Location-Assisted Energy-Efficient Content Search for Mobile Peer-to-Peer Networks

Yu-Chih Tung
Department of Computer Science and Information Engineering
National Taiwan University, Taiwan

Kate Ching-Ju Lin
Research Center for Information Technology Innovation
Academia Sinica, Taiwan

***Abstract*—As mobile devices have become more powerful and ubiquitous in our daily life, sharing content objects among mobile platforms has become increasingly popular. Without the help of server infrastructures, clients usually form a mobile peer-to-peer (P2P) system as an ad-hoc network, and discover content objects by flooding query to neighboring peers. Such a flooding-based query method consumes communication energy of all relay users. Fortunately, we notice that recent emerging location-based services (LBS) have encouraged part of clients to provide their location information. Hence, in this paper, we propose a framework, called LocP2P (Location-assisted P2P), that better utilizes partial location information to provide peers a candidate list of content owners and the corresponding potential routing paths. Our evaluation results demonstrate that LocP2P can reduce the number of query flooding, and thereby save up to 50% energy consumption for content search in a mobile peer-to-peer system.**

I. Introduction

With the growing popularity of Peer-to-Peer (P2P) systems, clients can share objects, e.g., files or multimedia content, on the Internet in a distribution manner. On the other hand, as mobile devices have become more powerful (e.g., higher network bandwidth, storage and computation capacity) and ubiquitous in our daily life, caching data on mobile platforms has become increasingly popular. Therefore, except searching content on the Internet, clients have an increasing demand for forming a *mobile P2P* network and sharing content among mobile devices over wireless networks [1] [2].

Without the help of server infrastructures, clients usually form a mobile P2P system as an ad-hoc network. In such an unstructured network, clients must discover content objects by flooding query (requests) to neighboring peers until getting the response from any client who holds the requested objects. Since neighboring users must help forward requests, such a flooding-based query method consumes communication energy of all relay users. However, due to a limited battery lifetime of mobile devices, power consumption is always one of the most critical design issues of mobile applications. Hence, the goal of this work is to develop a mobile P2P system that can improve energy efficiency of object discovery.

One intuitive solution to avoiding flooding is to track location information of each mobile peer [3]–[5], and build a network topology in a central server for looking up content owners and the corresponding routing paths for the requesting user. Unfortunately, it also costs energy consumption to keep tracking the ud-to-date location information of mobile peers. However, we notice that recent emerging location-based services (LBS), e.g., Google map, Twitter, and Facebook, have requested their clients to upload their location information. If we can better utilize these location information to help search mobile content, peers in mobile P2P systems can avoid flooding query to other neighboring users and, in turn, reduce energy consumption.

Specifically, if clients do not consume extra power to track position for content search, yet only provide their location information when they want to access location-based services of interest spontaneously, could we exploit those information to improve energy efficiency of content search? Some challenges of this concept are that not all the mobile peers would access those LBS, and could not upload their location information. In addition, even if part of mobile peers upload their locations as accessing LBS, the information cached in the database could be out-of-date. If the server provides an out-of-date query result to peers, mobile peers might consume additional power to validate the recommended content owner and routes, and then use conventional flooding-based query when the suggested information is incorrect, resulting even higher energy consumption.

Hence, in this paper, we propose a framework, called LocP2P (Location-assisted P2P), to manage the location information and content objects of partial clients. LocP2P is designed to index location information as well as content objects in a distributed manner, and recommend mobile clients a list of potential owners and the corresponding routing paths. Our evaluation results demonstrate that, by better utilizing those partial location information, LocP2P can reduce the number of flooding requests, and save up to 50% energy consumption for object query in a mobile peer-to-peer system.

The remainder of this paper is organized as follows. Section II provides a review of related works on mobile peer-to-peer systems and energy-efficient network protocols. Section III describes the framework of the proposed LocP2P. In Section IV, we evaluate the performance of LocP2P. Finally, Section V concludes this paper.

II. Related Work

There have been many work, DSRP [6], Ekta [7] [8], on improving the efficiency of content search in a purely distributed environment. They apply the concept of Distributed Hash Tables (DHTs) to eliminate the cost of flooding-based

978-1-61284-937-9/11/$26.00 ©2011 IEEE

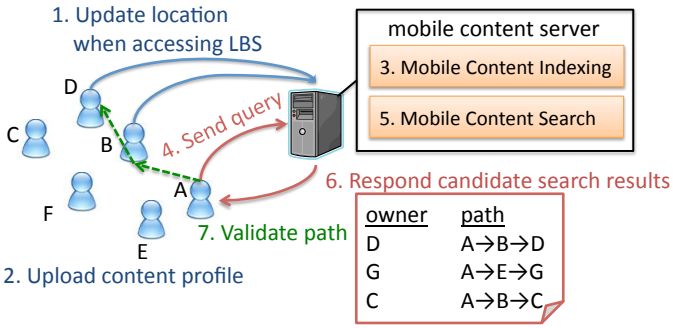
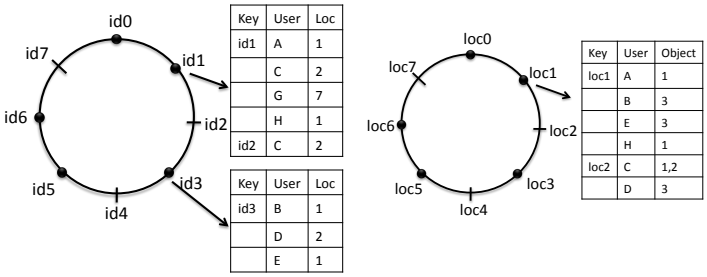


Fig. 1. LocP2P Framework



(a) hashed by object ID (b) hashed by location

Fig. 2. Distributed hashed table

routing in a mobile network. Our work however targets on better utilizing the users' location information that is already cached in a central server, and reducing the requirement of querying over an ad-hoc network. Therefore, our work can be integrated with any other purely distributed methods, which can further improve the efficiency of content search if the location information does not help.

On the other hand, several approaches, such as GCLP [3], Globase.KOM [4] and GeoKad [5], proactively ask users to report their location, and build an explicit lookup structure based on the full knowledge of location information to avoid flooding. By contrast, our work lets users update their location spontaneously without any extra energy cost, and harnesses the location of partial users to enhance energy efficiency of content search as much as possible.

Some other network protocols are proposed to improve energy efficiency of data transmission for mobile devices. CoolSpots [9] enables a wireless mobile device to automatically switch between multiple interfaces, such as WiFi and Bluetooth, to reduce energy consumption based on demanding bandwidth and transmission range. Cool-Tether [10] harnesses the cellular radio links of neighboring smartphone, and builds them as a WiFi hotspot to provide energy-efficient connection for mobile devices that do not have cellular connection. Bartendr [11] is an energy-aware scheduling algorithm that enables each 3G client to receive data only when the signal strength is good, and thus reduces power consumption per bit transmission. Orthogonal to the above protocols, LocP2P targets on reducing energy consumption of mobile content discovery.

III. LocP2P Framework

In conventional mobile peer-to-peer systems, when a client attempts to download an object of interest, it could flood the query to all neighboring users, and wait for the responses indicating the owners of the request object alone with the corresponding routing path. However, such a flooding-based query method consumes all neighbors' power for each query. Motivated by energy inefficiency caused by flooding query, the goal of our design is to harness the location information of users to reduce power consumption for content search.

Figure 1 illustrates our system framework. Suppose part of clients would report their location information to the servers when they are running other applications, such as Facebook or Twitter. Since users spontaneously report location information when accessing those location-based services (LBS) even without our mechanism, we assume that the energy consumption of reporting the location information can be neglected in our framework. Specifically, our framework does not force users to report extra location informations, but just attempts to best utilize those coarse (i.e., might be out-of-date) location information in order to recommend clients a list of *candidate search results*, including the potential owners alone with the corresponding routing paths.

We consider an environment where each client can access the server deployed in the Internet by 3G or WiFi connection, while attempts to search spatial data or collect social-based content from their neighboring mobile users. Clients can send the request to the servers, which help look up the database and compile the *candidate search results*. Once the client receives the results from the servers, it can evaluate the correctness of candidate results by probing the suggested routes. If all the suggested routes are out-of-date and do not exist anymore, it can then flood the request as what it does in conventional mobile P2P systems. Thus, the search performance of the proposed scheme must not be worse than flooding-based query. The extra energy consumption generated by our scheme includes sending query to the server, receiving the response from the server, and probing the candidate results. However, if the candidate results contain correct owner/routing information, users can avoid significant energy consumption of flooding. In the following sections, we will describe the components of server and client design in more details, and demonstrate when and how such a location-assisted content search scheme improves the battery life of mobile devices.

A. Mobile Content Indexing

Our system relies on the *mobile content server* that is formed by one or a cluster of server nodes[1], and can manage two types of information: content profile and location of each client. We believe that, with the growth of cloud computing, such functionality can be easily implemented in the current

[1]We use "server nodes" to indicate any machine of the mobile content server cluster, and represent mobile users as "clients" or "peers".

public platforms, like [12], with no or neglected costs. When a client joins the system, it compiles its content profile, which contains a list of the hashed keys (IDs) of its objects, and uploads that profile to the server. It then only needs to notify the server of the updated items if it adds/deletes some items to/from the content profile. Besides, when clients access other location-based services, the location information will be duplicated in our mobile content server.

Considering the scalability, we adopt a well-known indexing algorithm, distributed hashed table (DHT) [13], which allows a number of mobile content server nodes to manage the information of all mobile peers in a distributed manner. The original design of DHT hashes data based on a single major key, e.g., usually the hashed value of content objects. However, since our mobile content server must cache two hashed keys, i.e., location and content objects, we must select a proper attribute as our major key. In our design, we choose user location as the major key, and use the object ID as the secondary key. Because location coordination is a two-dimension continuous space, we divide locations into multiple grids. Each grid is assigned an one-dimension coordination by using dimension reduction technologies, such as [14] or [15], which guarantees that neighboring coordinations in a two-dimension space are also near to each other in the one-dimension space. Thus, we can use such one-dimension coordination as the hashed location key, and let clients close to each other be assigned similar location keys. To balance the number of clients in each grid (hashed key), we allow each grid to have a different size proportional to their client density.

The rationale behind this strategy is that the DHT algorithm would cache data with similar hashed keys in the same (or nearby) server nodes. In our applications, content objects are only useful for the client who sends the query if those objects are held by its k-*hop* neighboring peers, where k is the hop limit of flooding. Hence, for the mobile content server, it is more important to cache data with similar location in a single server node because the server can compile the candidate search results by only looking up data in a few server nodes that maintain the content objects located in the neighboring area of the request client. Consider the example shown in Figure 2(a), the information is cached based on the hashed key of content objects. When client A requests for object 3, the server can find all owners of object 3 at node $id3$. However, those owners might locate far from the client who sends the request (e.g., user C), and, thereby, cannot forward the object to that client. Even though the server can find that peer B locates nearby client A, it is also difficult to find relay nodes that can help forward data between peers A and B when users at location 1 are distributed in different server nodes. That is, unlike DHT-based P2P systems on the Internet, mobile P2P systems must filter the information that satisfies the location constraint. If we index data by using location information as the major key, as shown in Figure 2(b), the server can look up the client's neighboring peers at the same server node (e.g., $loc1$, client A's location), and then search whether those neighbors hold the objects of interest.

B. Location-based Mobile Content Search

When a client requests for an object of interest, it forwards the query to the server. The server can then find the node that is in charge of caching data with the corresponding hashed location key. Consider the same example shown in Figure 2(b), if client A searches for object 3, the server finds that client A locates at $loc1$ and looks up data cached at server node $loc1$, which is in charge of the neighboring area of location $loc1$, i.e., locations $loc1$ and $loc2$. The server can decide the size of area it wants to search. For example, assume that the server attempts to search the content objects located within a $500m \times 500m$ square field area around the request client A. If the grid size of location $loc1$ is a $250m \times 250m$ square filed, the server should also search for data in the surrounding grids, e.g., $loc2$, $loc3$, etc.

In this example, the server will find that peers (owners) B, D, and E hold the request object 3. We let $\mathcal{M}$ denote the collection of all candidate owners, e.g., $\mathcal{M} = \{B, D, E\}$. Next, it forms all peers in the searching area, i.e, locations $loc1$, as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of peers in the searching area, and $\mathcal{E}$ is the set of edges that connect two peers if they are within the transmission range of each other. For simplicity, we assume that each mobile client has the same transmission range. Based on the graph $\mathcal{G}$, the server can find the shortest path $p(m)$ between the requesting client and each owner peer $m \in \mathcal{M}$ based the cached location informations. Let $t_{loc(c)}$ denote the time-stamp of client c's latest updated location information. Assume that a path $p(m)$ includes the relay peers $(c_1 \rightarrow c_2 \cdots \rightarrow c_n)$; we let $p(m) = \{c_1, c_2, \cdots, c_k\}$. We then set the time-stamp of path $p(m)$ as $min_{c \in p(m)} t_{loc(c)}$. That is, the time-stamp of a path is the same with the time-stamp of the relay node that has the most out-of-date location information. After lookup, the server responds the candidate search results, including the owner set $\mathcal{M}$, the corresponding path to each owner $p(m)$, and the time-stamp of each path, to the requesting client. Table I shows the example candidate search results when client A requests for object 3.

TABLE I
CANDIDATE SEARCH RESULTS

owner	path	time-stamp
B	$(A \rightarrow H \rightarrow D \rightarrow B)$	7
D	$(A \rightarrow D)$	5
E	$(A \rightarrow D \rightarrow E)$	3

C. Route Validation and Direction Suggestion

Recall that the cached location information might be out of date. Therefore, once the client receives the candidate search results from the server, it must verify whether the recommended routing paths are correct or not. Hence, the client can trace (probe) the route, and check whether the probing message can reach the candidate owner. If the probing message fails to reach the candidate owner, we say that the recommended route is out-of-date. For the example shown in Table I, the path for owner B might be incorrect if peer H already left location $loc1$. Note that the probing message used to verify the routing paths consumes extra energy. Hence, if

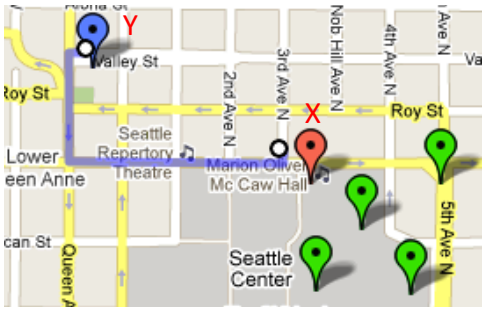


Fig. 3. Direction Suggestion

TABLE II
SIMULATION PARAMETERS

symbol	value	description
N	100	Number of total users
TR	100 (m)	Transmission Range
Ds	500 (min)	Simulate duration
H_max	2	Maximum relay hop count
$avg(T_r)$	150 (sec)	Average request interval
$avg(T_u)$	30 (sec)	Average location update interval
E_r	100	Location-enabled ratio
K	3	Number of candidate search results
Δ	50 (min)	Threshold of perfiltering time delay

the server suggests multiple candidates, the client can probe those recommended routes one-by-one until it finds a correct one.

In order to avoid the energy consumption of probing routes, we test the following three probing strategies: (1) shortest-path first, (2) latest-path first, and (3) the hybrid scheme, and evaluate their performances in Section IV. In the shortest-path first scheme, the client probes the candidate routes in the increasing order of path length. In the latest-path first scheme, the client probes the routes in the decreasing order of the route's time-stamp. In the hybrid scheme, the client verifies the shortest path, the latest path, the next shortest path, the next latest path, and so on. Finally, if all the routing paths fail to reach any candidate owner, the client then uses the conventional flooding scheme to query the object of interest.

Another potential benefit of our LocP2P is that the mobile content server could find the results that cannot be discovered by flooding. Consider the example shown in Figure 3. Client X is interested in object 5, which is held by owner Y. When the density of peers is too low or there exists a routing "hole" in the network topology, the query cannot be flooded to owner Y. However, owner Y indeed locates very close to the request client X. Hence, our framework can integrate with any online map database, e.g., Google map, and recommend the direction between the client and the content owner. If the client cannot find any routing path to the owner, it can move along the suggested direction toward the owner until it reaches the one-hop transmission range of the owner.

IV. Performance Study and Discussion

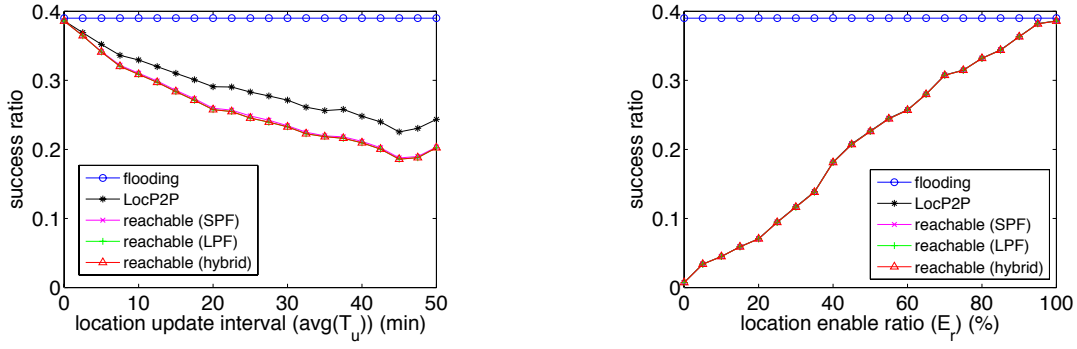
In order to evaluate the performance of LocP2P, we adopt SLAW [16] as the mobility model that mimics the properties of human mobility such as realistic waypoint selection and the proper inter-contact time of visiting the same place. We used log-based user profiles collected from Last.fm [17], a database that tracks listening habits of music content. We collected profiles for 100 users (denoted by N). For each user, the data set recorded the 100 songs he/she had listened to the most. For cross validation, we randomly divide each user's favorite songs into a training set (50 songs) and a test set (50 songs). Let the training set of songs be cached in each users buffer. Each user then requests songs in the test set to evaluate the performance of the query performance in LocP2P. In the following simulations, if we do not specify, the default parameter settings are summarized in Table II.

We compare the performance of the above schemes in terms of the following performance metrics:

- Flooding success ratio: The ratio of the number of successful searches to the total number of requests in the flooding-based scheme.
- LocP2P success ratio: The ratio of the number of successful searches to the total number of requests in LocP2P. The successful search here is defined as that the subset of candidate owners suggested by the mobile content server can be also found by flooding, no matter the suggested routing paths are useful or not.
- LocP2P reachable success ratio: The ratio of the number of successful and reachable searches to the total number of requests in LocP2P. The successful and reachable search is defined as that the search is successful and, at the same time, the routing paths provided by the mobile content server are correct and can help the client connect to the owner.
- Truth positive ratio: The ratio of LocP2P reachable ratio to the LocP2P success ratio, which indicates how many portions of the suggestions from the server are useful. Because our scheme lets clients consume extra power to validate the paths, it is a waste if clients still need to execute flooding-based query when the validation of paths fails. Thus, we want this ratio as large as possible.
- Energy consumption saving: The ratio of energy consumption of LocP2P to that of pure flooding-based query. We use the number of message exchange as energy consumption in our evaluation.

A. Impact of User Location Update Behavior

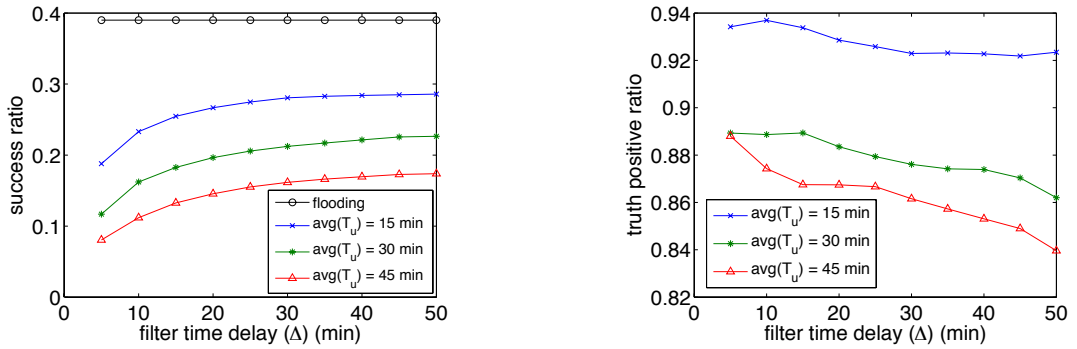
Since the performance of LocP2P is closely related to user behaviors of accessing the location-based services, we study the impact of the user behaviors in two aspects: average location update interval ($avg(T_u)$) and location-enabled ratio (E_r). Figures 4(a) and 4(b) show the results of various settings of $avg(T_u)$ and E_r. Figure 4(a) shows that both of the LocP2P success ratio and the reachable success ratio increase when the average location update frequency increases because, if clients update their location information more frequently, the information cached in the server is more up-to-date. However, we can still achieve 0.2 reachable success ratio, which is half of the pure flooding success ratio. That is, if the objects of



(a) Location Update Interval

(b) Location-Enabled Ratio

Fig. 4. Impact of Location Update Behavior



(a) Reachable Success Ratio

(b) Truth Positive Ratio

Fig. 5. Impact of Time-stamp Pre-filtering

interest indeed locate within the neighboring area, LocP2P can save nearly 50% of query flooding. Figure 4(b) shows that the LocP2P success ratio increases as the location-enabled ratio increases because, when more users update their current location, the server can construct a more complete location/content database, and compile a more accurate candidate search result. Another interesting observation is that, in these simulations, all of three routing strategies achieve the similar reachable success ratio. This is because that the shortest path contains the fewest relay peers, and, usually, has a lower probability to include a relay peer with an out-of-date location information.

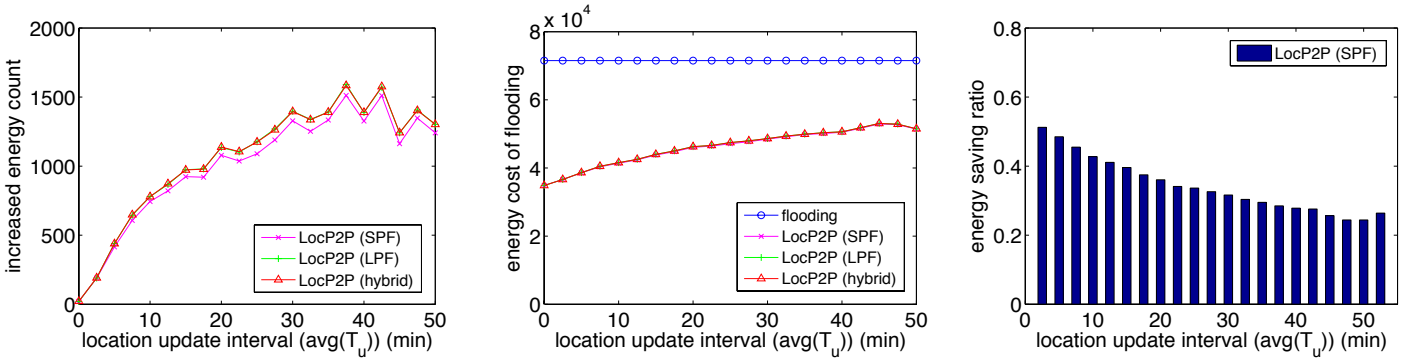
B. Impact of Time-stamp Pre-filtering

Since our scheme requires each client to validate the suggested routing paths, we define that the query is a truth positive result if at least one of the suggested routing paths can reach one of the candidate owners. Specifically, if the query is a false positive, i.e., cannot reach the owners, the requesting client not only needs to perform path validation, but also flood the request. Hence, we are interested in how the time-stamp of a routing path affects the truth positive ratio. We evaluate the performance of LocP2P when the mobile content server filters (discards) the location information that is older than a time-delay Δ, i.e., $t_{loc} < t_{current} - \Delta$, where $t_{current}$ is the current time.

Figures 5(a) and 5(b) show the LocP2P reachable success ratio and the truth positive ratio, respectively, under various thresholds of filtering time-delay. The figure shows that LocP2P provides a higher reachable success ratio when clients update the location information more frequently, i.e., smaller location update interval. In addition, we observe that the reachable success ratio decreases when we reduce the threshold of filtering time-delay. However, by keeping more up-to-date location information, we can improve the truth positive ratio, as shown in Figure 5(b), and avoid consuming extra energy of mobile devices to validate suggested paths. The figure also reveals that the suggested routes have a high probability to be a valid path even if the location update interval is smaller than the threshold of filtering time-delay, which means that the server almost does not filter out cached information. For instance, when the filtering time-delay is set to 50 minutes, which is higher than the average location update intervals ($T_u = 15, 30, 45$), the truth positive ratio can still be higher than 84%. That is, only 16% of path validations fail to reach the suggested owners, and require the clients to perform flooding-based query. Thus, by choosing a proper threshold of filtering time-delay, LocP2P can provide a higher reachable success ratio, while also improve the truth positive ratio, i.e., avoiding extra energy consumption.

C. Energy Saving

For simplicity, we assume that each query message and control message have similar packet length, and use the number of message exchanges to represent energy consumption. In



(a) Increased Energy Cost

(b) Energy Cost of Flooding

(c) Energy Consumption Saving

Fig. 6. Energy Consumption

flooding-based scheme, each query forwarding is counted as a message exchange. Figure 6(a) shows that extra energy consumption used to forward/receive the query/response to/from the server and used to probe the suggested routing paths. On the other hand, Figure 6(b) shows the energy cost of flooding in all comparison schemes. The figures demonstrate that the energy cost that LocP2P must pay for communicating with the server and validating paths is relatively small as compared to the reduced energy consumption of flooding due to the reachable suggested routes. On the other hand, Figure 6(c) computes the ratio of the total energy saving in LocP2P and that in the pure flooding scheme. The figure indicates that LocP2P can save up to 50% of the total energy consumption on content discovery as compared to the pure flooding-based query scheme. This is because the flooding-based scheme forwards query to all neighboring users, consuming power of each relay peer. Hence, by utilizing the location information, LocP2P can avoid the number of flooding requests, and thereby reduce energy consumption significantly.

V. Conclusion

In this paper, we proposed LocP2P, a reasonable scheme that reduces the energy cost caused by flooding content query in mobile peer-to-peer systems. Such a server-assisted scheme harnesses location information of peers to suggest user candidate search results, and allow peers to validate the recommended content owner in order to avoid flooding process. We conduct performance study via simulations to evaluate how user behaviors and parameter settings affect the efficiency of LocP2P. The simulation results also show that our scheme can save up to 50% energy consumption even when clients do no update their location aggressively. In the other aspect, by examining the human mobility model, we notice that users usually cluster together as several independent sets, where clients cannot discover each other by flooding. LocP2P provides another potential advantage that assists users in moving toward the candidate content owner by suggesting map directions. In this situation, LocP2P not only saves energy consumption, but also help discover content objects that cannot be found by flooding in mobile peer-to-peer systems.

References

[1] W. He, Y. Huang, K. Nahrstedt, and B. Wu, "Message propagation in ad-hoc-based proximity mobile social networks," in *8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, 2010.

[2] F. S. Tsai, W. Han, J. Xu, and H. C. Chua, "Design and development of a mobile peer-to-peer social networking application," *Expert Syst. Appl.*, vol. 36, pp. 11 077–11 087, October 2009.

[3] J. Tchakarov and N. Vaidya, "Efficient Content Location in Wireless Ad Hoc Networks," in *IEEE Internation Conference on Mobile Data Management*, 2004.

[4] A. Kovacevic, N. Liebau, and R. Steinmetz, "Globase.kom - a p2p overlay for fully retrievable location-based search," in *IEEE International Conference on Peer-to-Peer Computing*, 2007.

[5] M. Picone, M. Amoretti, and F. Zanichelli, "Geokad: A p2p distributed localization protocol," in *IEEE Pervasive Computing and Communications Workshops (PERCOM Workshops)*, 29 2010.

[6] Y. C. Hu, S. M. Das, and H. Pucha, "Exploiting the synergy between peer-to-peer and mobile ad hoc networks," in *USENIX HotOS*, 2003.

[7] H. Pucha, S. Das, and Y. Hu, "Ekta: an efficient dht substrate for distributed applications in mobile ad hoc networks," in *IEEE Workshop on Mobile Computing Systems and Applications*, 2004.

[8] N. Dutta, "A Peer to Peer Based Information Sharing Scheme in Vehicular Ad Hoc Networks," in *International Conference on Mobile Data Management (MDM)*, 2010.

[9] T. Pering, Y. Agarwal, R. Gupta, and R. Want, "CoolSpots: Reducing the Power Consumption of Wireless Mobile Devices with Multiple Radio Interfaces," in *ACM MobiSys*, 2006.

[10] A. Sharma, V. Navda, R. Ramjee, V. N. Padmanabhan, and E. M. Belding, "Cool-Tether: Energy Efficient On-the-Fly WiFi Hot-Spots Using Mobile Phones," in *ACM CoNEXT*, 2009.

[11] A. Schulman, V. Navda, R. Ramjee, N. Spring, P. Deshpande, C. Grunewald, K. Jain, and V. N. Padmanabhan, "Bartendr: a Practical Approach to Energy-Aware Cellular Data Scheduling," in *ACM MobiCom*, 2010.

[12] "Google App Engine," http://code.google.com/appengine/.

[13] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," in *ACM SIGCOMM*, 2001.

[14] Y. Fu and T. S. Huang, "Locally Linear Embedded Eigenspace Analysis," *IFP-TR, UIUC*, vol. 2005, pp. 2–05, 2005.

[15] J. B. Tenenbaum, V. d. Silva, and J. C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[16] K. Lee, S. Hong, S. Kim, I. Rhee, and S. Chong, "Slaw: A New Mobility Model for Human Walks," in *IEEE INFOCOM*, 2009.

[17] "Last.fm," http://www.last.fm.