Leveraging Social Networks for P2P Content-based File Sharing in Mobile Ad Hoc Networks

Kang Chen and Haiying Shen
Department of Electrical and Computer Engineering
Clemson University
Clemson, USA
{kangc, shenh}@clemson.edu

Haibo Zhang
Computer Science and Computer Engineering Department
University of Arkansas
Fayetteville, USA
hxz009@uark.edu

***Abstract*—Current P2P file sharing methods in mobile ad hoc networks (MANETs) can be classified into three groups: flooding-based, advertisement-based and social contact-based. The first two groups of methods can easily generate high overhead and low scalability, and the third group fails to consider the social interests (content) of mobile nodes, which otherwise can improve file searching efficiency. In this paper, we propose a P2P content-based file sharing system for MANETs. The system uses an interest extraction algorithm to derive a node's interests from its files for complex queries. For efficient file searching, it groups common-interest nodes that frequently meet with each other as communities. Further, it takes advantage of node mobility by designating stable nodes, which has frequent contact with community members, as community coordinators for intra-community searching, and highly-mobile nodes as community ambassadors for inter-community searching. An interest-oriented file searching scheme further enhances the file searching success rate. We first deployed our system on the real-world GENI Orbit testbed with a real trace and then conducted experiment on the ns2 simulator with both real trace and simulated disconnected and connected MANET scenario. The test results show that our system significantly lowers transmission cost and improves file searching success rate compared to current methods.**

***Index Terms*—MANETs, P2P, File sharing, Social Network.**

I. Introduction

In the past few years, more and more people have begun to use personal mobile devices such as laptops, PDAs and smart phones (e.g., iPhone, BlackBerry and Android phones). Indeed, the number of smart-phone users is expected to reach around 300 million by 2013 [1]. The incredibly rapid growth of mobile users is leading to a promising future in which they can freely share files between each other whenever and wherever. Currently, mobile users interact with each other and share files via an infrastructure formed by geographically distributed base stations. The number of mobile searching users in infrastructure-based wireless data sharing is estimated to reach 901.1 million in 2013 from 266.0 million in 2006 [2]. However, the infrastructure based server-client model has difficulty in dealing with the challenges posed by the daily increase of user number. Many users accessing one base station leads to traffic congestion and delayed response. Also, users may find themselves in an area without wireless service (e.g., mountain areas) or under emergency situations like disaster and wars. The infrastructure-based model becomes a major impediment to the vision of large-scale and pervasive mobile file sharing.

The P2P file sharing model makes large-scale networks a blessing instead of a curse, in which nodes share files directly with each other without relying on a centralized server. Wired P2P file sharing systems (e.g., BitTorrent [3] and Kazaa [4]) have already become a popular and successful paradigm for file sharing among millions of users. Recent survey shows that more than 50% of the files downloaded and 80% files uploaded on the Internet are through P2P networks [5]. The successful deployment of P2P file sharing systems and the impediments to file sharing in large-scale MANETs make the P2P file sharing over MANETs (P2P MANETs in short) an inevitable developing trend for the promising vision of pervasive file sharing for mobile users.

Traditional methods supporting P2P MANETs are either flooding-based [6]–[9] or advertisement-based [10]–[12]. The former relies on flooding for file searching. However, it generates high overhead due to a tremendously high volume of transmitted messages, and local broadcasting cannot guarantee file discovery. In the latter methods, nodes advertise their available files, build content tables from received advertisements, and forward file requests to the nodes with high probability of possessing the files. However, they cannot guarantee file discovery because of possible expired routes in the content tables caused by transient network connections.

Recently, social network has been exploited to facilitate routing or content publishing in MANETs and Delay Tolerant Networks (DTNs) [13]–[19]. The social network possesses a property (P1) that nodes (i.e., people) usually exhibit certain movement patterns (e.g., local gathering, diverse centralities and skewed visiting preferences). These methods take advantage of this property to improve the efficiency of message forwarding. However, these methods only consider end-to-end forwarding but fail to take into account other properties of social networks to facilitate content sharing. Recent studies on social networks revealed that:

- (P2) Users usually have a few file interests that they visit frequently [20] and a user's file visit pattern follows a power-law distribution [21].
- (P3) Users with common interests tend to meet with each other more often than with other users [22].

By leveraging these properties of social networks, we propose Social network based P2P cOntent file sharing in mObile ad-hoc Networks (SPOON) with four components as shown in Figure 1.

(1) Based on P2, we propose an *interest extraction algorithm* to derive a node's interests from its files. The node interest facilitates queries in content-based file sharing. It is also required by other components of SPOON.
(2) We refer to a collective of nodes that share common interests and frequently meet each other as a *community*. According to P3, a node has high probability to find its interested files in its community. If it fails, according to P1, the node can rely on nodes frequently travel to other communities for file searching. Thus, we propose the *community construction algorithm*. It builds communities to enable users to efficiently retrieve files using intra- and inter-community communication.
(3) According to P1, we propose *node role assignment algorithm*, which takes advantage of node mobility for efficient file searching. The algorithm designates a stable node that tightly connects others in its community as the community coordinator, which guides intra-community searching. For each foreign community, a node that frequently travels to it is designated as the community ambassador for inter-community searching.
(4) We propose an *interest-oriented file searching and retrieval scheme* which includes an *interest-oriented routing algorithm (IRA)* and utilizes above components. Based on P3, IRA makes forwarding decision by considering the probability of meeting interest keywords rather than nodes. The searching scheme has two phases: intra- and inter-community. In the intra-community search, a node first queries nearby nodes, then relies on coordinator to search the entire home community. If it fails, the inter-community searching is executed, in which the ambassador matched to the query sends the query to a foreign community. When a file is found, it is sent back through either the search path or IRA.

SPOON is novel in that it leverages social network properties of both node interest and movement pattern. First, it classifies common-interest and frequently-encountered nodes into social communities. Second, it considers the frequency a node meets different interests rather than different nodes in routing for enhanced searching success rate. Third, it chooses the highly mobile nodes that travel frequently to foreign communities as ambassadors, so that a query can be directly forwarded to the community of the queried file. Consequently, SPOON achieves high efficiency in file searching.

The rest of the paper is arranged as follows. Section 2 provides an overview of the related works. Section 3 presents the design of the components of the SPOON system. In Section 4, the performance of SPOON is evaluated in comparison with other systems through simulations. The last section presents concluding remarks and future work.

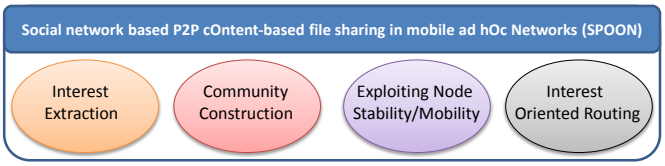


Fig. 1. Components of SPOON.

II. Related Work

A. Flooding-based Searching Methods

In the flooding-based methods, 7DS [6] is one of the first approaches to port P2P technology to mobile environments. It exploits the mobility of nodes within a geographic area to disseminate web content among neighbors. Passive Distributed Indexing (PDI) [8] is a general-purpose distributed search service. It uses local broadcasting for content searching and sets up content indexes on nodes along the reply path to guide subsequent searching. Klemm *et al.* [7] proposed a special-purpose on-demand searching and file transferring algorithm based on an application layer overlay network, which transparently aggregate query results from other peers to eliminate redundant routing paths. Anna Hayes *et al.* [9] extended the Gnutella system to mobile environments and proposed the use of a set of keywords to represent user interests. However, these flooding-based methods produce high overhead due to a high volume of traffic. Also, local broadcasting used in some methods cannot guarantee file searching success.

B. Advertisement-based Searching Methods

Tchakarov and Vaidya [10] proposed the Geography-based Content Location Protocol (GCLP) for efficient content discovery in location-aware ad hoc networks. It disseminates contents and requests in crossed directions to ensure their encountering. P2PSI [11] is a hybrid file sharing system that comprises both advertisement (push) and discovery (pull) processes. Each file holder regularly broadcasts an advertisement message to inform surrounding nodes about its shared files. The discovery process locates the desired file, and leaves pheromone to help subsequent search requests. Repantis and Kalogeraki [12] proposed a file sharing mechanism in which nodes use the Bloom filter to build content synopses of their data and adaptively disseminate them to other nodes to guide queries. Though the advertisement-based methods reduce the overhead of flooding-based methods, they still generate high overhead for advertising. Also, the methods cannot guarantee the success of file searches, especially when the routes expire due to node mobility.

C. Social Network-based Routing and Sub/Pub Services

Recently, social networks have been utilized in routing or content publishing algorithms in MANETs or DTNs. Since the node movement in a social network usually follows a certain pattern, social network based routing algorithms [13]–[17] consider node contact frequency, predict future contact possibility, and choose the node with the highest possibility of successfully delivering a packet as the next forwarder in

routing. However, these algorithms cannot be directly used for content-based file searching since the destinations are unknown in this service.

Considering the long-term neighboring relationship between nodes in a social network, MOPS [18] provides content-based sub/pub service. It groups nodes with frequent contacts and selects nodes that connect nodes from different groups as brokers, which is responsible for inter-community communication. SocialCast [23] publish contents to subscribers by calculating the utility of a node for each interest based on its mobility and co-location records. However, the interest in SocialCast is only an indication of a destination. ContentPlace [19] defines social relationship based communities and a set of content caching policies. Specifically, each node calculates a utility value for each encountered object regarding its connected communities and caches these objects in a highest utility value first manner. However, the three methods only leverage the contact property of social networks, but fail to consider the interest property in social network for further performance improvement. Purely considering contact for content searching/publishing may lead to low efficiency due to frequent inter-community communications. SPOON is novel in that it leverages both properties as described previously.

III. The Design of SPOON

A. Interest Extraction

It was found that users usually have a few file interests that they visit frequently in a file sharing system. Specifically, for the majority of users, 80% of their shared files fall into only 20% of total file categories [20]. Thus, SPOON derives the interests of a node from its files.

To derive its interests, a node infers keywords from each of its files using the document clustering technique [24]. If a group of keywords appear frequently in a number of files, these files belong to a category, which can be used to indicate the node's interest. Specifically, a node derives a file vector from the metadata of each of its files, denoted by $v_F = (t_0, w_{t_0}; t_1, w_{t_1}; t_2, w_{t_2}; ...; t_m, w_{t_m})$. In previous equation, t_i and w_i $(0 \leq i \leq m)$ denote a keyword and its weight that represents the importance of the keyword in describing the file. We adopt the method in the text retrieval literature [25] to calculate the weight of a keyword in file F using the following formula.

$$w_t = 1 + log(n_t), \tag{1}$$

where n_t refers to the number of occurrences of keyword t. In order to make the keyword weights comparable, we further normalize the weights by:

$$w_t = w_t / \sum_{i=0}^{m} w_{t_i}. \tag{2}$$

To calculate the similarity of two arbitrary vectors, say $v_1 = (t_0, w_{1_{t_0}}; t_1, w_{1_{t_1}}; t_3, w_{1_{t_3}})$ and $v_2 = (t_0, w_{2_{t_0}}; t_2, w_{2_{t_2}}; t_4, w_{2_{t_4}})$, we first generate the *common vector* for each vector. It consists of their common keywords and corresponding weights in their own vectors. For example, the common vector of v_1 and v_2 is $(t_0, w_{1_{t_0}})$ for v_1 and $(t_0, w_{2_{t_0}})$ for v_2. We then use the following formula to calculate the similarity between v_1 and v_2:

$$sim(v_1, v_2) = \frac{\sum_{i=1}^{m'} w_{1_i} \times w_{2_i}}{m'}, \tag{3}$$

where m' is the total number of common keywords and w_{1_i} and w_{2_i} represent the weights of the i^{th} common keyword in two common vectors, respectively.

After retrieving the file vector v_F for each of its files, a node classifies its files to derive its interest groups. It creates a file similarity matrix $A = sim(v_i, v_j)$ $(1 \leq i \;\&\; j \leq \tilde{m})$, where $\tilde{m}$ is the number of files the node has. The matrix stores the similarity value of each pair of files. We pre-define a threshold of the similarity denoted by T_s. If $sim(v_i, v_j) > T_s$, file i and file j are classified into the same interest group. A node scans the matrix to classify all files into groups. We allow partial similarity transitivity in the interest group extraction process that once a file satisfies the similarity threshold with at least half of files in a group, it is granted the membership of the group. Each group has a number of files denoted by $(v_1, v_2, \ldots, v_g)$. The node calculates the average weight for each keyword in the vectors $\bar{w}_{t_i} = \sum_{j=1}^{g} w_{t_i}^{v_j} / g$, where $w_{t_i}^{v_j}$ denotes the w_{t_i} in v_j. We also pre-define a threshold for the average weight, denoted by $T_{\bar{w}}$. We use the keywords whose $\bar{w}_{t_i} > T_{\bar{w}}$ and their $\bar{w}_{t_i}$ to represent the interest group, represented by *group vector*:

$$v_G = (t_0, \bar{w}_{t_0}; t_1, \bar{w}_{t_1}; t_2, \bar{w}_{t_2}; ...; t_{n'}, \bar{w}_{t_{n'}}). \tag{4}$$

Thus, each node has a number of group vectors to represent its interests. The weight of each interest group $W(G)$ equals the portion of files that belong to it. We then generate a *node vector* (v_N) to describe a node's interest. The keywords of v_N is the keyword union of its group vectors and the weight of each keyword t is the sum of products of its weight $\bar{w}_t$ in each v_G it belongs to and the weight of the interest group $W(G)$. That is, the keyword list in v_N is $t(G_1) \cup t(G_2) \cup \cdots \cup t(G_{n'})$, where $t(G)$ means the keyword list in interest group G. The weight for term t in the *node vector* is $w_t = \sum_{i=1}^{n'} \bar{w}_t(G_i) * W_{G_i}$, where W_{G_i} is the weight of G_i and $\bar{w}_t(G_i)$ is the weight of keyword t in G_i. If keyword t does not belong to interest group v_{G_i}, $\bar{w}_t(G_i) = 0$.

B. Community Construction

Social network theory reveals that people with the same interest tend to gather and communicate with each other more frequently [22]. For example, people working in the ECE department of our university have more chances to encounter each other since they have similar interest in the area of ECE. By exploiting this social network property, SPOON classifies nodes with common interests and high contact frequencies into a community. Distant visitors with similar interests but few contacts are excluded from the community. Nodes having multiple interests belong to multiple communities. Common-interest nodes in a community share high similarity in individual's one or more interest groups.

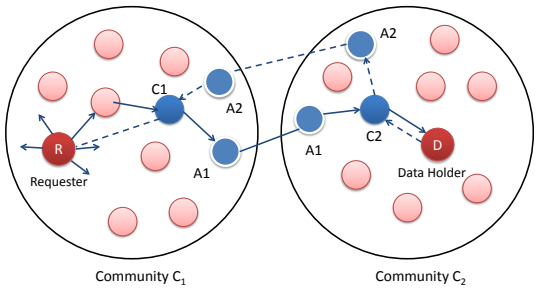


Fig. 2. File searching in SPOON.

When two nodes, N_1 and N_2, meet with each other, if they are not the members of any community, they calculate the similarity between each pair of their group vectors $sim(v_{G_i}(N_1), v_{G_j}(N_2))$ using Formula (3). A pair of group vectors, say $v_{G_i}(N_1)$ and $v_{G_j}(N_2)$, is called *matched interest group* when $W_{G_i} * W_{G_j} * sim(v_{G_i}(N_1), v_{G_j}(N_2)) > T_G$, where T_G is a predefined threshold. The purpose of taking into account the weight of each interest group is to eliminate the noise of small interest groups and achieve better interest clustering. If N_1 and N_2 have at least one pair of matched interest groups, and their contact frequency, $F(N_1, N_2)$, exceeds a predefined threshold T_e, the two nodes form a new community. The keywords in their matched interest groups and corresponding weights constitute the *community vector (v_C)* of the community. v_C represents the common interests of the community members.

If one of the two encountering nodes, say N_2, is already a member of community C, N_1 calculates $sim(v_{G_i}(N_1), v_C)$ to decide if it should join in the community C. If at least one similarity value is larger than T_G, the contact frequency of the node with the community nodes is calculated by: $\sum_{j \epsilon C} F(N_i, N_j)$. If it exceeds threshold T_e, N_1 is granted the membership to the community. This means that the frequency of node N_i contacting other members in community C is larger than a threshold. N_1 then copies the community vector and the community coordinator's information from N_2. In a community, when a member meets its community coordinator, it reports its new files generated after last report to the coordinator, which then records the files in its file index. Figure 2 shows an example of two communities.

Note that at the initial stage, each node only needs to keep track of contacts with nodes sharing similar interests, rather than all nodes. This relieves the burden on system during the community construction process. The values of the thresholds used in the interest extraction and the community construction process (e.g., T_s, T_G and T_e) should be determined by many factors such as number of nodes, number of interests and applications. We use empirical values in our experiments and leave the determination of these values to future work.

C. Node Role Assignment

A previous study has shown that in a social network consisting of mobile users, a small portion of nodes may have a large degree while most nodes have a small degree [23]. We can often find an important or popular figure who coordinates the communication between unfamiliar people or closely connects to members in a community in our daily life. For example, the college dean connects different departments in the college, and the department head connects different members in the department. Thus, we take advantage of different types of node mobility/ability in file sharing.

We define community coordinator and ambassador nodes in the view of a social network. A community coordinator is a relatively stable node with frequent connections to other members in its community. It keeps an index of all files in its community and maintains a stack of file queries that should be forwarded to other communities. It also maintains the v_C of foreign communities and corresponding ambassadors in order to map queries to ambassadors. A community has one ambassador for each foreign community. An ambassador for a foreign community is a highly mobile node that serves as the bridge between its community coordinator and members in the foreign community. It handles inter-community communication by receiving requests from its community coordinator and forwarding them to the foreign community.

In order to realize good system scalability and reliability, the number of ambassadors and coordinators can be adaptively adjusted based on the network size. In this paper, we use a single ambassador for each foreign community and a single coordinator in each community as an example.

1) Community Coordinator Node Selection: We define a stable node as the community coordinator which has high contact frequency with community members. It meets community members frequently and holds the most important role in the community. Thus, the criterion we use to select the communicator coordinator is the tightness of a node's relationship with other members.

We adopt the improved degree centrality [14], which adds weight on each link, for coordinator selection since it reflects the tightness of a node with other community members. When calculating the degree centrality in a MANET, a weight value is assigned for each edge linking two nodes based on the contact frequency. In the initial phase of coordinator discovery, each node, say node i, in a community collects contact information from its community neighbors and then calculates its degree centrality by:

$$D(p_i) = \sum_{j=1}^{N} w_{ij}, \tag{5}$$

where w_{ij} is the weight between node i and node j. In order to reflect the property that the coordinator has the most connections with all community members, w_{ij} equals 1 if the contact frequency between node i and node j is larger than a threshold and 0 otherwise. Then, nodes exchange the calculated degree centrality scores and select the community coordinator that has the highest value.

2) Community Ambassador Node Selection: We use egocentric centrality [26] for the ambassador selection since it reflects a node's ability to bridge nodes from two communities:

$$C(p_i) = \sum_{j=1}^{N} \sum_{k=1}^{M} \frac{w_{ji} * w_{ik} * g_{jk}(p_i)}{g_{jk}}, \quad (6)$$

where g_{jk} is the count of all geodesic paths linking nodes p_j and p_k, and $g_{jk}(p_i)$ is the number of those geodesic paths that involve node p_i. Node j is from the same community with node i, and node k is from another community. The weights w_{ji} and w_{ik} are the encountering frequencies between nodes j and i, and between nodes i and k, respectively.

A mobile node's egocentric network for the ambassador selection consists of nodes from both its home community and foreign communities. When nodes meet with each other, they exchange information needed for the centrality calculation. Each mobile node calculates its centrality score using Formula (6) and reports it to the coordinator in its home community, along with the community vectors of foreign communities that it frequently travels to. Then, the community coordinator chooses one ambassador for each foreign community (i.e., interest). If the ambassadors for some foreign communities do not exist, we select the node with the most inter-community connections by default.

An ambassador of a foreign community is responsible for the inter-community communication between the home community and the foreign community. They can carry requests targeting foreign communities out of current community and seek for potential forwarders outside. This arrangement facilitates interest-oriented file searching by enabling a coordinator to send a file request to the mapped foreign community quickly. We can also take into account other factors in ambassador selection, such as storage capacity and power.

D. Interest-oriented File Searching and Retrieval

In social networks, people usually have a few file interests [20] and their file visit pattern generally follows a certain distribution [21]. Also, people with the same interest tend to contact each other frequently [22]. Thus, interests can be a good guidance when routing requests to file holders.

Accordingly, the interest-oriented file searching scheme has two steps: intra-community and inter-community searching. A request first searches its home community. If that fails, inter-community searching is initiated to search files in foreign communities. During the search, a node sends a message to another node using the *interest-oriented routing algorithm (IRA)*, in which a message is always forwarded to the node which is likely to hold or to meet the queried keywords. The retrieved file is routed along the search path or through IRA if the route expires. This is because though the search path may expire, it provides a good guidance on the path back to the file requester. If a coordinator finds that its community cannot satisfy the received request, it launches the inter-community searching and looks up its ambassadors for forwarding opportunity. A request is deleted when its TTL (Time To Live) expires.

1) Interest-oriented Routing Algorithm: In SPOON, every node maintains a history vector that records its frequency of encountering interest keywords. The history vector is in the form of $v_H = (t_0, w_{h0}; t_1, w_{h1}; t_2, w_{h2}; ...; t_n, w_{hn})$, where w_{hi} is the aggregated times of encountering keyword t_i. w_{hi} decays periodically as time passes by $w_{hi} = \gamma w_{hi} (\gamma < 1)$. When two nodes meet, they exchange their node vectors and update history vectors. The history vector is used to evaluate the probability that the node meets the queried content.

The destination is represented by a vector (v_{dest}) represented by: $v_{dest} = (t_0, w_0; t_1, w_1; t_2, w_2; ...; t_n, w_n)$. In the interest-oriented routing algorithm (IRA), a node uses the fitness score $\mathcal{F}$ to evaluate its neighbors' probabilities to be or to meet the file holder. The fitness $\mathcal{F}$ of neighbor i is measured as $\mathcal{F} = \alpha sim(v_{dest}, v_{Ni}) + (1-\alpha) sim(v_{dest}, v_{Hi})$, where v_{Ni} and v_{Hi} are the node vector and history vector of node i, respectively. The factor of $sim(v_{dest}, v_{Ni})$ aims to find the node sharing the most similar interests with the destination, and the factor of $sim(v_{dest}, v_{Hi})$ aims to find a node that is very likely to meet the destination in its movement. α is used to control the weight of these two factors. In IRA, when a node receives a message, if its neighbor with the highest $\mathcal{F}$ has higher $\mathcal{F}$ than itself, it forwards the message to the neighbor. This process repeats until the message arrives at the destination. Note that coordinators do not use IRA but send messages to its community members when meeting them because a coordinator has tight connections with all community members.

2) Intra-Community File Searching and Retrieval: The query message is represented by a query vector (v_Q) represented as: $v_Q = (t_0, w_0; t_1, w_1; t_2, w_2; ...; t_n, w_n)$. Since the query is initiated by users, the weights of terms in v_Q are constant values. In the intra-community searching, the destination that a query is sent to is represented by a combination of the v_Q and the node vector of the requester's community coordinator (v_{N_C}), represented by:

$$v_{dest} = \lambda v_Q + (1-\lambda) v_{N_C}, \quad (7)$$

λ equals 1 when the remaining hop counter ($count$) is larger than 0 and 0 otherwise. This means that a requester first searches nearby nodes within $count$ hops, and then resorts to its community coordinator. The hop counter of a query is decreased by one after each forwarding. If the file is not found when $count = 0$, it is forwarded to the community coordinator ($v_{dest} = v_{N_C}$).

When node N_j receives a request, if $v_{dest} = v_Q$ and $sim(v_{dest}, v_{N_j})$ reaches the similarity threshold specified by the requester, it tries to send the satisfied files, if exist, back to the requester along the original path. If a forwarder is not available due to node mobility, IRA is used to forward the file. If $v_{dest} = v_Q$ but $sim(v_{dest}, v_{N_j})$ does not reach the specified similarity, N_j uses IRA to further forward the query. If $v_{dest} = v_{N_c}$ and N_j is not the coordinator N_c, N_j uses IRA to forward the request to N_c. After N_c receives the query, it checks its file index for the queried file in the community. If the index has files satisfying the request, the coordinator sends

the request to the file holder when meeting it, which sends the file back to the requester using IRA. Otherwise, N_c initiates the inter-community file searching.

3) Inter-Community File Searching and Retrieval: In the inter-community searching algorithm, a coordinator maps a request to foreign communities whose v_c has the highest $sim(v_Q, v_C)$. It then asks the ambassador to forward the request to the corresponding foreign community. The ambassador uses IRA to send the request to the coordinator in the foreign community.

Upon receiving the request, the coordinator in the foreign community checks its file index to see if its community has the file. If the file isn't held by any node, the coordinator repeats the inter-community file searching by looking up its ambassadors to check for further forwarding opportunities. If the file is found, the community coordinator asks for the file from the file holder when meeting it and sends the file back to the requester's community through the corresponding ambassador. The coordinator of the requester's community will further forward the file to the requester.

Figure 2 depicts the process of file searching, in which a requester (node R) in community C_1 generates a file request. Since its neighbors within $count$ hops don't have the file, the request is then forwarded to the community coordinator N_{C1}. N_{C1} checks the community file index but still can't find the file. It then asks the community ambassador N_{A1} to forward the request to the foreign community matching the queried file. Community coordinator N_{C2} finds the file and sends it back to the requester's community via ambassador N_{A2}. The file is first sent to N_{C1}, and then forwarded to the requester.

IV. Performance Evaluation

We first deployed the systems on the real-world GENI Orbit testbed [27], [28]. We used the real trace from the MIT Reality project [29], in which 94 smart phones were deployed among students and staff at MIT, to drive node encountering in the test. In GENI, nodes are equipped with wireless cards and communicate with each other through the wireless interface. A node can only communicate with others within its communication range through the wireless connection. We then conducted experiments on NS-2 [30] using the converted one-day trace data since the whole trace is too long for simulation. We also used a community based mobility model [31] to further evaluate the system in both connected and disconnected MANET environment in order to see the applicability of SPOON in different networks.

We evaluated the performance of SPOON in comparison with MOPS [18], PDI+DIS [8], [12] and Epidemic [32]. MOPS is a social network based content service system. It forms nodes with frequent contacts into a community, and selects the nodes with frequent contacts with other communities as brokers for inter-community communication. PDI provides distributed search service through local broadcasting (3 hop), and builds content tables in nodes along the response path. We complemented PDI with the advertisement-based DISsemination method [12], in which each node disseminates its contents to its neighbors, and call the combined method PDI+DIS. In Epidemic, when two nodes meet each other, they exchange messages that they haven't seen. We use the following metrics in the experiments:

(1) *Hit rate:* the percent of requests that are successfully delivered to the file holders. This metric reflects the capability of a method to discover the requested files.
(2) *Average delay:* the average delay time of the successfully delivered requests. This metric reflects the efficiency of a method to discover the requested files.
(3) *Maintenance cost:* the total number of all messages except the requests, which are for routing information establishment and update (i.e., node content exchange in all the four methods, request exchange in Epidemic and routing table establishment in PDI+DIS). This metric represents the cost for supporting file searching.
(4) *Total cost:* the total number of messages that have been generated during a test. This metric reflects the overall cost of a method in the discovery of requested files.

A. Performance in the GENI Experiment

1) GENI Experiment Parameters: We set the first 0.3 million seconds as the initialization period, in which each node builds and updates its neighbor table. In the following 1 million seconds, 1 node is randomly picked to generate a query every 100 seconds. Since the total length of the trace is about 2.56 million seconds, the TTL (Time to live) of each query was set to 1.2 million seconds. Also, considering that people usually generate queries according to their interests, we set 70% of total queries as intra-queries, which search for files located in the same community with the query originator. As the work of MOPS, we used 40% of the data set to detect communities and identified 7 communities in total. For node contents, we collected articles from 7 news categories (e.g., sports, entertainment and technology) from CNN.com and mapped them to 7 communities. Each node has 20 articles. We assume one node has one interest. For each node, we generated its group vector with around 40 keywords. A query aims for an article randomly selected from the collected article pool. To be practical, each node can store at most 2000 queries.

TABLE I
EFFICIENCY AND COST IN THE EXPERIMENTS ON GENI

Method	Hit Rate	Average Delay (s)	Maintenance Cost	Total Cost
SPOON	0.665	155356.9	251954	271981
MOPS	0.625	161070.0	311302	328266
PDI+DIS	0.508	7562.5	301918	361506
Epidemic	0.8745	15230.1	676685	867939

TABLE II
MEMORY USAGE IN THE EXPERIMENTS ON GENI

Metric	SPOON	MOPS	PDI+DIS	Epidemic
Ave. num. of queries in buffer	37.5	43.5	12.3	1998.6
Ave. size of a neighbor table	9.9	17.5	15.7	0

2) GENI Experiment Results: Table I shows the results of the GENI experiments of the four methods. From the table, we find that Epidemic generates the highest hit rate with the

highest total cost and a low average delay. This is resulted from the dissemination nature of Epidemic.

SPOON produces the second highest hit rate at the lowest total cost and relatively high average delay. This is because SPOON utilizes both contact and content properties of social network to guide the request in file searching and only keeps one copy of each request. Therefore, it can successfully locate queried files without the need of much information exchange and many request messages, though at a relatively slow speed. SPOON is superior over MOPS in terms of hit rate, delay and cost. This is because SPOON utilizes IRA for intra-communication and dedicated ambassadors for inter-communication while MOPS relies heavily on brokers. Also, MOPS only considers node contact in routing, while SPOON considers both content and contact. We will elaborate the explanation in describing the simulation results later on.

PDI+DIS generates the lowest hit rate at relatively high total cost and low average delay. The low hit rate is caused by the poor mobility resilient of the route table. As a result, only partial queries are resolved quickly in the local broadcasting while others passively wait for the file holders or updated routes and usually cannot be resolved in time. This means that most successful queries are resolved by the local broadcasting. Then, since we only count the average delay of successful queries, PDI+DIS has the lowest average delay.

We also evaluated the performance of the four methods in memory utilization in terms of the average number of queries in the buffer and the average size of the content table. The results are shown in Table II. For the average number of buffered queries, we find that PDI+DIS<SPOON<MOPS<Epidemic. Nodes in Epidemic buffer the most queries since Epidemic tries to replicate each request to all nodes in the system. Both SPOON and MOPS keep one copy of each request during the searching process. However, since SPOON completes file query more quickly than MOPS, as shown in Table I, it buffers fewer quires in memory than MOPS. For PDI+DIS, it stores the fewest number of queries in memory because of the local broadcasting, which just forward the query without buffering.

Considering each entry in the content table has roughly the same size as it records the content of one node, we used the number of entries in a table to represent its size. Except Epidemic, all other three methods need to build content tables for file searching. SPOON needs content synopses for both intra- and inter- community searching. MOPS needs that to guide query forwarding between brokers. PDI+DIS uses that to construct routing table and guide file searching. The results are illustrated in the second row of Table II, in which we find that Epidemic<SPOON<MOPS<PDI+DIS. SPOON stores the fewest content synopses because most nodes only store the information of the same community members. Though ambassadors store community vectors of corresponding communities and coordinators store node vectors of community members, the size of these vectors usually is limited and would not increase the memory usage significantly. In MOPS, brokers exchange content synopses of all nodes in their communities upon meeting with each other and usually consume a large amount of memory. Therefore, MOPS produces the largest average number of stored content synopses, though some nodes just store the content synopses of the same community members. PDI+DIS stores the most amount of content synopses because each node collects content synopses from all nodes it has met and all received reply messages. In summary, the results in Table I and Table II show that SPOON is superior over other three methods in terms of hit rate, average delay, total cost and memory-efficiency.

TABLE III
SIMULATION PARAMETER IN TESTS WITH REAL TRACE AND SYNTHESIZED NODE MOBILITY.

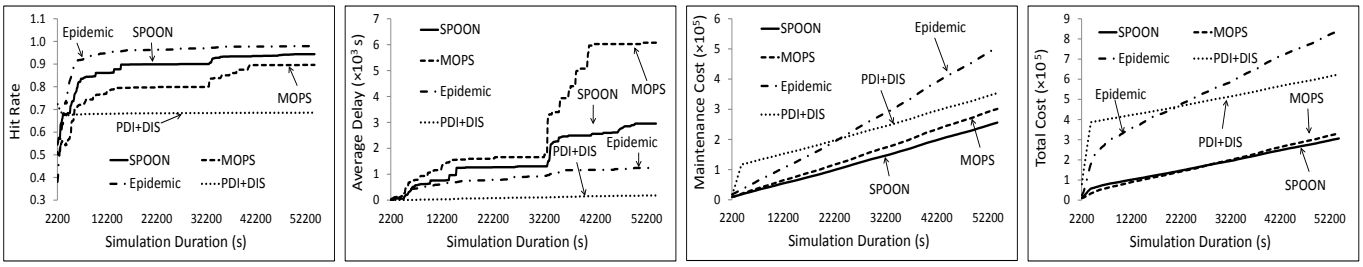
Node mobility	**Real trace**	**Synthesized**
Environment Parameters		
Simulation area	$2.5km \times 2.5km$	$4km \times 4km$
Simulation length	54000s	15000s
Community number	7	10
Node Parameters		
Node number	45	100
Communication range	250m	250m
Node speed	–	$1m/s - 6m/s$
Number of terms	40	40
Query Parameters		
Query rate	$8/s$	$15/s$
Intra-query percentage	50%	70%
Query period	2000s	900s

B. Performance in the Trace-driven Simulation

1) Experiment Settings and Parameters: We also conducted trace-driven simulation using one day records of connections with cellular towers in the MIT Reality Mining project [29] trace data in ns-2. Since the physical location of each tower was not provided due to privacy reasons, it poses a challenge to infer node mobility as required by the ns-2. We used a $2.5km \times 2.5km$ square which approximately covers the entire test area. We split the area into many small squares and placed towers in the center of the small squares in increasing order of their IDs. This setting of node movement is reasonable because it enables nodes, that connect to the same or nearby towers at approximately the same time, to have a high probability to be within the communication range of each other.

We adopted the same community structure and node content distribution as the experiment in GENI. We used 2000s for initialization and the next 2000s for querying activity. We purposely enlarge the TTL of each query to 50000s in order better observe the performance of different methods. We measured the experiment metric once every 100s for 5×10^4s. Table III shows a summary of default parameters in this test.

2) Hit Rate: Figure 3(a) shows the hit rates over time in different methods. We find that Epidemic can resolve almost all requests while PDI+DIS can only complete about 70% of requests. The hit rates of SPOON and MOPS finally reach about 95% and 90%, respectively. Epidemic has the highest hit rate because a node copies its stored requests to all met nodes, which will be eventually resolved. SPOON achieves higher hit rate than MOPS. In SPOON, coordinators who always reside in their home communities are responsible for contacting with ambassadors from its own or other communities. Also, the ambassador chosen for forwarding a query is the one with



(a) Hit rate. (b) Average delay. (c) Maintenance cost. (d) Total cost.

Fig. 3. Performance in the real trace driven experiments.

frequent communication with the destination community. In addition, a query is always actively forwarded to the node which has a high probability of meeting the destination by the interest-oriented routing. Thus, the request forwarder has high probability of meeting the coordinator in the destination community in file searching. MOPS only relies on the encountering of mobile brokers for file searching. This probability is lower than that of SPOON, resulting in lower hit rate.

In PDI+DIS, many routes in a content table expire quickly because of node mobility. As a result, most successful requests are those that request files located within 3 hops or reachable through routes. Without updated routes, requests for remote content (i.e., more than 3 hops away) have to passively stay in the buffer until meeting the file holders or next-hop nodes in the routes. Therefore, many requests cannot be resolved in the test, leading to a low hit rate.

We also notice that Epidemic, SPOON and MOPS exhibit a sharp rise in hit rate initially and then remain stable, while PDI+DIS remain nearly constant. In the former three methods, requests that can be resolved by nearby nodes contribute to the sudden increase of the hit rates. Requests that cannot be resolved immediately gradually arrive at file holders, causing slow increase of hit rates even after the querying activity stops. In PDI+DIS, after 3-hop broadcasting, buffered requests passively wait for file holders or routes to file holders, generating much fewer successful searches.

3) Average Delay: Figure 3(b) illustrates the average delay over time of the four methods. From the figure, we find that the delay follows PDI+DIS<Epidemic<SPOON<MOPS. Also, the average delay of SPOON and MOPS exhibit a sharp increase at the time around 32200s. Recall that we only measure the delay of successful requests. In PDI+DIS, most successful requests are resolved in the initial 3-hop broadcasting stage. Therefore, it generates the least average delay. In Epidemic, requests are rapidly distributed to more and more nodes at the cost of multiple copies. As a result, a request can reach its destination after a short waiting time.

MOPS exhibits large delay because requests in it usually have to wait for a long time for brokers or intra-community file holders. In contract, SPOON always tries to find an optimal neighbor to send a request to the file holder with the interest-oriented routing algorithm. In addition, the design of coordinator also increases the possibility of relaying requests to the destination community. From Figure 3(a), we see that starting at the time of 32200s, MOPS's hit rate gradually increases from 0.8 to 0.9 until 42200s, and SPOON's hit rate increases from 0.9 to 0.94 quickly. This phenomenon implies that after the staying in the buffer for 32200s in MOPS and SPOON, initially generated requests are gradually forwarded to the file holders, leading to an increase in successfully delivered requests and average delay.

4) Cost: Figure 3(c) plots the maintenance cost of different methods. The cost follows: Epidemic>PDI+DIS> MOPS>SPOON. Also, PDI+DIS's cost has a rapid increase at first, and then grows linearly at similar rate as SPOON.

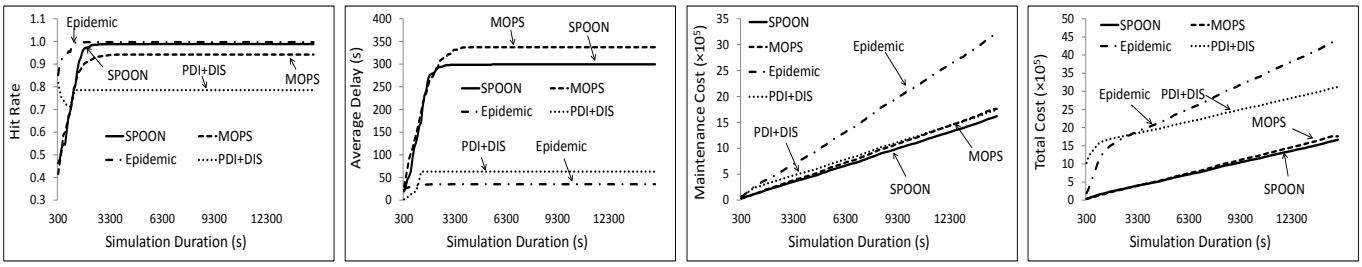
In all the four methods, each node exchanges its content with newly met node, which contributes to the maintenance cost. Other than this, the four methods need to exchange additional information. In SPOON, ambassadors report to coordinators the contents of other communities they collect. PDI+DIS needs to build content table, resulting a relatively high maintenance cost. In MOPS, brokers exchange the requests and contents of all nodes from their home communities when meeting each other. Therefore, MOPS produces slightly higher cost than SPOON. In Epidemic, two nodes also need to exchange their requests information. That's why the maintenance cost of Epidemic is much higher than that of SPOON.

We are very curious about the initial increase of PDI+DIS. From the stable hit rate of PDI+DIS in Figure 3(a), we know that most requests are resolved at the initial stage. Recall that PDI+DIS needs to build content tables for successfully resolved requests. Therefore, the maintenance cost increase fast at the beginning. Afterwards, few buffered requests are gradually resolved as shown in Figure 3(b), leading to a steady increase in maintenance cost due to content exchange.

Figure 3(d) shows the total cost of each method. We observe that by including the request cost, the relationship between Epidemic, MOPS and SPOON (Epidemic>MOPS>SPOON) remains the same as Figure 3(c), which means that the maintenance cost is the majority part of their total cost. Combined with the previous results, we conclude that SPOON is the most effective in terms of hit rate, searching delay and cost.

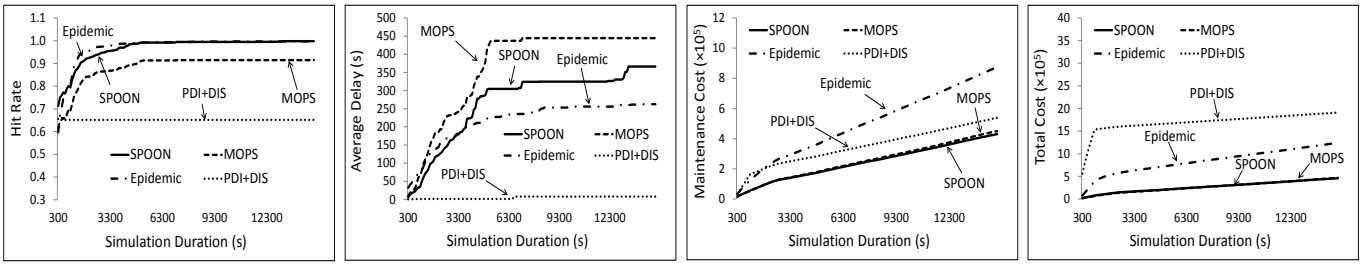
C. Performance in Disconnected and Connected MANETs

In this section, we tested SPOON and the three comparison methods in both connected and disconnected MANETs, which are synthesized by a community based mobility model [31].



(a) Hit rate. (b) Average delay. (c) Maintenance cost. (d) Total cost.

Fig. 4. Performance in a connected MANET.



(a) Hit rate. (b) Average delay. (c) Maintenance cost. (d) Total cost.

Fig. 5. Performance in a disconnected MANET.

1) Experiment Settings and Parameters: As the work in [23], we use the community based mobility model to create communities. This mobility model has been validated to be able to show similar properties with real traces [31]. We set all the parameters of the mobility model to the same as the work in [23]. The simulation area size was set to $1000m \times 1000m$ and $4000m \times 4000m$ for the connected and disconnected MANET, respectively. The entire simulation area is divided into 100 squares and each square holds one community. We set the number of nodes to 100, and created 10 communities with each representing one interest group (community). The movement speed of a normal node was randomly chosen from [1,6] m/s, a normal speed range for walking and bicycling. We selected 10% of all nodes as the travelers, which are evenly distributed among communities. The movement speed of travelers was set to the twice of the normal node's average speed, i.e., 7m/s. In the experiment, each person belongs mainly to one community, and contains the same files as in the trace-driven experiments. We set the TTL to 14000s. The query activity starts at 100s and lasts for 900s. The query rate was set to 15/s. Table III shows a summary of all default parameters in this test.

2) Hit Rate: Figure 4(a) and 5(a) show the hit rates over time of the four methods in the connected and disconnected MANETs, respectively. We observe that the results in both environments are similar as that of Figure 3(a) due to the same reasons. We also find that compared to in the disconnected MANET, PDI+DIS has much higher hit rate and others have slightly higher hit rates in the connected MANET. This is because in connected MANETs, nodes are close with each other. Then, each request in PDI+DIS can reach more nodes through broadcast, leading to a higher hit rate. However, since Epidemic, SPOON and MOPS temporarily buffered requests on nodes until they move close to a proper next hop, which alleviates the negative influence of node disconnection, their hit rates only increase marginally in the connected MANET.

3) Average Delay: Figure 4(b) and Figure 5(b) illustrate the average delay over time of the four methods in connected and disconnected MANETs, respectively. In both scenarios, SPOON generates 20% less delay than MOPS. We also observe that the results in Figure 5(b) for the disconnected MANET are nearly consistent with those in the trace-driven experiment in Figure 3(b) due to the same reasons.

It is very interesting to see that the results in Figure 4(b) show differences from those in Figure 3(b). Comparing Figure 4(b) and Figure 5(b), we find that PDI+DIS generates relatively higher delay in connected MANETs than in disconnected MANETs. Epidemic generates lower delay than PDI+DIS in connected MANETs, but higher delay in disconnected MANETs. This is because nodes are more likely to meet each other in the connected MANETs than in the disconnected MANETs, leading to more forwarding possibilities. As a result, PDI+DIS's hit rate significantly increases as shown in Figure 4(a) and average delay of all successful requests. In a connected MANET, Epidemic can quickly copy requests to all nodes in the system, resulting in a much lower delay. For SPOON and MOPS, each request has only one copy in the network and has to wait for a proper message forwarder. Therefore, the decreases in their delays are not large.

4) Cost: Figure 4(c) and Figure 5(c) plot the maintenance cost of different methods in connected and disconnected MANETs, respectively. The results in the two figures approximately match the results in the trace-driven experiment in Figure 3(c) due to the same reasons. We observe that Epidemic still has the highest maintenance cost and PDI+DIS>MOPS>SPOON. Comparing the absolute values in

Figure 4(c) and Figure 5(c), we find that the cost generated in the connected MANET is much higher than that in the disconnected MANET. This is because nodes have higher probability to meet each other in the connected MANET, thereby increasing the number of exchanged messages.

Figure 4(d) and Figure 5(d) show the total cost of the four methods in connected and disconnected MANETs, respectively. Both results match that of the trace-driven experiment in Figure 3(d) due to the same reasons. In Epidemic and PDI+DIS, nodes disseminate requests to all met nodes in the network. MOPS and SPOON only forward one request, generating much lower request cost than Epidemic and PDI+DIS.

These results, along with what we obtained in the GENI experiment and trace-driven simulation, confirm the high efficiency of SPOON in file searching in both connected and disconnected MANETs by leveraging social networks. By considering both node interest and contact frequency social network properties, SPOON reduces delay and overhead and increases hit rate of MOPS, which only considers the contact frequency. Though Epidemic and PDI+DIS show small average delays, Epidemic generates a high cost while PDI+DIS is not mobility resilient.

V. Conclusion

In this paper, we propose a Social network based P2P cOntent file sharing system in mObile ad-hoc Networks (SPOON). SPOON considers both node interest and contact frequency for highly efficient file sharing. We introduce four SPOON components: *interest extraction*, *community construction*, *node role assignment*, and *interest-oriented file searching and retrieval*. *Interest extraction* identifies nodes' interests. *Community construction* builds common-interest nodes with frequent contacts into communities. Although node disconnection occurs frequently due to node mobility, the *node role assignment* component exploits the node mobility for efficient file searching. A node with frequent contact with community members is elected as the community coordinator, which helps intra-community file search. Highly mobile nodes that connect external communities are chosen as the community ambassadors, which help inter-community file search. The *interest-oriented file searching scheme* first searches a requester's home community, and then search a foreign community of the queried file through ambassadors. Both the system deployment on real world Orbit platform and the trace-driven experiment in simulated disconnected and connected MANETs show that SPOON can dramatically reduce the traffic cost of flooding-based and advertisement-based methods, and can reduce the delay in another social contact-based method. Also, SPOON achieves a much higher success rate in file sharing than other methods. In our future work, we will explore how to determine appropriate thresholds in SPOON and how the thresholds affect the file sharing efficiency.

Acknowledgements

This research was supported in part by U.S. NSF grants OCI-1064230, CNS-1049947, CNS-1025652, CNS-1025649, CNS-1057530 and CNS-0917056, Microsoft Research Faculty Fellowship 8300751, and Sandia National Laboratories grant 10002282.

References

[1] "Next Generation Smartphones Players, Opportunities & Forecasts 2008-2013," Juniper Research, Tech. Rep., 2009.
[2] "A Market Overview And Introduction to GyPSii," http://corporate.gypsii.com/docs/MarketOverview.
[3] "Bittorrent," http://www.bittorrent.com/.
[4] KaZaA, http://www.kazaa.com.
[5] Http://www.bloglines.com/ref/p2p-statistics.html.
[6] M. Papadopouli and H. Schulzrinne, "A Performance Analysis of 7DS: a Peer-to-Peer Data Dissemination and Prefetching Tool for Mobile Users," *Advances in wired and wireless communications, IEEE Sarnoff Symposium Digest*, 2001.
[7] A. Klemm, C. Lindemann, and O. Waldhorst, "A Special-Purpose Peer-to-Peer File Sharing System for Mobile Ad Hoc Networks," in *Proc. of VTC*, 2003.
[8] C. Lindemann and O. P. Waldhort, "A Distributed Search Service for Peer-to-Peer File Sharing," in *Proc. of P2P*, 2002.
[9] D. W. A. Hayes, "Peer-to-Peer Information Sharing in a Mobile Ad hoc Environment," in *Proc. of WMCSA*, 2004.
[10] J. B. Tchakarov and N. H. Vaidya, "Efficient Content Location in Wireless Ad Hoc Networks," in *Proc. of MDM*, 2004.
[11] C. Hoh and R. Hwang, "P2P File Sharing System over MANET based on Swarm Intelligence: A Cross-Layer Design," in *Proc of WCNC*, 2007, pp. 2674–2679.
[12] T. Repantis and V. Kalogeraki, "Data Dissemination in Mobile Peer-to-Peer Networks," in *Proc. of MDM*, 2005.
[13] C. Boldrini, C. Conti, M. Jacopini, and A. Jacopo Passarella, "HiBOp: a History Based Routing Protocol for Opportunistic Networks," in *Proc. of WoWMoM*, 2007, pp. 1–12.
[14] E. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant manets," in *Proc. of MobiHoc*, 2007.
[15] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble Rap: Social-based Forwarding in Delay Tolerant Networks," in *Proc. of MobiHoc*, 2008.
[16] P. Hui, E. Yoneki, S. Y. Chan, and J. Crowcroft, "Distributed Community Detection in Delay Tolerant Networks," in *Proc. of MobiArch*, 2007.
[17] P. Hui and J. Crowcroft, "How Small Labels Create Big Improvements," in *Proc. of PerComW*, 2007.
[18] F. Li and J. Wu, "MOPS: Providing Content-Based Service in Disruption-Tolerant Networks," in *Proc. of ICDCS*, 2009.
[19] C. Boldrini, M. Conti, and A. Passarella, "Contentplace: Social-aware data dissemination in opportunistic networks," in *Proc. of MSWIM*, 2008.
[20] A. Fast, D. Jensen, and B. N. Levine, "Creating social networks to improve peer-to-peer networking," in *Proc. of KDD*, 2005.
[21] A. Iamnitchi, M. Ripeanu, and I. T. Foster, "Small-world file-sharing communities," in *Proc. of INFOCOM*, 2004.
[22] M. Mcpherson, "Birds of a feather: Homophily in social networks," *Annual Review of Sociology*, vol. 27, no. 1, pp. 415–444, 2001.
[23] P. Costa, C. Mascolo, M. Musolesi, and G. P. Picco, "Socially-aware Routing for Publish-Subscribe in Delay-Tolerant Mobile Ad Hoc Networks," *IEEE JSAC*, vol. 26, no. 5, pp. 748–760, 2008.
[24] H. Schtze and C. Silverstein, "Projections for Efficient Document Clustering," in *Proc. of SIGIR*, 1997, pp. 74–81.
[25] P. Bonacich, "Factoring and Weighting Approaches to Status Scores and Clique Identification," *J. of Math.Sociol.*, 1972.
[26] P. Marsden, "Egocentric and sociocentric measures of network centrality," *Social Networks*, vol. 24, no. 4, pp. 407–422, 2002.
[27] "GENI project," http://www.geni.net/.
[28] "Orbit," http://www.orbit-lab.org/.
[29] N. Eagle, A. Pentland, and D. Lazer, "Inferring Social Network Structure using Mobile Phone Data," *PNAS*, vol. 106, no. 36, 2009.
[30] "The network simulator ns-2," http://www.isi.edu/nsnam/ns/.
[31] M. Musolesi and C. Mascolo, "Designing Mobility Models Based on Social Network Theory," *ACM SIGMOBILE Comput. and Comm. Rev.*, 2007.
[32] A. Vahdat and D. Becker, "Epidemic routing for partially-connected ad hoc networks," Duke University, Tech. Rep., 2000.