.

nection and use other communication sensors (i.e., Bluetooth or NFC sensor), but more applications running on the back-end, more CPU resources of the smartphone are occupied. The experiment results show that the CPU occupation rate has the limited influence on the network connection time especially when increasing the CPU occupation from 60% to 80%, the connection time goes from 11.591s to 12.433s, which infers that the hardware constraint for switching network board is still the main fact that affects the system performance in the term of the network connection time.

Moreover, the testing result of time consumed on getting paired (i.e., successfully connected with another smartphone with target ID) is listed in table III. The measurement results show that the performance of centralized strategy is much

TABLE II: The CPU OCCUPATION RATE INFLUENCE ON THE CONNECTION TIME.

CPU occupation rate	The connection time(s)
20%	11.032
40%	11.215
60%	11.591
80%	12.432
90%	13.452

better than the distributed strategy. On the centralized strategy, since smartphones uploading their information to a central station, the station knows the global distribution of each user. Thus, it can switch to the optimal mode referring to the col-

TABLE III: THE TIME THAT THE SMART PHONE FIRST GET CONNECTED.

Test cases	The mean value(s)	The variance
Centralize+stable	11.342	1.231
Centralize+moving	12.564	4.543
Distributed+stable	19.243	6.431
Distributed+moving	20.686	8.982

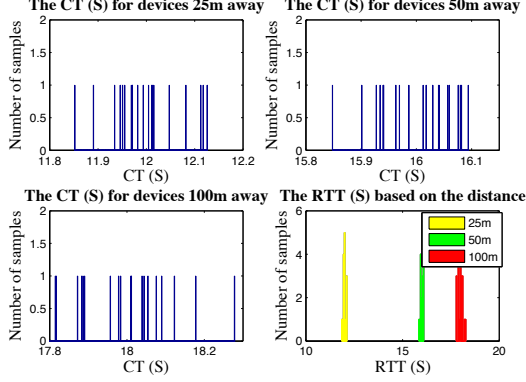


Fig. 9: The CT(s) and RTT(s) based on the distance.

lected global information, enabling quickly to find all nearby potential friends. In a more general case, on the distributed strategy, after collecting information (i.e., SSID, RSSI) of all neighbors, the smartphone decides the working mode by itself. However, when adapting the distributed algorithm, the average time required is around 20s, longer than the average time of the centralized strategy. Moreover, the statistics show that the stable cases are better than the dynamic cases, considering both the mean and variance value. When users moving, the connection time has high variance value, since the moving of user introduces more uncertain factors to the system, the distances between each point are changed over time.

VI. SIMULATION RESULTS

In this section, we simulate the local network in order to evaluate the efficiency of the proposed mode selection algorithm in the large scale scenarios. In each case, we assumed that users joining the network hold the similar interests so that they are potential friends to each other. To have a deep insight into the influence of the user number on the system performance, the number of users joining the network varies from 10 to 100. And we assume the radius for the communication range is 50m. The system should ensure the time for a user finding the first potential friend to be minimum. Therefore, we exam the system performance based on the percentage of users finding an AP to connect in the first round. Then, we evaluate the total number of the iteration for setting up all connections so as to simulate a user finally finding all potential friends nearby. Finally, we exam the percentage of potential friends being found in each iteration, considering totally 80 users in the network. We compare the centralized algorithm with the distributed algorithm, taking the random

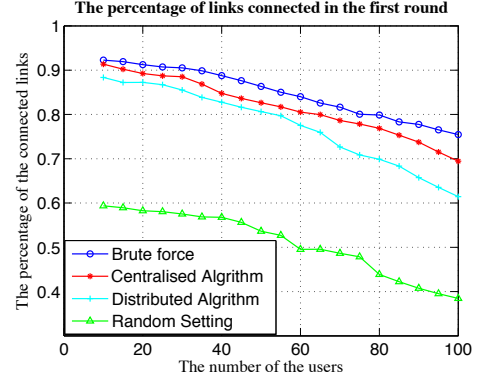


Fig. 10: The percentage of the link get connected in the 1st round.

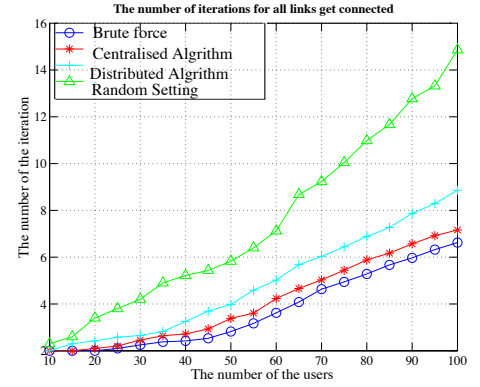


Fig. 11: The number of iterations for all links get connected.

setting as the worst performance benchmark and the brute force method as the optimal performance benchmark.

In Fig. 6, the lines represent that two users of the link are potential friends who have the same interest and the distance between them is within the communication range. In each iteration, the system assigns the proper mode to users where nodes in the AP mode are represented by small red points. Once two users get paired, the line between them is removed. As illustrated in Fig. 6, under the condition that 20 users with the same interest join the network, all users can find all other potential friends after six iterations by adapting the distributed algorithm.

In Fig. 10, in the first round, the state (i.e. AP or client) is initially assigned to each node by the centralized and distributed algorithm, respectively. The simulation result shows that with the number of the arriving users increased, the percentage of the link initially get connected is decreased. This is because when more users concurrently arrive, more of them is assigned to the AP state where other nodes in the AP mode can't connect to. Compared with the random setting, the simulation result shows that our proposed centralized and distributed algorithm ensure that above 80% of the nodes can find a node to connect with in the first round, even the total number of users in the given area is less than 30.

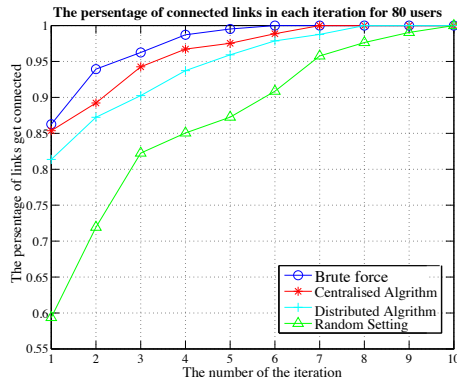


Fig. 12: The percentage of connected links in each iteration for the network containing 80 users.

To enable all potential links to get connected, nodes with unconnected paths will be reassigned new modes. Found friends in the friend list that are represented as connected links should not join the next round calculation. All potential links get connected representing that a user having found all potential friends nearby. In the simulation, the result of required number of iterations to set up all connections is illustrated in Fig. 11. The result shows that the centralized strategy can complete all connection process with less iterations compared to the distributed strategy. The performance of centralized strategy is converged close to the optimal one. Also, we can find when more users are coming, the corresponding iteration increases for setting up all connections. Moreover, the percentage of connected links in each iteration is illustrated in Fig. 12, considering 80 users in the given area. From simulation results, we find that the convergence speed of centralized strategy is faster than the distributed strategy, and it is close to the brute force method. The reason is that centralized strategy knows a priori so that the state of each node is assigned by the central station. Furthermore, the simulation results show that all links can get connected after 6 and 7 iterations, based on the centralized and distributed strategy, respectively. Thus, it further proofs that our proposed approaches is feasible in the large scale scenarios.

VII. CONCLUSION

We develop an Internet independent application which helps people automatically add ‘potential friends nearby’ into each other’s friend list. Since the time cost on the mode switching lows down the performance of the system, we design the set of mode selection mechanisms which dynamically determine smartphones how long to stay in each mode to maximize the number of people who can exchange information. Field experiments have further confirmed that our proposed Wi-friend can efficiently help users to find ‘potential friends nearby’. Moreover, the extensive simulations are conducted to evaluate the performance and effectiveness of our proposed system in the large scale scenarios.

VIII. ACKNOWLEDGMENT

This research is financially supported by the NSFC/RGC Joint Research Scheme with RGC No: N_PolyU519/12 and NSFC Key Grant with project No: 61332004.

REFERENCES

- [1] TECH. www.smh.com.au/news/technology/mobile-phone-users-top-33-billion Online accessed on July 21 2016.
- [2] Facebook help center. <https://www.facebook.com/help/> Online accessed on July 21, 2016.
- [3] Wechat help center. <https://help.wechat.com> Online accessed on July 21, 2016.
- [4] Li, Yong. “Social-aware D2D communications: qualitative insights and quantitative analysis.” *IEEE Communications Magazine* 52.6 (2014): 150-158.
- [5] Zheng, Yu. “Location-based social networks: Users Computing with spatial trajectories.” *Springer New York*, 2011. 243-276.
- [6] Pelusi, Luciana, Andrea Passarella, and Marco Conti. “Opportunistic networking: data forwarding in disconnected mobile ad hoc networks.” *IEEE Communications Magazine* 44.11 (2006): 134-141.
- [7] Huang, Chung-Ming, Kun-chan Lan, and Chang-Zhou Tsai. “A survey of opportunistic networks.” *Advanced Information Networking and Applications-Workshops, 2008. AINAW 2008. 22nd International Conference on*. IEEE, 2008.
- [8] Franklin, Michael, and Stanley Zdonik. “A framework for scalable dissemination-based systems.” *ACM SIGPLAN Notices. Vol. 32. No. 10*. ACM, 1997.
- [9] Gardner-Stephen, Paul, et al. “The serval mesh: A platform for resilient communications in disaster & crisis.” *Global Humanitarian Technology Conference (GHTC), 2013 IEEE*. IEEE, 2013.
- [10] Turkes, Okan, Hans Scholten, and Paul JM Havinga. “Friend-to-friend short message service with opportunistic Wi-Fi beacons.” *2016 IEEE International Conference on Pervasive Computing and Communication Workshops*. IEEE, 2016.
- [11] Beacon Friend Finder. <http://bff-app.com> Online accessed on July 21, 2016.
- [12] Jung, Woo-Sung, Hyochun Ahn, and Young-Bae Ko. “Designing content-centric multi-hop networking over Wi-Fi Direct on smartphones.” *IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2014.
- [13] Trifunovic, Sacha. “WLAN-Opp: Ad-hoc-less opportunistic networking on smartphones.” *Ad Hoc Networks* 25 (2015): 346-358.
- [14] Turkes, Okan, Hans Scholten, and Paul JM Havinga. “Blessed with opportunistic beacons: A lightweight data dissemination model for smart mobile ad-hoc networks.” *MobiCom Workshop on Challenged Networks*. ACM, 2015.
- [15] Kuhn, Fabian, Rogert Wattenhofer, and Aaron Zollinger. “Ad-hoc networks beyond unit disk graphs.” *Workshop on Foundations of mobile computing*. ACM, 2003.
- [16] Dai, Fei, and Jie Wu. “An extended localized algorithm for connected dominating set formation in ad hoc wireless networks.” *IEEE transactions on parallel and distributed systems* 15.10 (2004): 908-920.
- [17] Kuhn, Fabian, and Roger Wattenhofer. “Constant-time distributed dominating set approximation.” *Distributed Computing* 17.4 (2005): 303-310.
- [18] M. Garey and D. Johnson. “Computers and Intractability: A Guide to the Theory of NP-completeness.” WH Freeman & Co., 1979.
- [19] Wu, Jie, and Hailan Li. “On calculating connected dominating set for efficient routing in ad hoc wireless networks.” *The 3rd international workshop on Discrete algorithms and methods for mobile computing and communications*. ACM, 1999.