Cont^2: Social-Aware Content and Contact Based File Search in Delay Tolerant Networks

Kang Chen and Haiying Shen
Department of Electrical and Computer Engineering
Clemson University, Clemson, SC 29631
Email: {kangc, shenh}@clemson.edu

***Abstract*—In this paper, we focus on distributed file search over a delay tolerant network (DTN) formed by mobile devices that exhibit the characteristics of social networks. Current file search methods in MANETs/DTNs are either content-based or contact-based. The former builds routing tables for node contents but is not resilient to high node mobility, while the latter exploits node contact patterns in the social networks but may lead to high latency. Recent research also reveal the importance of interests in realizing efficient file dissemination in DTNs. In this paper, we first analyze node interest and mobility from real traces, which confirms the shortcomings of a contact based method and show the importance of considering both content/interest and contact in file search. We then propose Cont^2, a social-aware file search method which leverages both node social interests (content) and contact patterns to enhance search efficiency. First, considering people with common interests tend to share files and gather together, Cont^2 virtually groups common-interest nodes into a community to direct file search. Second, considering human mobility follows a certain pattern, Cont^2 exploits nodes that have high contact frequency with the queried content. Third, Cont^2 also exploits active nodes that have more connections to others as a complementary approach to expedite file search. Trace-driven experimental on the real-world GENI testbed and NS-2 simulator show that Cont^2 can significantly improve the search efficiency compared to current methods.**

***Keywords*-Social-Aware, File Search, Delay Tolerant Networks**

I. Introduction

In this paper, we focus on distributed peer-to-peer file search in a delay tolerant network (DTN) formed by mobile devices, the holders of which exhibit certain social network properties. Imagine students could easily acquire course materials through mobile phones while walking on campus and drivers could acquire weather and traffic conditions while driving. This research is driven by the shortcomings of the infrastructure-based wireless networks: 1) service is not available everywhere (e.g., in rural area), and 2) its centralized model imposes high capacity stress on the central server, which may lead to bottlenecks. Moreover, such a file sharing model enables nodes (people) to exploit common interest and movement pattern in a decentralized manner. This may provide potential new social applications/services that integrate both interests and locations.

File searching has been studies thoroughly in mobile ad hoc networks (MANETs), which are similar to DTNs on the distributed nature. One simple distributed file search method in MANETs is broadcasting [1]–[3], in which queries are flooded throughout all or a portion of nodes to search for a requested file. Though this method can find files quickly, it is burdened by huge amounts of traffic, leading to low efficiency. In another group of content-based file search methods [4]–[7], nodes disseminate their content synopses to certain other nodes, which build routing tables (i.e., content tables) containing routes to the content holders for query forwarding. However, node mobility can make routing tables expire quickly, thereby degrading the search reliability. SPOON [8] leverages social networks for content search in MANETs where common-interest nodes always stay together. However, it cannot be used in DTNs where common-interest nodes, though meet often compared to other nodes, do not necessarily always stay together.

Recently, some methods [9], [10] have been proposed to leverage node contacts for content dissemination or replication in DTNs. They group nodes with frequent contact to realize efficient file service. For example, MOPS [9] creates communities in which nodes have frequent contacts for intra-community file sharing and selects nodes with frequent contacts with other communities as brokers for inter-community content distribution. However, only considering node contacts in DTNs may lead to low efficiency. First, brokers in different communities are sometimes disconnected to each other. Second, nodes in the same community do not always stay together. Third, a node usually has multiple interests, and few nodes share many interests. This implies that contact based communities may hold files from different interests, leading to frequent inter-community search. Our study on a real trace obtained from students and staff on a campus [11] also confirms these drawbacks.

To overcome these shortcomings, we propose a social-aware Content and Contact based file search method, namely Cont^2, for DTNs in a social network environment. The cornerstone for the design of Cont^2 comes from two properties of social networks in our DTN scenario:
(P1) Common interest: every node has social interests and nodes with a common interest, though may be seperated, tend to meet more often with each other than with other nodes [12];
(P2) Movement pattern: people usually visit the same location periodically with skewed preferences [13].

By leveraging P1, we develop a social community creation algorithm that virtually classifies nodes into different communities based on their interests. This helps to forward a file query toward the destination community. By leveraging P2, we develop a file searching algorithm that always forwards requests to nodes that have high contact frequency with the queried content (i.e., they have high possibility to meet at the locations they visit periodically). If such a node does not exist, the algorithm then utilizes active nodes to expedite the search. Cont^2 is particularly proposed for the social network environment where file interests indicate node mobility and file requests. Such a condition can be found in many environments for the purpose of file sharing such as campus, community and company, though it cannot be guaranteed in all DTN scenarios.

II. Related Work

A lot of works have been proposed to achieve distributed file or content search in MANETs. One group of methods is based on broadcasting [1]–[3]. 7DS [1] exploits the mobility of nodes within a geographic area to disseminate content through neighbors. In Epidemic [3], two nodes exchange messages they haven't seen upon their encountering. Hayes *et al.* [2] extended the Gnutella peer-to-peer data sharing system [14] to a mobile environment, in which a node broadcasts its initiated or received queries to its connected nodes. Another group of file search methods is based on file content. PDI [4] and ORION [5] create a content table on each node along the reply path of broadcasted queries, which directs future searches. Repantis *et al.* [6] proposed to let each node disseminate its content synopses to its neighbors one or several hops away to guide query messages. GCLP [7] sends out node content and queries in vertical directions to ensure the success of file searches. Though these methods can find files in MANETs, they suffer from a huge amount of broadcasted messages and easily expired routing table caused by node mobility, respectively, leading to degraded system efficiency.

There are some methods for efficient data dissemination/publish in DTNs [9], [10], [15]–[20]. MOPS [9] groups nodes with frequent contact into a community and selects nodes having frequent contact with a neighboring community as brokers for inter-community communication. The socio-aware overlay [10] builds brokers into an overlay, in which brokers use unicast or direct communication protocols (e.g., WiFi access points) for communication. In [15], besides node mobility, the author also considers users' impatience in acquiring files in deciding the optimal file caching in opportunistic network, However, these methods only utilize node contacts to guide the content publish and fail to further consider node interests to enhance file availability.

Researchers also have proposed efficient content dissemination algorithms in DTNs based on node interests [17]–[20]. Lenders *et al.* [17] investigated realizing podcasting in MANETs and discussed different content solicitation strategies. Zhang *et al.* [18] defined friends as nodes with similar interests and evaluated four data diffusion strategies (i.e, diffusing similar or different data between friends or strangers). They concluded that diffusing similar data between friends and different data between strangers lead to the best performance. In ContentPlace [19], nodes collect files that are possibly interested by nodes in their social communities. In addition to node interests, Gao *et al.* [20] further considered social contact patterns in data dissemination by forwarding data to nodes that are more likely to meet nodes interested in it. These methods mainly investigate how to disseminate contents to nodes, which is different from the goal of file search in Cont^2. Pitkäen *et al.* [16] studied how to stop the content searching at a suitable step in single-copy routing [21], Spray-and-Wait [22] and epidemic routing so that the number of discovered files and searching overhead can be balanced in DTNs. Unlike this work, our work focuses on a content search method to quickly and efficiently find requested contents.

SPOON in our previous work [8] shares the same goal of efficient file search as Cont^2. However, SPOON was developed for MANETs with strong node interest and contact correlation, i.e., nodes with similar interests often meet together, while Cont^2 is developed for DTNs where nodes with similar interests do not necessarily always stay together. Due to this difference, SPOON and Cont^2 have different file search systems. SPOON focuses on identifying common-interest nodes based on their stored files and forming them into a cluster to facilitate file search, while Cont^2 focuses on file search among dispersed common-interest node community members. Cont^2 novelly builds *community contact tables* and *neighbor table* on each node to direct the file searching.

III. Real-Trace Analysis

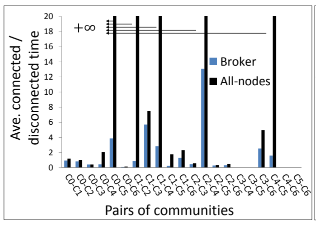
We analyzed the MIT Reality trace [11], which records the encountering of 94 smart phones held by students and staff at MIT, to verify the drawbacks of only considering node contacts in file search in DTNs. Using the method in MOPS, we classified nodes in the trace into 7 communities (C_{1-7}), in each of which nodes share frequent contacts. We selected one pair of brokers for each community pair and analyzed the data for one day (15 hours=54000s). A broker in community C_1 for community C_2 refers to the node in C_1 that has the highest accumulated contact frequency with nodes in C_2.

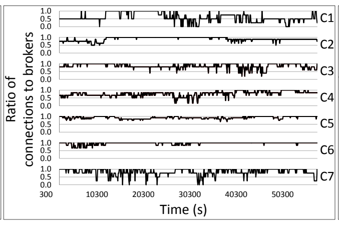
A. Trace Analysis and Observation

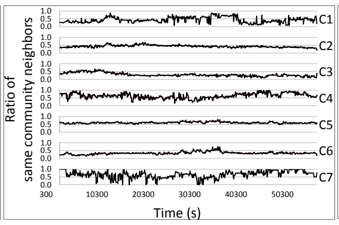
1) Can brokers always connect to communities? We measured the average ratio of connected/disconnected time for each community pair when all nodes, denoted *All-nodes*, and only brokers, denoted Broker, are capable of establishing connections between two communities, respectively. The results are given in Figure 1(a). In Broker, the ratio is very small (less than 0.5) for many community pairs. This means these community pairs cannot assure stable connections due to broker disconnections. *All-nodes* configuration produces much higher ratios than Broker. Its ratios of five community pairs are even infinite (i.e., the two communities are always connected). From the results, we made an observation (O) below:

O1: *Only relying on brokers for inter-community communication may lead to community disconnections and reduces the chances of interactions between communities.*

2) Can nodes in one community always connect to brokers? A community's *ratio of connections to brokers* is the portion of community members that connect to at least one broker. Figure 1(b) plots this metric for each


(a) Ratio of connected/disconnected time.


(b) Ratio of connections to brokers.


(c) Ratio of same community neighbors.

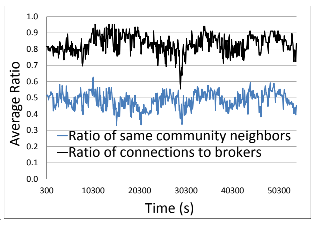

(d) Overall average ratio.

Figure 1. Real-trace analytical results.

community every 100s and shows that the ratio varies in range [0.5,1]. Figure 1(d) plots the average value of all communities, which shows that the average value is around 0.8. This means that at every moment, averagely 20% of nodes cannot communicate with their brokers, and in some communities, 50% of nodes cannot communicate with their brokers.

O2: *Due to node mobility, community nodes do not always connect to their brokers.*

From O1 and O2, we derived the following inference (I).

I1: *Exploiting all nodes to forward requests in the direction of destination in both intra- and inter-community file searching can enhance search efficiency and reliability.*

3) Can nodes in one community always stay together? A node's *ratio of same community neighbors* is the portion of a node's neighbors that belong to the node's community. Figure 1(c) plots the average of the ratios of all nodes in each community every 100s, and shows that the ratio varies greatly in [0.1, 0.9]. Figure 1(d) plots the average value of all communities and shows that the average value is around 0.5, which means that on average half of a node's neighbors are not from the same community.

O3: *Due to node mobility, nodes in one community do not always stay together.*

The above observations obtained from the trace on a campus indicate that MOPS is not suitable for such a campus social network environment, and a new file search system is needed for this application scenario with the observed features.

4) Do nodes with a common interest also share many other interests? We crawled the data of 117 users from Facebook [23] and examined their interest closeness. These people watched a video of a randomly selected user in Facebook. We identified 20 interests (e.g., sports, gaming and pop music) and found 86 users who have at least one of these interests. We found 1462 pairs of users that share at least one common interest. We then calculated the interest closeness between user i and j (c_{ij}) by: $c_{ij} = m_{ij}^2/s_i s_j$, where m_{ij} is the number of shared interests of the two users, s_i and s_j are the number of interests of user i and user j, respectively. Figure 2 shows the interest closeness of different pairs of users. We observe that the interest

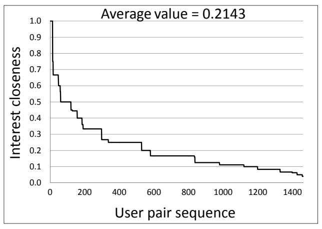


Figure 2. Interest closeness of common-interest user pairs.

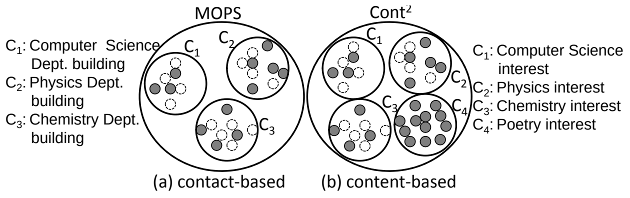

MOPS
Cont²
C1: Computer Science Dept. building
C2: Physics Dept. building
C3: Chemistry Dept. building
C1: Computer Science interest
C2: Physics interest
C3: Chemistry interest
C4: Poetry interest
(a) contact-based
(b) content-based

Figure 3. Community creation of MOPS vs. Cont^2.

closeness mainly varies in the range [0.1, 0.33] and few exceed 0.4, and the average closeness is only 0.2143. This result proves that though each pair of nodes share at least one common interest, they rarely share most interests. Although this result is obtained from Facebook, it reflects users' interests and matches our daily experience that people usually do not share many common interests [12].

O4: *In a social network, nodes usually have multiple interests. Two nodes sharing one interest usually do not necessarily share more other interests.*

From O3 and O4, we infer:

I2: *Forming frequently contacted nodes into a community may not be able to limit most file searches within a community due to nodes' movements and diverse interests.*

B. Why Combined Content and Contact Design

Figure 3 shows an example of the communities constructed with a contact-based method (MOPS) and a content-based method (Cont^2), respectively. In MOPS, each community consists of nodes in a department since they meet frequently. In Cont^2, nodes are grouped according to their interests (contents). In the figure, some students (gray nodes) in different departments also attend a poetry class and gather together regularly during the poetry class.

Obviously, MOPS is efficient if most queries can be satisfied within its current community for its tight intra-community connections. However, this does not hold in practice (**I2**), leading to frequent inter-community searchers. Also, since communities in MOPS do not represent content information, a forwarder needs to know the content indexes of all other communities, which is costly and inefficient. For example, in Figure 3(a), a requester in community C_1 queries a file in community C_3. The broker of C_1 (b_1) has to know the file index of C_3 first. Otherwise, the file request either waits on b_1 or gets forwarded blindly. Therefore, purely relying on node contacts for file search may lead to low searching efficiency.

On the other side, nodes in interest (content) based communities are not as tightly connected as in the contact based communities. Therefore, a query reaching the destination community cannot meet the file holder easily.

For example, in Figure 3(b), when a poetry related request arrives at community C_4, it's file holder may stay in other communities at the moment, leading to a long waiting time. In this case, social properties about node contacts (**P2**) should be utilized to forward the request to the file holder greedily, thereby overcoming the relatively loose structure in interest (content) based communities and enhancing file search efficiency.

In summary, both contents and contacts are necessary for efficient file search in DTNs.

IV. The Design of Cont^2

Cont^2 has three components: community creation, neighbor table construction and update, and content and contact based file search. We elaborate them in below.

A. Community Creation

Without loss of generality, we assume that the files stored in a node can be reflected by the node's interests. We then define a *community* as a group of nodes sharing the same interest to gear towards the purpose of file searching. Thus, in Cont^2, common-interest nodes virtually form into a community. By virtually, we mean that nodes in one community do not have to always stay together. However, based on previous discussion, same interest nodes still tend to have higher encountering possibility with each other than with others, which connects our community definition with node mobility for the purpose of file sharing. A node with multiple interests belongs to multiple communities and carries the IDs of these communities.

Cont^2 has a server that functions as a bootstrap for node (or interest) joining. The server maintains an interest list containing all interests and associated keywords in the system. This list is initially configured by the system administrator and is updated when necessary. For example, in a movie file sharing system, the interests include action, comedy, and romance. The server can map a set of keywords to an interest. Such automatic mapping can be conducted through machine learning techniques, which are beyond the discussion of this paper. A node needs to register to the server when joining in the system. The node derives its keywords from its files through a keyword abstraction technique [24] and report that to the server during the registration process. Based on the keywords, the server identifies the node's interests and community IDs. If there is no community matching the interest, a new community ID is created for the node. When a node changes its interests, it reports its updated interests to the server, which then assigns new community IDs to the node. A node can manually connect to the server through the Internet or 3G network for the registration, and then switches to the P2P mode for file sharing. This does not incur much extra cost or limitation since each node is only required to report interests to the server.

Common-interest communities benefit efficient file search from two aspects: (1) it increases the probability that a node finds its interested files in its own community since common-interest nodes tend to meet more frequently, as introduced in Section I, and (2) it helps to forward queries toward the destination community without detouring since it can learn the destination community in advance.

B. Neighbor Table Construction and Update

Each node in Cont^2 maintains a *neighbor table* (Table I) that helps to select the next hop node in file searching. It records the information of a node's current neighbors and previous same-community neighbors. The information includes node ID, community ID, content synopses, *community contact table* (CCT) and a connection bit (CB). The content synopses of a neighbor shows its contents, which are generated with the keywords of all its files. Each keyword k is associated with a weight w_k, which is the portion of files that contain the keyword on the node. Thus, a content synopses is represented by $<k_1, w_{k_1}; k_2, w_{k_2}; \cdots>$. The content synopses is updated when two nodes meet. CB is a boolean value indicating whether the node is currently connected to the neighbor.

Table I
Neighbor Table

Node ID	Community ID	Content synopses	CCT	CB
1	0x0001, 0x0010	$< k_1, w_{k_1}; k_2, w_{k_2}; \cdots >$	CCT$_1$	1
2	0x0008	$< k_1, w_{k_1}; k_2, w_{k_2}; \cdots >$	CCT$_2$	0
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

Table II
Community Contact Table (CCT)

Community ID	1-hop contact frequency	2-hop contact frequency
0x0001	0.7	2.9
0x0010	1.1	1.9
0x0011	0	0.5
0x0100	2.3	4.2
$\cdots$	$\cdots$	$\cdots$

In Table I, CCT$_i$ denotes the CCT (Table II) of neighbor i, which records its n-hop ($n = 1, 2, 3, \cdots$) contact frequency with each community. A node's n-hop contact frequency with a community represents its probability of connecting to the community through n hops. A node's probability of connecting to a community equals to its accumulated probabilities of meeting the members in the community. We use this definition since CCT serves to forward a file query to its matched community. Specifically, node i's 1-hop contact frequency with community C represents i's direct contact probability with C; node i's n-hop ($n > 1$) contact frequency with community C refers to the accumulated $n-1$ hop contact frequency of node i's current and past neighbors with C. CCT helps inter-community search by selecting nodes with the highest probability of meeting the destination community as the next relay node. Although more information in the CCT would give better direction on request forwarding, we confine n to 2 in order to balance the routing performance and storage and transmission cost.

We use F_{inC_x} ($n \geq 1$) to denote node i's n-hop contact frequency with community C_x, which is initialized to 0 at the beginning. Node i periodically updates its F_{inC_x} with each community after each unit time period T. Specifically, when node i encounters a new neighbor j, they exchange their neighbor tables for subsequent periodical table updates. Suppose the accumulated time period that nodes i and j connect with each other is t_{ij} during T and node j belongs to community C_1. Then, node i's CCT is updated by:

$$\begin{cases} F_{inC_1} = F_{inC_1} + \frac{t_{ij}}{T};\ \text{if}\ \ n = 1 \\ F_{inC_x} += \frac{t_{ij}}{T} * F_{j(n-1)C_x};\ \text{if}\ \ n > 1\ \&\ x \neq 1 \end{cases} \quad (1)$$

Node i increases F_{i1C_1} by its contact frequency with node j during period T since it belongs to community C_1. F_{inC_x} $(n > 1)$ refers to node i's n-hop contact frequency with each community C_x. Thus, node j's (n-1)-hop contact frequency with C_x is added to F_{inC_x} because a message from node i needs one more hop to reach C_x through node j. In the end of each T, node i updates its contact frequency to each community by

$$F^{new}_{inC_x} = \beta F^{old}_{inC_x} + (1-\beta)F_{inC_x}\ (n \geq 1), \quad (2)$$

where $\beta < 1$ is a fading weight, $F^{old}_{inC_x}$ and $F^{new}_{inC_x}$ are node i's old and new contact frequency with community C_x, respectively. The value of β is determined by the weight of previous and most recent meeting frequency on deciding F_{inC_x}. Note the system administrator should decide a suitable T based on the active level of nodes in the system.

The *community contact table* reflects a node's contact frequencies with different communities and guides the file searching algorithm, as explained later. Therefore, it should be updated properly so that it can reflect both long term meeting ability and short term changes. β is designed for this purpose by controlling how fast the overall contact frequency evolve along with the contact frequency measured in current time unit. β should match the actual frequency change rate, which is a complex problem. We leave how to decide it to future research. In this paper, considering meeting frequencies in daily social network usually present both long term stability and short term changes, we set β to a medium value of 0.5.

C. Content and Contact Based File Searching

The searching algorithm is developed based on the social network property described in Section I. Since each node knows the interest list in the system, a file requester can map its request to an interest (destination community). When a node receives a request, if it is the file holder, it returns the file. Otherwise, if the node is located in the destination community, it conducts intra-community searching. If it is not in the destination community, it conducts inter-community searching, which forwards the request gradually to the destination community. When the request arrives at the destination community, the intra-community search is launched. Note that the definition of community ensures that the requested file, if exists in the system, is highly possible to be held by nodes in the matched community (i.e., nodes with files in the interest are classified into the corresponding community). Then, it is not needed to query encountered nodes for the requested file before arriving at the destination community, thereby saving energy for file searching, which is a desired property in the energy-constrained mobile scenario. Each request has a Time To Live (TTL), after which it is dropped.

1) Intra-Community Searching: In this step, requests are forwarded within the destination community to find the file holder. In each forwarding, the request is forwarded to a neighbor node that has more intra-community connections toward the node having the highest similarity with the queried file. Such a design comes from two reasons. First, the node having high similarity with a query has high probability of containing the requested file. Second, an interest can usually be further classified into sub-interests, and people in a sub-interest group have a higher probability of meeting with each other than with other members in the interest community. For example, lab members majoring in computer systems tend to meet more often. Then, even the high similarity node fails to satisfy the request, it's frequently met nodes may contain the requested file. Therefore, the similarity works as an indication of the probability of satisfying the request.

We denote the node with the highest similarity with the queried file as the *temporary destination node (TDN)*. The similarity is calculated as following:

$$Sim(f,i) = \sum_{k \in K} w_k, \quad (3)$$

where f is the queried file, K denotes the keyword group in the request, and w_k denotes the weight of keyword k in node i's content synopses. w_k=0 if the synopses does not contain k. The similarity here shows the percentage of files on the node that matches the keywords in the query and indicates the possibility that the requested file is on the node. Therefore, the TDN should be the node with $\max\{Sim(f,i)\}$ among all nodes that have been visited by the request. That is

$$Sim(f, TDN) = \max\{Sim(f,i)\ (i \in AN(R_f)\}, \quad (4)$$

where R_f represents a request and $AN(R_f)$ represents all nodes that R_f has already visited. When a request arrives at a new node, the $AN(R_f)$ and TDN are updated accordingly.

Specifically, suppose node N_a receives the file request, if it holds the requested file, it returns the file to the requester and the file search is successful. Otherwise, it first checks whether the file holder or the TDN currently is connected. If yes, that node is the next relay node. If not, N_a then chooses the node (i.e., active node) that is lightly loaded and has the highest F_{i1C} with its current community as the next relay node by referring to the CCT in its neighbor table. We consider node load status in routing because the active node may become overloaded. A node measures its overload status by the occupation of its buffer, and piggybacks it on the "hello" messages used in neighbor scanning process.

2) Inter-Community Searching: In inter-community searching, node N_a first checks its neighbor table to see whether there is a current neighbor belonging to the destination community (C_d), and takes it as the next relay node if one exists. If more than one exists, N_a chooses the one that has the highest F_{i1C_d} by referring to the CCT in its neighbor table since it has more intra-community connections. If no node from the destination community can be found, N_a chooses a node from itself and its neighbors that has the highest F_{i1C_d} as the next relay node.

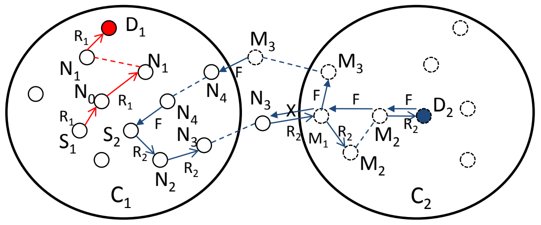


Figure 4. File searching and retrieval process.

If multiple nodes have the same highest F_{i1C_d}, the node with the highest F_{i2C_d} is chosen. In this way, the request can be quickly forwarded to the destination community, because F_{inC} reflects a node's probability of meeting nodes in community C through n hops.

It is possible that all nodes in current community have very few contacts with the destination community. To deal with this problem, we pre-define a threshold for F_{i1C} and F_{i2C}, denoted by T_d. If no node in the neighbor table has $F_{inC} > T_d$, the active node with the highest 1-hop contact frequency with the current community is chosen. The purpose of this strategy is to quickly move the request out of current area. If the node itself is chosen as the next relay node after these steps, it holds the request. While it is moving, it updates its neighbors and repeats the above steps until the request is forwarded to the destination community.

3) File Retrieval: Upon receiving a request, the destination node first tries to send the file back along the route the request traversed, which is inserted into the request during the forwarding process. If the reverse route is broken, the intra-community and inter-community searching algorithms can be used to send the file back to the requester according to the IDs of the requester and its community.

In current design, we only consider the scenario that there is only one matched file for each query. This can be extended to multiple matched queries by allowing each query to continue searching after a successful hit.

Figure 4 shows an example of file searching between two communities (C_1 and C_2). Two requests (R_1 and R_2) are initiated in C_1. R_1's destination community is C_1. Thus, the requester uses intra-community searching. It first finds the temporary target N_0 in its neighbor table and forwards R_1 to N_0. N_0 updates the temporary target with D_1, which has the closer similarity with the requested file. Because D_1 is not N_0's current neighbor, N_0 forwards R_1 to its neighbor N_1, which has the highest 1-hop contact frequency with C_1. Since D_1 is not N_1's current neighbor, N_1 holds the request, and delivers R_1 to D_1 when moving close to it. D_1 notices that its synopses match the request exactly, so it replies the requested file to the requester.

The file requested by R_2 belongs to community C_2, since the requester cannot find a current neighbor that belongs to C_2, it forwards R_2 to its current neighbor N_2 which has the highest 1-hop contact frequency with C_2. N_2 forwards R_2 to N_3. On the way moving to C_2, N_3 forwards R_2 to M_1 of C_2 and has a higher 1-hop contact frequency with C_2. Then, intra-community searching is used, in which M_1 forwards R_2 to destination D_2 through M_2.

We use a line with a solid arrow to stand for the file retrieval process. Node D_2 first sends the request for the requested file (F) back to the requester S_2 along the query route. However, when M_1 receives the request, it finds that node N_2 on the reverse route is not available. Node M_1 then launches a search for requester S_2. First, inter-community searching is used to route F to node N_4 in community C_1 through node M_3 in community C_2. Second, N_4 starts intra-community searching. N_4 finds itself has the highest mobility among all neighbors, so it carries the request until it moves close to S_2.

V. Performance Evaluation

We first deployed the systems on the real-world GENI Orbit testbed [25], [26]. We then conducted experiments on NS-2 [27] using the converted one-day trace data since the whole trace is too long for simulation. We also used a community based mobility model [28] to further evaluate the systems with different network sizes and node mobility. Detailed introduction and configuration of the community based model are given later in Section V-A4. Considering the storage and energy constraint of mobile devices, we expect the system size to be at most several hundreds of nodes.

A. Experiment Settings

In order to better evaluate the file search, we determined the community construction and file generation in advance. We extracted interest groups and corresponding keywords from the profiles we crawled from 117 Facebook users who watched the same randomly selected video. We selected the top seven mostly shared interests and mapped them to the 7 communities identified from the real trace. For each of a node's interests, we randomly selected 40 keywords from the keyword pool of the interest and generated about 20 files. A queried file of an interest is randomly picked from the file pool of the interest, and each query matches only one file in the system. Consider people would like to query file in its interests, 70% of requested files have the same interest of the originator. We run each test for 5 times and present the average of the results within the 95% confidence interval of all the test results.

1) Comparison Methods: We used below comparison methods in the experiments:

(1) MOPS [9]: MOPS leverages social networks to provide content-based service (e.g., publish/subscribe system) in disruption-tolerant networks by grouping nodes that have frequent contacts into a community. Each community uses nodes that have frequent contact with other communities as brokers for inter-community communication. We use 2 brokers for each community to balance the cost and search efficiency. MOPS buffers requests on the current node if no better routes can be discovered.

(2) PDI+DIS [6], [4]: PDI [4] uses 3-hop local broadcast and content tables for file searching. A content table contains content and corresponding routes to the content owners and is built in nodes along the response path. We complement PDI with the advertisement-based DISsemination method [6], in which each node disseminates its content to its neighbors. In PDI+DIS, a request is first broadcasted for 3 hops. During and after the 3-hop broadcasting, if a node finds a route to the queried content in its content table, it uses the route for routing. A node

buffers a request if it cannot forward the request due to expired routes or lack of routes.

(3) Epidemic [3]: Epidemic is a buffering based broadcasting mechanism. It tries to disseminate requests to all nodes in the system by letting two nodes exchange requests they haven't seen upon their encountering.

2) Test Metrics: Since the routing process of file retrieval is similar as the file searching process, we only test the performance of file searching. We used the number of generated messages to represent costs. We define that one message only contains the information of one node.

(1) *Hit rate:* the percentage of requests that are successfully delivered to the file holders. It represents the reliability of a file search method to discover queried files.

(2) *Average delay:* the average delay time of all successful requests. The delay of a request is the time elapsed after the request is initiated and before it arrives at the file holder. It represents the efficiency of a file search method.

(3) *Maintenance cost:* the total number of all messages except the requests. These messages 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 generated during the simulation including request messages. This metric represents the total cost of file searching.

3) GENI Experiment Parameters: To demonstrate the effect of Cont^2 in scenario with real communication interfaces, we first tested it on the GENI Orbit testbed, which consists of 400 nodes that can connect with each other through wireless connections. We use it to simulate DTNs and adopt the MIT Reality project trace to drive node mobility, which lasts for about 2.56 million seconds. We set the first 0.3 million seconds as the initialization period so that nodes can collect enough records to reflect their meeting probabilities with others. After that, we randomly picked one node to generate one query every 100 seconds for 1 million seconds. The watching period (or TTL) of each query was set to 1.2 million seconds. To be practical, each node can hold at most 2000 queries in its buffer. When the buffer is full, the oldest packet is dropped.

Table III
PARAMETERS IN SIMULATION

	Real trace	Synthesized
Environment Parameters	**Value**	**Default Value**
Simulation area	$2.5km \times 2.5km$	$2.5km \times 2.5km$
Number of caves	-	25
Cave size	-	$500m \times 500m$
Number of communities	7	7
Node Parameters	**Value**	**Default Value**
Number of nodes	45	100
Communication range	250m	250m
Average node speed	–	$1.5m/s$
Re-wiring probability	–	0.1
Number of travelers/cave	–	2
Traveler speed	–	2*(average speed)
Number of keywords	40	40
Querying Parameters	**Value**	**Default Value**
Querying rate	$8/s$	$8/s$
Intra-query percentage	70%	70%
Initialization period	1000s	1000s
Querying period	2000s	2000s
Waiting period	51000s	51000s

4) Simulation Parameters: According to the settings in the GENI experiment and the works in [29]–[31], we determined the parameters shown in Table III for the simulation on NS-2. We recorded the experimental metrics every 100s after the initialization period. In the community based mobility model [28], the test area is divided into many caves, each of which represents one community area. Nodes move within its home community randomly in most of the time. We map each interest community defined in Cont^2 to a randomly selected cave in the mobility model [28]. The model also allows setting travelers that frequently commute between two communities with high speed. Considering that travelers are more active, we set a normal node's speed to a value randomly selected from the range $[2v/3, 4v/3]$ (v denotes the average speed) and a traveler's speed to $2v$. In the tests with different node mobility, we varied the average speed (v) from the medium walking speed of human beings ($1.5m/s$) to the medium vehicle speed ($15m/s$).

B. Performance in GENI Experiment

Table IV shows the GENI experiment results. We see that Epidemic generates the highest hit rate but also the highest cost since it tries to replicate each request to all nodes. PDI+DIS generates the lowest hit rate because the routes in content tables often expire due to node mobility and many requests wait passively for the file holders. Therefore most successful queries in PDI+DIS are resolved locally within 3 hops, leading to a low average delay. Cont^2 achieves the second highest hit rate but significantly lower cost than Epidemic. Cont^2 also generates higher hit rate than PDI+DIS, which shows Cont^2's higher mobility-resilience. Comparing Cont^2 and MOPS, we see that Cont^2 is superior over MOPS in terms of hit rate, delay and cost. This is because Cont^2 utilizes all possible forwarding opportunities around a node while MOPS relies heavily on brokers. Also, MOPS only considers node contact, while Cont^2 considers both content and contact.

We also evaluated the memory usage of the four methods, measured by the average number of queries in the buffer and the average size (i.e., number of entries) of the neighbor table. Table V shows the average values of all nodes. We find that PDI+DIS has the smallest average number of queries in the buffer. This is because most queries are quickly resolved in the 3-hop local broadcasting without further buffering. In Cont^2 and MOPS, queries are buffered until meeting better forwarding nodes. Cont^2 generates fewer queries in the buffer than MOPS since it can deliver request more quickly as shown in Table IV. Epidemic produces the largest result as it tries to distribute each query to all nodes.

Each node in all methods except the Epidemic also stores the content synopses of nodes it has met. Cont^2 and MOPS need content synopses for both intra- and inter-community searches. PDI+DIS uses it to build content tables. From Table V, we find that Cont^2 stores fewer content synopses than the other two methods because it only stores the information of same community nodes and currently connected neighbors. For MOPS, brokers store that of all communities they have known, and normal nodes store that of the nodes in their own communities. Therefore, MOPS has the largest average number of stored content synopses. In PDI+DIS, a node disseminates its contents to its neighbors. Thus, each node stores the content synopses of all nodes it has met, leading to higher memory consumption than Cont^2.

In summary, Table IV and Table V show that Cont^2 has superior performance over other methods considering overall efficiency, delay, cost and memory-efficiency.

Table IV
EFFICIENCY AND COST IN THE EXPERIMENTS ON GENI

Method	Hit Rate	Average Delay (s)	Maintenance Cost	Total Cost
Cont^2	0.696	142892.8	231918	269917
MOPS	0.625	161070.0	311302	328266
PDI+DIS	0.508	7562.5	301918	361506
Epidemic	0.8745	15230.1	676685	867939

Table V
MEMORY USAGE IN THE EXPERIMENTS ON GENI

Metric	Cont^2	MOPS	PDI+DIS	Epidemic
Ave. # of queries in buffer	33.5	43.5	12.3	1998.6
Ave. size of a neighbor table	7.9	17.5	15.7	0

C. Performance in Trace-drive Simulations

1) Hit Rate: Figure 5(a) shows the hit rates of different methods over time. We see that the hit rates of Epidemic and Cont^2 reach 99%, that of MOPS is about 95% and that of PDI+DIS is below 90%. Epidemic tries to disseminate each request to all nodes in the system, thereby resulting in a high hit rate. In Cont^2, request forwarders fully utilize nearby nodes to forward the request in the direction of the destination, leading to efficient file search.

MOPS only relies on brokers for inter-community search and direct encountering of community members for intra-community search. As previously introduced, due to the diversity of node interests, inter-community search is always needed in MOPS. However, without the guidance of content, MOPS has to rely on the information exchange between brokers for inter-community search, which may miss some forwarding opportunities. Moreover, the passive waiting in the intra-community search also takes a long time. Consequently, MOPS cannot resolve some queries before they expire, resulting in a lower hit rate. PDI+DIS only completes about 74% of requests, and its hit rate remains nearly constant throughout the test. This is because many requests cannot be resolved in the test since the routes in a content table expire quickly due to node mobility. After the local broadcast, requests just wait passively, leading to almost no increase in the hit rate.

We also find that the hit rates of Epidemic, Cont^2, and MOPS exhibit sharp rises at the initial stage and increase slightly afterwards, while PDI+DIS remains nearly constant throughout the test. In Epidemic, Cont^2, and MOPS, requests that cannot be immediately resolved stay in current nodes and gradually arrive at file holders using query forwarding algorithms. Therefore, the hit rate gradually increases as time goes on. In PDI+DIS, after 3-hop broadcasting, buffered requests passively wait for routes to file holders, generating much fewer successful searches. Therefore, its hit rate remains almost stable.

2) Average Delay: Figure 5(b) shows that the average delays of the four methods follow MOPS>Cont^2 >Epidemic>PDI+DIS. Recall that we only measure the delay of successful requests. So PDI+DIS has the least average delay since most successful requests are resolved in the initial 3-hop broadcasting stage. Cont^2 reduces the delay of MOPS by half for the same reasons as in Figure 5(a). This result confirms that only considering contacts for file search cannot provide high efficiency. Epidemic results in relatively lower average delay than Cont^2 due to its broadcasting nature. It is reasonable that Cont^2 generates higher delay than Epidemic since Cont^2 only maintains one copy for each request.

It is interesting to observe that the delay of MOPS increases rapidly at around 40000s while that of Cont^2 increases steadily. In MOPS, a broker may need to buffer inter-community requests for a long period before being able to contact brokers in a neighboring community. Then, the encountering of two brokers with many unresolved requests may lead to a rapid increase in hit rate and average delay, as occurred at 40000s. Without relying on fixed brokers, Cont^2 utilizes every forwarding opportunity, leading to a steady increase in hit rate and average delay. This result confirms the drawback of depending only on brokers and the advantage of utilizing all available nodes in request forwarding.

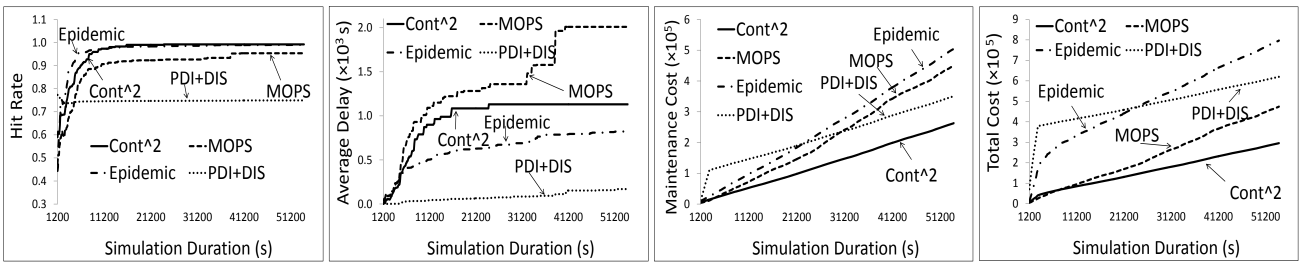
3) Maintenance Cost: Figure 5(c) plots the maintenance costs of different methods over time. After 31200s, the costs follow Epidemic>MOPS>PDI+DIS>Cont^2. In all methods, node content is exchanged among encountered nodes, which contributes to the linear growth of maintenance cost over time. Except Cont^2, nodes in the other three methods need to exchange other information in addition to their own contents. PDI+DIS builds routes in nodes along the response paths of successful queries. Thus, PDI+DIS generates higher maintenance cost than Cont^2. In MOPS, brokers exchange contents of all nodes from their home communities, leading to an even higher maintenance cost. In Epidemic, two nodes exchange information about all known requests to decide unseen requests, resulting in the highest maintenance cost.

4) Total Cost: Figure 5(d) shows the total cost of each method over time. We observe that Epidemic>PDI+DIS >MOPS>Cont^2. Epidemic has very high cost since it tries to disseminate a request to all nodes in the system. Each message has only one copy in MOPS and Cont^2. PDI+DIS has a local broadcast, which generates many query copies, leading to higher total cost than MOPS and Cont^2.

D. Performance With Different Network Sizes in Simulation

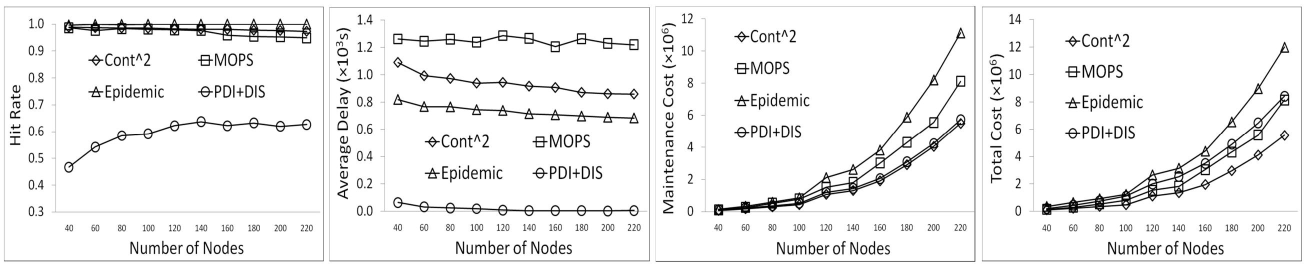
In this test, we examined the performance of the methods when the total number of nodes varied from 40 to 220.

1) Hit Rate: Figure 6(a) plots the hit rates of the four methods. We find that the hit rates of Epidemic, Cont^2 and MOPS reach over 95% (MOPS<Cont^2<Epidemic) while PDI+DIS resolves only 60% of requests. Epidemic achieves high hit rate due to its system-wide message dissemination. The reasons for MOPS<Cont^2 and the low hit rate of PDI+DIS are the same as explained in Figure 5(a). Also, the hit rates of Cont^2, MOPS, and Epidemic remain relatively stable while that of PDI+DIS increases as the network size increases. The former three methods actively



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

Figure 5. Performance in trace-driven simulation.



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

Figure 6. Performance with different network sizes in simulation.

forward messages to file holders by broadcasting or routing algorithms, which ultimately forwards most requests to their destinations, even in a sparse network. In contrast, unsolved requests after the 3-hop broadcasting stage in PDI+DIS passively wait for routes to the file holders. Hence, higher node density enables more forwarding opportunities, thus increasing the hit rate of PDI+DIS.

2) Average Delay: Figure 6(b) shows the average delay of each method. The result matches what was obtained in Figure 5(b), i.e., MOPS$>$Cont$^2>$Epidemic$>$PDI+DIS and Cont^2 reduces the delay of MOPS by about 20% on average. We also find the delay of MOPS remains relatively stable while those of Epidemic and Cont^2 exhibit a slight decrease as network size increases. This is because higher node density means more forwarding opportunities, thereby reducing the delay. MOPS only relies on brokers for inter-community communication. Though the number of nodes increases, the probability that two brokers meet does not change. Therefore, the average delay of MOPS remains stable though the network size increases.

3) Maintenance Cost: Figure 6(c) illustrates the maintenance cost of each method. It can be observed that the maintenance costs follow Epidemic$>$MOPS$>$PDI+DIS$>$Cont^2. Generally, Epidemic produces much higher maintenance cost than Cont^2, the maintenance cost of MOPS is approximately 40% higher than that of Cont^2, and PDI+DIS generates a 5% higher maintenance cost than Cont^2 in dense networks. The reasons for this result remain the same as in Figure 5(c). Also, the costs of all the methods grow quickly as network size increases. This is because with more nodes in the network, the amount of exchanged messages increases.

4) Total Cost: Figure 6(d) demonstrates that Epidemic generates the highest total cost. PDI+DIS and MOPS show moderate total costs while Cont^2 has the lowest total cost. Epidemic has the highest total cost since it generates a high maintenance cost (Figure 6(c)) and a large amount of requests in the test. The number of request messages in PDI+DIS increases quickly as the number of nodes increases due to the local broadcasting, resulting in a high total cost. MOPS incurs approximately the same number of request messages as Cont^2, but renders higher maintenance cost (Figure 6(c)), leading to higher total cost than Cont^2. The results in Figure 6 show the superior performance of Cont^2 and its efficiency in networks with different sizes.

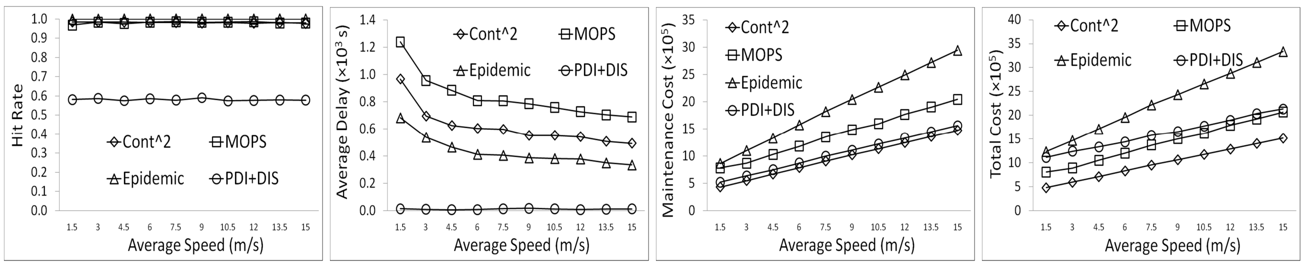
E. Performance With Different Node Mobility in Simulation

We also evaluated the performance of the methods when the average movement speed varied from walking speed ($1.5m/s$) to medium vehicle speed ($15m/s$).

1) Hit Rate: Figure 7(a) shows the hit rate of each method. Epidemic, Cont^2, and MOPS have hit rates close to 100% at all speeds while PDI+DIS exhibits a low hit rate. The results show the same relative performance of the four methods as in Figure 5(a) and Figure 6(a) with the same reasons.

2) Average Delay: In Figure 7(b), the average delays of the methods follow MOPS$>$Cont$^2>$Epidemic with PDI+DIS having almost no delay at all node movement speeds. This matches the results in Figure 5(b) and Figure 6(b) due to the same reasons. Moreover, we find that Cont^2 has a 20% lower delay than MOPS at all movement speeds, which confirms the high efficiency of Cont^2 in file searching. Also, we see that the delays of MOPS, Cont^2 and Epidemic decrease as node movement speed increases. This is because fast node movement increases the frequency of node encountering and reduces the waiting time of requests in the buffers, hence shortening the average delay.

3) Maintenance Cost: Figure 7(c) shows that the maintenance costs of the four methods follow Epidemic$>$MOPS $>$PDI+DIS$>$Cont^2 for the same reasons as in Figure 5(c) and Figure 6(c). We also find that the maintenance costs



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

Figure 7. Performance with different node mobility in simulation.

of Epidemic, PDI+DIS, MOPS and Cont^2 increase linearly when nodes move faster. This is because with faster movement, nodes meet frequently and exchange more contents, leading to a higher maintenance cost.

4) Total Cost: Figure 7(d) shows that the total costs of the four methods follow Epidemic>PDI+DIS >MOPS>Cont^2. The reasons are the same as in Figure 5(d) and Figure 6(d). The total costs of the four methods increase as the average speed increases because nodes generate higher maintenance costs with faster movement (Figure 7(c)). The total cost of Cont^2 still remains lower than other methods, which verifies Cont^2's low cost in different mobility rates.

VI. Conclusion

This paper presents a content and contact based file search method for DTNs in a social network environment, namely Cont^2. It exploits the properties of social networks to enhance file searching efficiency. Through the study of a real trace, we found that the interests (content) of each node can help guide file searching and that the movement patterns of mobile nodes based on interests can more accurately predict the encountering of nodes holding the interested files. Node mobility can also be utilized to further enhance searching efficiency. Thus, Cont^2 virtually builds common-interest nodes into a community and forwards a file request to nodes with higher meeting frequency with the interest community or the node that has the most similar content with the requested file. We compared Cont^2 with other file search methods using mobility from both real-trace and a community based mobility model on the real-world GENI testbed and NS-2 simulator. Cont^2 shows superior performance in hit rate, delay and overall cost. In the future, we plan to investigate how the influence of a node's interest weights on its movement patterns and how to leverage it to enhance file search efficiency.

Acknowledgements

This research was supported in part by U.S. NSF grants CNS-1254006, CNS-1249603, CNS-1049947, CNS-0917056 and CNS-1025652, Microsoft Research Faculty Fellowship 8300751.

References

[1] S. Srinivasan, A. Moghadam, S. G. Hong, and H. Schulzrinne, "7ds - node cooperation and information exchange in mostly disconnected networks," in *Proc. of ICC*, 2007, pp. 3921–3927.
[2] D. W. A. Hayes, "Peer-to-peer information sharing in a mobile ad hoc environment," in *Proc. of WMCSA*, 2004.
[3] A. Vahdat and D. Becker, "Epidemic routing for partially-connected ad hoc networks," Duke University, Tech. Rep., 2000.
[4] C. Lindemann and O. Waldhorst, "A distributed search service for peer-to-peer file sharing in mobile applications," in *Proc. of P2P*, 2002.
[5] A. Klemm, E. Klemm, C. Lindemann, and O. P. Waldhorst, "A special-purpose peer-to-peer file sharing system for mobile ad hoc networks," in *Proc. of VTC*, 2003, pp. 2758–2763.
[6] T. Repantis and V. Kalogeraki, "Data dissemination in mobile peer-to-peer networks," in *Proc. of MDM*, 2005, pp. 211–219.
[7] J. B. Tchakarov and N. H. Vaidya, "Efficient content location in wireless ad hoc networks," in *Proc. of MDM*, 2004.
[8] K. Chen, H. Shen, and H. Zhang, "Leveraging social networks for p2p content-based file sharing in mobile ad hoc networks." in *Proc. of MASS*, 2011.
[9] F. Li and J. Wu, "Mops: Providing content-based service in disruption-tolerant networks," in *Proc. of ICDCS*, 2009.
[10] E. Yoneki, P. Hui, S. Chan, and J. Crowcroft, "A socio-aware overlay for publish/subscribe communication in delay tolerant networks," in *Proc. of MSWiM*, 2007, pp. 225–234.
[11] N. Eagle, A. Pentland, and D. Lazer, "Inferring social network structure using mobile phone data," *PNAS*, vol. 106, no. 36, 2009.
[12] M. Mcpherson, "Birds of a feather: Homophily in social networks," *Annual Review of Sociology*, vol. 27, no. 1, pp. 415–444, 2001.
[13] W.-J. Hsu, T. Spyropoulos, K. Psounis, and A. Helmy, "Modeling time-variant user mobility in wireless mobile networks," in *Proc. of INFOCOM*, 2007, pp. 758–766.
[14] "Gnutella," http://en.wikipedia.org/wiki/Gnutella.
[15] J. Reich and A. Chaintreau, "The age of impatience: optimal replication schemes for opportunistic networks," in *Proc. of CoNEXT*, 2009.
[16] M. Pitkänen, T. Kärkkäinen, J. Greifenberg, and J. Ott, "Searching for content in mobile DTNs." in *Proc. of PerCom*, 2009.
[17] V. Lenders, M. May, G. Karlsson, and C. Wacha, "Wireless ad hoc podcasting," *SIGMOBILE Mob. Comput. Commun. Rev.*, 2008.
[18] Y. Zhang, W. Gao, G. Cao, T. L. Porta, B. Krishnamachari, and A. Iyengar, "Social-aware data diffusion in delay tolerant," 2012.
[19] C. Boldrini, M. Conti, and A. Passarella, "Design and performance evaluation of contentplace, a social-aware data dissemination system for opportunistic networks." *Computer Networks*, 2010.
[20] W. Gao and G. Cao, "User-centric data dissemination in disruption tolerant networks." in *Proc. of INFOCOM*, 2011.
[21] A. Balasubramanian, B. N. Levine, and A. Venkataramani, "DTN routing as a resource allocation problem." in *Proc. of SIGCOMM*, 2007.
[22] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: An efficient routing scheme for intermittently connected mobile networks," in *Proc. of WDTN*, 2005.
[23] "Facebook," http://www.facebook.com/.
[24] A. Y. Halevy, "Piazza: Data management infrastructure for semantic web applications," in *Proc. of WWW*, 2003, pp. 556–567.
[25] "GENI project," http://www.geni.net/.
[26] "Orbit," http://www.orbit-lab.org/.
[27] "The Network Simulator ns-2," http://www.isi.edu/nsnam/ns/.
[28] M. Musolesi and C. Mascolo, "Designing mobility models based on social network theory." *Mobi. Comp. and Comm. Rev.*, 2007.
[29] 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.
[30] E. M. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant manets," in *Proc. of MobiHoc*, 2007.
[31] A. Lindgren, A. Doria, and O. Scheln, "Probabilistic routing in intermittently connected networks." *Mobi. Comp. and Comm. Rev.*, 2003.