This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2014.2308211, IEEE Transactions on Computers

Maximizing P2P File Access Availability in Mobile Ad hoc Networks Though Replication for Efficient File Sharing

Kang Chen, *Student Member, IEEE,* Haiying Shen*, *Senior Member, IEEE,*

Abstract—File sharing applications in mobile ad hoc networks (MANETs) have attracted more and more attention in recent years. The efficiency of file querying suffers from the distinctive properties of such networks including node mobility and limited communication range and resource. An intuitive method to alleviate this problem is to create file replicas in the network. However, despite the efforts on file replication, no research has focused on the global optimal replica creation with minimum average querying delay. Specifically, current file replication protocols in mobile ad hoc networks have two shortcomings. First, they lack a rule to allocate limited resource to different files in order to minimize the average querying delay. Second, they simply consider storage as resource for replicas, but neglect the fact that the file holders' frequency of meeting other nodes also plays an important role in determining file availability. Actually, a node that has a higher meeting frequency with others provides higher availability to its files. This becomes even more evident in sparsely distributed MANETs, where nodes meet disruptively. In this paper, we introduce a new concept of resource for file replication, which considers both node storage and meeting frequency. We theoretically study the influence of resource allocation on the average querying delay and derive a resource allocation rule to minimize the average querying delay. We further propose a distributed file replication protocol to realize the proposed rule. Extensive trace-driven experiments with synthesized traces and real traces show that our protocol can achieve shorter average querying delay at a lower cost than current replication protocols.

Index Terms—MANETs, Peer-to-Peer, File Sharing, File Availability

1 Introduction

With the popularity of popularity of mobile devices, i.e., smartphones and laptops, we envision the future of MANETs consisted of these mobile devices. By MANETs, we refer to both normal MANETs and disconnected MANETs (or delay tolerant networks (DTNs). The former has a relatively dense node distribution in a local area while the latter has sparsely distributed nodes that opportunistically meet each other. On the other side, the emerging of mobile file sharing applications (e.g., Qik [1] and Flixwagon [2]) also motivates the investigation on the peer-to-peer (P2P) file sharing over such MANETs.

The local P2P model provides three advantages. Firstly, it enables file sharing when no base stations are available (e.g., rural area). Secondly, with the P2P architecture, the bottleneck on overloaded servers in current client-server based file sharing systems can be avoided. Thirdly, it exploits the otherwise wasted peer to peer communication opportunities among mobile nodes. As a result, nodes can freely and unobtrusively access and share files in the distributed MANET environment, which can possibly support some interesting applications. For example, mobile nodes can share files based on users' proximity [3] in the same building or a local community. Tourists can share their travel experiences or emergency information with other tourists through digital devices directly even when no base station is available in remote areas. Drivers can share road or weather information through the vehicle-to-vehicle communication.

However, the distinctive properties of MANETs, including node mobility, limited communication range and resource, have rendered many difficulties in realizing such a P2P file sharing system. For example, file searching turns out to be non-trivial and time consuming since nodes in MANETs move around freely and can exchange information only when they are within the communication range. Broadcasting can quickly discover files, but it generates the broadcast storm problem [4] with high energy consumption. Probabilistic routing and file discovery protocols [5]–[7] avoid broadcasting by forwarding a query to a node with higher probability of meeting the destination. But the opportunistic encountering of nodes in MANETs makes file searching and retrieval non-trivial.

File replication is an effective way to enhance file availability and reduce file querying delay. It creates replicas for a file to improve its probability of being encountered by requests. Unfortunately, it is impractical and inefficient to enable every node to hold the replicas of all files in the system considering limited node resources. Also, file querying delay is always a main concern in a file sharing system. Users often desire to receive their requested files quickly no matter whether the files are popular or unpopular. Thus, a critical issue is raised for further investigation: *how to allocate the limited resource in the network to different files for replication so that the overall average file querying delay is minimized?*

- * *Corresponding Author. Email: shenh@clemson.edu; Phone: (864) 656 5931; Fax: (864) 656 5910.*
- *The authors are with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC, 29634. E-mail: {kangc, shenh}@clemson.edu*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2014.2308211, IEEE Transactions on Computers

Recently, a number of file replication protocols have been proposed for MANETs [8]–[12]. In these protocols, each individual node replicates files it frequently queries [8]–[10], or a group of nodes create one replica for each file they frequently query [10]–[12]. In the former, redundant replicas are easily created in the system, wasting resources. In the latter, though redundant replicas are reduced by group cooperation, neighboring nodes may separate from each other due to node mobility, leading to large query delay. There are also some works addressing content caching in more sparsely distributed MANETs (disconnected MANETs/DTNs) for efficient data retrieval [13]–[19] or message routing [20]. They basically follow an intuitive way to cache data that are frequently queried on places that are visited frequently by mobile nodes. Both the two categories of replication methods fail to thoroughly consider that a node's mobility affects the availability of its files.

In spite of the efforts, current file replication protocols lack a rule to allocate limited resource to different files for replica creation in order to achieve the minimum global average querying delay, i.e., global search efficiency optimization under limited resource. Moreover, they simply consider storage as the resource for replicas, but neglect that a node's frequency to meet other nodes (meeting ability in short) also influences the availability of its files. Files in a node with a higher meeting ability have higher availability.

In this paper, we introduce a new concept of resource for file replication, which considers both node storage and node meeting ability. We theoretically study the influence of resource allocation on the average querying delay and derive an optimal file replication rule that allocates resources to each file based on its popularity and size. To the best of our knowledge, this work is the first attempt to theoretically investigate the problem of resource allocation for replica creation to achieve global file searching optimization in MANETs. We further propose a file replication protocol based on the rule, which approximates the minimum global querying delay in a fully distributed manner. Our experiment and simulation results show the superior performance of the proposed protocol in comparison with other representative replication protocols.

The remainder of this paper is organized as follows. Section 2 presents an overview of the related works. Section 3 presents the analysis and modeling of the influence of the resource allocation on file searching efficiency under two representative mobility models. Section 4 details the file replication protocol. In Section 5, 6, and 7, the performance of our proposed system is evaluated through real traces and synthesized mobility. Section 8 concludes the paper.

2 Related Work

2.1 File Sharing in Normal MANETs

The topic of file replication for efficient file sharing applications in MANETs has been studied recently. In [10]–[12], individual or a group of nodes decide the list of files to replicate according to file visiting frequency. Hara [10] proposed three file replication protocols: Static Access Frequency (SAF), Dynamic Access Frequency and Neighborhood (DAFN) and Dynamic Connectivity based Grouping (DCG). In SAF, each node replicates its frequently queried files until its available storage is used up. SAF may lead to many duplicate replicas among neighboring nodes when they have the same interested files. DAFN eliminates duplicate replicas among neighbors. DCG further reduces duplicate replicas in a group of nodes with frequent connections. It sums the access frequencies of all nodes in a group and creates replicas for files in the descending order. Though DAFN and DCG enable replicas to be shared among neighbors, neighboring nodes may separate from each other due to node mobility. Also, they incur high traffic load in identifying duplicates or managing groups.

Zhang *et al* [11] proposed to let each node collect access statistics from neighbors to decide the creation or relinquishment of a replica. Duong and Demeure [12] proposed to group nodes with stable connections and let each node checks its group members' potential possibility of requesting a file and their storage status to decide replicate the file or not. Also, each node notifies all other nodes in the system about its newly created files by broadcasting. Yin and Cao [9] proposed to cache popular files on the intersection nodes of file retrieval paths. Though it is effective for popular files, it fails to utilize all storage space in nodes other than the intersection nodes.

Gianuzzi [21] investigated the probability of acquiring a file, which has n replicas in the network, from the potentially partitioned network. He also studied the file retrieval performance when erasure coding [22] is employed to segment files. Chen [23] discussed how to decide the minimal number of mobile servers needed to satisfy the requirement that every data item can be obtained within at most k $(k \geq 1)$ hops by any node in the system. Moussaoui *et al.* [8] proposed two steps of file replication, primary replication and dynamic replication, to disseminate replicas in the network in order to meet user needs and prevent data loss in the case of network partition. In the primary replication step, newly created files are distributed evenly among nodes that are three hops away from each other through replication. Later, when the network topology changes, dynamic replication is conducted, in which each node checks its visiting frequency to a file or the density of a file to make the replication decision.

2.2 File Sharing in Disconnected MANETs/DTNs

Huang *et al.* [13] discussed how to cache files in servers to realize the optimal file availability to mobile users in WiFi-based wireless networks based on node mobility pattern, AP topology and file popularity. However, the file servers in this paper are fixed nodes connecting to APs, while we consider a more general P2P scenario, in which all mobile nodes are both file servers and clients. Pitkanen and Ott [17] proposed the DTN storage module to leverage the DTN store-carry-and-forward paradigm

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2014.2308211, IEEE Transactions on Computers

and make DTN nodes keep a copy of a message for a longer period of time required by forwarding. Gao *et al.* [14] proposed a cooperative caching method in DTNs by copying each file to the node in each network central location, which is frequently visited by other nodes. When the central node is full, less-popular replicas are moved to its neighbor nodes. However, central nodes may be frequently changed, leading to frequent file transfers and high overhead. QCR [15] leverages caching for multimedia content dissemination in opportunistic networks. It considers data retrieval delay and the probability that users will require the same content based on previously experiences to decide the caching policy. SEDUM [20] also uses replication to create redundant messages in routing for DTNs, thereby enhancing routing success rate. PSEPHOS [16] considers three factors including data access frequency, user preference and node mobility to decide the data caching. The author in [18] considers the contact duration in DTNs to better improve data retrieval probability through replication. In [19], both social community structures and contact duration among nodes are considered to decide where and how much to cache data in DTNs. However, these methods fail to consider that the mobility of a node affects the availability of files or messages and further optimize the replication distribution to enhance file availability or routing success rate.

2.3 Modeling Replication Optimization Problem

We present the general process to model the expected file querying delay with file replication. We let $\hat{m}_i$ be the probability that a node's newly met node in the coming time interval is node i, which reflects the meeting ability of the files on node i. We also use X_{ij} to denote whether node i owns file j or its replication. Then, the average number of time intervals needed to meet a specific file, say file j, can be represented as:

$$\overline{\hat{T}_j} = \frac{1}{\sum_{i=1}^{N} \hat{m}_i X_{ij}} \tag{1}$$

Then, the average number of intervals needed to satisfy a request is

$$\overline{\hat{T}} = \sum_{j=1}^{F} q_j \overline{\hat{T}_j} = \sum_{j=1}^{F} \frac{q_j}{\sum_{i=1}^{N} \hat{m}_i X_{ij}}, \tag{2}$$

where q_j is the probability of querying file j. With Formula (2), we can formulate the global optimization problem as minimizing $\overline{\hat{T}}$, which can be further utilized to deduce the optimal replication rule.

However, the calculation of $\hat{m}_i$ may be complex and makes the minimization problem non-trivial. We will discuss how this is handled with the two common mobility models in Section 3.

3 Theoretical Analysis of Globally Optimal File Replication

3.1 Node Movement Models

Recall that we consider two types of MANETs (i.e., normal MANETs and disconnected MANETs) in this paper. In the research area of MANETs, usually, the random waypoint model (RWP) [24] is used for the normal MANETs and the community-based mobility model [25] is used for the disconnected MANETs (and DTNs). Thus, we also use the two models to represent the two types of MANETs in theoretical analysis. We leave the analysis for other mobility models (i.e., created by Bonn Motion Tool [26]) as our future work.

3.1.1 Random Waypoint Model for Normal MANETs

As some MANET replication protocols [10], [11], [21], we use the random waypoint model (RWP) [24] to model node mobility in normal MANETs. In RWP, nodes repeatedly move to a randomly selected point at a random speed, which means each node has roughly similar probability to meet other nodes. However, nodes usually have different probabilities of meeting nodes in reality (i.e., nodes with faster speed can meet others more frequently). We hence let each node have a randomly obtained speed, rather than continuously varying a node's speed in different paths as in the normal RWP model.

3.1.2 Community-Based Mobility Model for Disconnected MANETs

The community-based mobility model [25] has been used in some content dissemination or routing algorithms for disconnected MANETs/DTNs [27], [28] to depict node mobility. In this model, the entire test area is split into different sub-areas, denoted as caves. Each cave holds one community. A node belongs to one or more communities (i.e., home community). The routines and (or) social relationships of a node tend to decide its mobility pattern. When moving, a node has probability P_{in} to stay in the home community and probability $1 - P_{in}$ to visit a foreign community. A node moves within its home communities for most of the time (i.e., P_{in} usually is large). Please refer to [25] for more detail.

3.1.3 Assumptions and Limitations

With above two mobility models, our analysis replies on two assumptions: 1) the probability of meeting a certain node is the same for all nodes (RWP model) or all nodes in its home community (community-based model) and 2) nodes move independently in the network (both models). The two assumptions may not hold in real cases, which limits the applicability of the analysis results in our paper to different real scenarios. However, the analysis results can provide instructions on file replication because the two models can represent key characteristics in real mobility and have been widely used in research works [10], [11], [21], [27], [28]. We also have briefly discussed how to expand the analysis to general scenarios, which do not have the two assumptions, in Section 2.3 and 3.2.3. Due to the complexity of such general cases, we leave the detailed research without the two assumptions to future work.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2014.2308211, IEEE Transactions on Computers

3.2 Theoretical Analysis

TABLE 1: Notations in analysis.

Notation	Meaning
q_j	The probability of querying file j in the system
m_i	The probability that the next encountered node is node i
p_j	The probability of obtaining file j in the next encountered node
N	Total number of nodes
V_i	Node i's meeting ability (i.e., frequency of meeting nodes)
S_i	Storage space of node i
$\overline{V}$	Average meeting ability of all nodes in the system
F	Total number of files in the system
b_j	Size of file j
X_{ij}	Whether node i contains file j or not
V_{jk}	Meeting ability of the k^{th} node that holds file j
n_j	The number of nodes holding file j or its replicas
A_j	Allocated resource for file j for replication
$\overline{T_j}$	Average number of time intervals needed to meet file j
$\overline{T}$	Average number of time intervals needed to meet a file
$\mathcal{R}$	Total amount of resource in the system
P_j	Priority value of file j, $P_j = \sqrt{q_j/b_j}$

In this section, we theoretically analyze the influence of the file replica distribution on the overall query efficiency in MANETs under the two mobility models following the process introduced in Section 2.3. Please refer to Table 1 for the meanings of notations.

3.2.1 Optimal File Replication with the RWP model

In the RWP model, we can assume that the inter-meeting time among nodes follows exponential distribution [29], [30]. Then, the probability of meeting a node is independent with the previous encountered node. Therefore, we define the **meeting ability** of a node as the average number of nodes it meets in a unit time and use it to investigate the optimal file replication. Specifically, if a node is able to meet more nodes, it has higher probability of being encountered by other nodes later on. We use m_i to denote the probability that the next node a request holder meets is node i. Then, m_i is proportional to node i's meeting ability (i.e., V_i). That is

$$m_i = \frac{V_i}{\sum_{k=1}^{N} V_k} = \frac{V_i}{\overline{V}N} \tag{3}$$

where N denotes the total number of nodes and $\overline{V}$ denotes the average meeting ability of all nodes.

We use vector $(V_{j1}, V_{j2}, \ldots, V_{jn_j})$ to denote the meeting abilities of a group of nodes holding file j or its replica, where n_j is the number of file j (including replicas) in the system. Then, the probability that a node obtains its requested file j from its encountering node is the sum of the probabilities of encountering nodes that hold file j or its replica. That is,

$$p_j = \sum_{i=1}^{N} m_i X_{ij} = \sum_{i=1}^{N} \frac{V_i}{\overline{V}N} X_{ij} = \sum_{k=1}^{n_j} \frac{V_{jk}}{\overline{V}N} \tag{4}$$

where X_{ij} is a zero-one variable that denotes whether node i contains file j or its replica.

As stated above, a node's probability of being encountered by other nodes is proportional to the meeting ability of the node. This indicates that files residing in nodes with higher meeting ability have higher availability than files in nodes with lower meeting ability. So we take into account both meeting ability and storage in measuring a node's resource. When a replica is created in a node, it occupies the memory on the node. Also, its probability of being met by others is decided by the node's meeting ability. This means that the replica naturally consumes both the storage resource and the meeting ability resource of the node. Therefore, we denote the resource on a node by S_iV_i, in which S_i denotes node i's storage space and V_i denotes its meeting ability. Then, the total amount of resource in the system ($\mathcal{R}$) is:

$$\mathcal{R} = \sum_{i=1}^{N} S_i V_i \tag{5}$$

Thus, the total resource allocated to file j is:

$$R_j = b_j \sum_{k=1}^{n_j} V_{jk} \tag{6}$$

where b_j is the size of file j. Based on Equation (6), Equation (4) can be represented as

$$p_j = \frac{b_j \sum_{k=1}^{n_j} V_{jk}}{b_j \overline{V} N} = \frac{R_j}{b_j \overline{V} N} \tag{7}$$

Thus, the probability of meeting file j after k ($k = 1, 2, 3, \cdots$) time intervals (i.e., average inter-meeting time among nodes) is

$$(1 - p_j)^{k-1} p_j$$

and the average number of time intervals needed for a node to meet a node containing file j is

$$\overline{T_j} = \sum_{k=1}^{\infty} k(1 - p_j)^{k-1} p_j = \frac{1}{p_j} = \frac{b_j \overline{V} N}{R_j} \tag{8}$$

We use $q_j \in [0, 1]$ to denote the probability of a node's originating a request for file j in the system during a unit of time period ($\sum_{j=1}^{F} q_j = 1$). Then, the average number of intervals needed to satisfy a request is

$$\overline{T} = \sum_{j=1}^{F} q_j \overline{T_j} = \sum_{j=1}^{F} q_j \frac{b_j \overline{V} N}{R_j} = \overline{V} N \sum_{j=1}^{F} \frac{q_j b_j}{R_j} \tag{9}$$

We aim to minimize the global file querying delay (i.e., $\overline{T}$) by file replication. According to Equation (9), $\overline{T}$ is decided by q_j, b_j and R_j, and the values of q_j and b_j are decided by the system. Thus, the problem of optimal resource allocation is then converted to finding the optimal amount of resource (R_j) for each file j under the restriction of total available resource in order to achieve the minimum average querying delay.

Suppose $B_j = q_j b_j$, with Equations (5) and (9), the problem of optimal resource allocation is expressed by

$$\min(\overline{T}) = \min\{\sum_{j=1}^{F} \frac{q_j b_j}{R_j}\} = \min\{\sum_{j=1}^{F} \frac{B_j}{R_j}\} \tag{10}$$

subject to:

$$\sum_{j=1}^{F} R_j \leq \mathcal{R}.$$

Equation (9) also indicates that each R_j should be as large as possible in order to minimize $\overline{T}$. Therefore, we assume all resources ($\mathcal{R}$) are allocated.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2014.2308211, IEEE Transactions on Computers

$$\sum_{j=1}^{F} R_j = \mathcal{R} \tag{11}$$

By applying Formula (11), Formula (10) is changed to

$$\min(\overline{T}) = \min\{\frac{B_1}{R_1} + \frac{B_2}{R_2} + \cdots + \frac{B_F}{\mathcal{R} - (R_1 + R_2 + \cdots + R_{F-1})}\} \tag{12}$$

Next, we try to find the value of R_j $(1 \leq j \leq F-1)$ that satisfies Formula (12). Specifically, we first calculate the first order (necessary) condition by differentiating $\overline{T}$ on each R_j $(1 \leq j \leq F-1)$ respectively, and find the value of R_j that makes the differentiated formula equal 0. The resultant formulas after differentiation are

$$\frac{B_1}{R_1^2} - \frac{B_F}{\{\mathcal{R} - (R_1 + R_2 + \cdots + R_{F-1})\}^2} = 0 \tag{13}$$

$$\cdots\cdots\cdots$$

$$\frac{B_{F-1}}{R_{F-1}^2} - \frac{B_F}{\{\mathcal{R} - (R_1 + R_2 + \cdots + R_{F-1})\}^2} = 0 \tag{14}$$

Combine all of the above $F-1$ equations, we get

$$\frac{B_1}{R_1^2} = \frac{B_2}{R_2^2} = \frac{B_3}{R_3^2} = \cdots = \frac{B_{F-1}}{R_{F-1}^2} = \frac{B_F}{R_F^2} \tag{15}$$

To achieve the minimal average delay, the second order (sufficient) condition should be larger than 0 as below:

$$\frac{-2B_1}{R_1^3} - \frac{-2B_F}{\{\mathcal{R} - (R_1 + R_2 + \cdots + R_{F-1})\}^3} > 0 \tag{16}$$

$$\cdots\cdots\cdots$$

$$\frac{-2B_{F-1}}{R_{F-1}^3} - \frac{-2B_F}{\{\mathcal{R} - (R_1 + R_2 + \cdots + R_{F-1})\}^3} > 0 \tag{17}$$

If Equation (15) is true, based on Equation (11), Formulas (16) and (17) can be transformed to below.

$$(\frac{1}{R_F} - \frac{1}{R_1})\frac{2B_1}{R_1^2} > 0 \tag{18}$$

$$\cdots\cdots\cdots$$

$$(\frac{1}{R_F} - \frac{1}{R_{F-1}})\frac{2B_{F-1}}{R_{F-1}^2} > 0 \tag{19}$$

When $R_F < R_j$ $(j \in [1, F-1])$, Equations (18) and (19) (and also the second order condition) are satisfied. Recall that above result is obtained when we replace R_F with $\mathcal{R} - (R_1 + R_2 + \cdots + R_{F-1})$ in Equation (10). If we replace R_k $(k \in [1, F])$ with $\mathcal{R} - (R_1 + \cdots R_{k-1} + R_{k+1} + \cdots R_F)$, the second order is also satisfied when $R_k < R_j$ $(j \in [1, F], j \neq k)$. In summary, the second order is satisfied when the resource allocated for one file is less than the resource allocated for any other file. This condition is always true because there always exists a file with the minimum allocated resource. Therefore, as long as the first order condition (Equation (15)) is satisfied, the second order condition is also satisfied.

Then, according to Equation (11) and Equation (15), we can see that the optimal allocation is

$$R_j = \frac{\sqrt{B_j}}{\sum_{k=1}^{F} \sqrt{B_k}} \mathcal{R} \ (j = 1, 2, 3, \cdots, F) \tag{20}$$

This means that the optimal resource allocation is achieved through the square root policy, i.e., the portion of resource for file j is in direct proportion of the square root of B_j:

$$R_j \propto \sqrt{B_j} \Rightarrow b_j \sum_{k=1}^{n_j} V_{jk} \propto \sqrt{b_j q_j} \tag{21}$$

That is

$$\sum_{k=1}^{n_j} V_{jk} \propto \sqrt{\frac{q_j}{b_j}} \Rightarrow \sum_{k=1}^{n_j} V_{jk} \propto P_j \tag{22}$$

We call $\sqrt{q_j/b_j}$ the *Priority Value* (P) of file j as it represents the relative priority in acquiring resource for the global optimization on querying delay.

Based on Formula (22), we derive the Optimal File Replication Rule (OFRR) that gives the direction for the optimal resource allocation for each file that leads to the minimum average file querying delay under the RWP model.

OFRR. *In order to achieve minimum overall file querying delay, the sum of the meeting ability of replica nodes of file j should be proportional to $P_j = \sqrt{q_j/b_j}$.*

3.2.2 Optimal File Replication with the Community-Based Mobility Model

In this section, we conduct the analysis under the community-based mobility model. Unless otherwise specified, we use the same notations in Table 1 (which is for the RWP model) but add $'$ to each notation to denote that it is for the community-based mobility model. Recall that in the RWP model, we can assume that the inter-meeting time of nodes follows exponential distribution. Based on this assumption, we can calculate the probability that a newly met node is node i (i.e., m_i), which is used to find the expected time $\overline{T}$ to satisfy a request and finally deduce OFRR to minimize $\overline{T}$. However, under the community-based mobility model, this assumption does not hold [31]. This makes it difficult to calculate m_i, which makes the process of minimizing the overall delay $\overline{T'}$ a formidable problem. To deal with this problem, rather than considering meeting ability, we consider each node's **satisfying ability**. It is defined as a node's ability to satisfy queries in the system (denoted by V_i') and is calculated based on the node's capacity to satisfy queries in each community.

We use N_c to denote the number of nodes in community c. Then, community c holds $\frac{N_c}{N}$ fraction of nodes in the system. Node i's satisfying ability to community c depends on both the number of different nodes in c it meets in a unit time period (denoted by M_{ic}), and the number of queries generated by nodes in c. In this model, since nodes' file interests are stable during a certain time period, we assume that each node's querying pattern (i.e., different querying rates for different files) remains stable during a certain period of time.

Then, the number of nodes in a community represents the number of queries for a given file generated in this community. As a result, a file holder has low ability to satisfy queries from a small community. Thus, we integrate each community's fraction of nodes (i.e., $\frac{N_c}{N}$) into the calculation of the satisfying ability. Therefore,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2014.2308211, IEEE Transactions on Computers

$$V_i' = \sum_{c=1}^{C} M_{ic} \frac{N_c}{N} \tag{23}$$

where C is the total number of communities.

Given n_j nodes that hold file j or its replicas, we again use vector $(V_{j1}', V_{j2}', \ldots, V_{jk}', \ldots, V_{in_j}')$ to denote the satisfying abilities of these nodes. Then, the overall ability of nodes in the system to satisfy requests for file j (denoted by O_j) is the sum of all the satisfying abilities times a redundancy elimination factor α.

$$O_j = \alpha \sum_{k=1}^{n_j} V_{jk}' \ (\alpha \in [0,1]) \tag{24}$$

α is added because different holders of file j may meet the same requester for file j in the same time unit. Since the requester has only one request for file j, only the first meeting satisfies the file request, and the subsequent meetings do not satisfy any requests for file j. In other words, α denotes the "discount" on the overall satisfying ability considering the fact that the satisfying abilities of different file holders may overlap.

Then, the number of time intervals (i.e., average inter-meeting time among nodes) needed to satisfy a request for file j is

$$\overline{T_j'} = \frac{1}{O_j} = \frac{1}{\alpha \sum_{k=1}^{n_j} V_{jk}'} \tag{25}$$

Recall that b_j denotes the size of file j and q_j denotes the probability of initiating a request for file j from nodes in the system. Similar to Equation (6), the total resource (satisfying resource and storage resource) allocated to file j can be represented by $R_j' = b_j \sum_{k=1}^{n_j} V_{jk}'$. As a result, the average number of time intervals needed to satisfy a request in the system is

$$\overline{T'} = \sum_{j=1}^{F} q_j \overline{T_j'} = \sum_{j=1}^{F} q_j \frac{1}{\alpha \sum_{k=1}^{n_j} V_{jk}'} = \frac{1}{\alpha} \sum_{j=1}^{F} \frac{q_j b_j}{R_j'} \tag{26}$$

Then, the problem of optimal resource allocation can be expressed by

$$\min(\overline{T'}) = \min\{\sum_{j=1}^{F} \frac{q_j b_j}{R_j'}\} = \min\{\sum_{j=1}^{F} \frac{B_j}{R_j'}\} \tag{27}$$

subject to:

$$\sum_{j=1}^{F} R_j' \leq \mathcal{R}.$$

We can find that Equation (27) is the same as Equation (10). Then, we follow the same process after Equation (10) and deduce the OFRR rule in disconnected MANETs as

$$\sum_{k=1}^{n_j} V_{jk}' \propto \sqrt{\frac{q_j}{b_j}} \Rightarrow \sum_{k=1}^{n_j} V_{jk}' \propto P_j \tag{28}$$

We see that the OFRR under the community-based mobility model (Equation (28)) is the same as the OFRR deduced with the RWP model (Equation (22)) except that V_{jk}' is the satisfying ability (Equation (23)) in the former while is the meeting ability (defined in Table 1) in the later. It is intriguing to find that Equation (23) turns to be the same as the definition of V_i in Table 1 if the number of community is 1. This means that the OFRR expressed by Equation (22) is a special case of the OFRR expressed by Equation (28). **As a result, our previously deduced OFRR can be the OFRR for MANETs under the two mobility models.**

It is interesting to find that the OFRR matches the "square root assignment rule" derived by Kleinrock [32] for the link capacity assignment in wireless communication to maximize the network efficiency. It also matches the findings in [33] that when file servers may be unavailable due to node dynamism, the wired P2P content distribution systems can achieve the maximum file hit rate when available storage is allocated in proportion to a constant value plus $ln(q_j/b_j)$ for each file.

3.2.3 Extension to General Node Mobility Models

In the above two subsections, we deduced the OFRR rule in RWP mobility model and community-based mobility model following the basic idea in Section 2.3. However, above analysis relies on two assumptions mentioned in Section 3.1.3, which may not hold in general node mobility models. Therefore, it is nontrivial to extend above analysis to general cases directly. Specifically, in certain mobility models, different nodes may have different visiting preferences or patterns, making different node's probabilities of meeting node i in the next encountering ($\hat{m}_i$) lack a direct general expression.

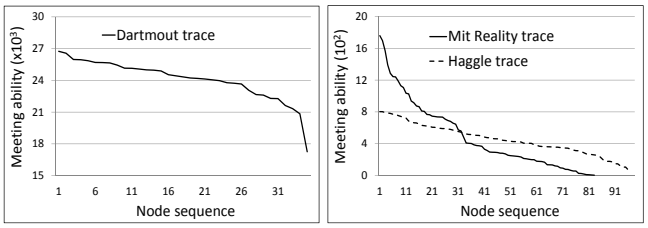
However, there are some ways to make the analysis in general cases possible. For example, we can incorporate new factors into $\hat{m}_i$ to express each node's distinct pattern, e.g., active levels and community identities. These factors usually represent how frequent a node meets other nodes. We can also first measure the meeting abilities of different nodes in a real scenario. Then, we can assign labels to each node to indicate its rough meeting ability. With these simplifications, $\hat{m}_i$ can be expressed and the analysis can be conducted. We leave the research following such a direction to future work.

On the other hand, there are possibly fixed nodes in the system, which are naturally supported in our analysis. This is because we only care a node's storage and meeting ability regarding creating file replicas. Though fixed nodes do not move, they can meet other nodes, which means their meeting abilities can be measured or even formulated. As a result, fixed nodes are regarded the same as mobile nodes in the system.

3.3 Meeting Ability Distribution in Real Traces

We measured the meeting ability distribution from real traces to confirm the necessity to consider node meeting ability as an important factor in the resources allocation in our design. Specifically, for normal MANETs, we used the Dartmouth trace [34], which was obtained through an outdoor project in Dartmouth College. The trace provides position records of 35 laptop nodes moving randomly and independently across different sections of an open field. For disconnected MANETs, we used the MIT Reality trace [35] and the Haggle trace [36]. In the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2014.2308211, IEEE Transactions on Computers



(a) In a connected MANET. (b) In disconnected MANETs.

Fig. 1: Meeting ability distribution.

former, 97 smart phones were distributed to students and faculties at MIT. In the latter, 98 iMotes were assigned to scholars attending the Infocom'06 conference. In both traces, nodes' contact records were recorded.

For each trace, we measured the meeting abilities of all nodes and ranked them in decreasing order, as shown in Figure 1(a) and Figure 1(b). We see that in all the three traces, node meeting ability is distributed in a wide range. This matches our previous claim that nodes usually have different meeting abilities. Also, it verifies the necessity of considering node meeting ability as a resource in file replication since if all nodes have similar meeting ability, replicas on different nodes have similar probability to meet requesters, and hence there is no need to consider meeting ability in resource allocation.

4 Distributed File Replication Protocol

In this section, we propose a distributed file replication protocol that can approximately realize the optimal file replication rule (OFRR) with the two mobility models in a distributed manner. Since the OFRR in the two scenarios (i.e., Equation (22) and Equation (28)) have the same form, we present the protocol in this section without indicating the specific scenario. We first introduce the challenges to realize the OFRR and our solutions to these challenges. Then, we propose a replication protocol to realize OFRR and analyze the effect of the protocol.

4.1 Challenges and Solutions to Achieve the OFRR

Challenge 1: resource allocation without a central server. OFRR shows that in order to realize the globally optimal querying delay, each file's popularity (q_j) and size (b_j), and the system resource ($\mathcal{R}$) information (both node storage size and moving ability) must be known in order to decide the portion of resource for each file for replica creation. Specifically, suppose there are F files in the system with $b_1q_1 \cdots b_Fq_F$ and total resource $\mathcal{R}$, the resource allocated to file j (R_j) should be

$$R_j = \mathcal{R} \times \sqrt{b_j q_j} / \sum_{k=1}^{F} \sqrt{b_k q_k} \tag{29}$$

Then, an intuitive way to achieve this goal is to setup a central server to collect all above-mentioned information, conduct the resource allocation for each file, and distribute the information to file owners to replicate their files. However, the nature of the distributed network, node mobility and transmission range constraint become obstacles of building such a central service. For example, since nodes are constantly moving and have limited communication ranges, it is impossible for each node to update its information to or receive information from the server timely. Thus, a severe challenge is to enable a node to distributively figure out the proper portion of resource for each of its files without a central server.

Even when each node knows $\sqrt{b_j q_j} / \sum_{k=1}^{F} \sqrt{b_k q_k}$ of each of its files, the total amount of resources available in the system may change due to node joins and departures, which makes it difficult for a node to calculate the portion of resource of each of its file (R_j). For example, suppose there are only two files in the system, say f_1 and f_2, and the ratio of their allocated resources should be 4:1. If the total amount of resource $\mathcal{R} = 40$, the amount of resource allocated to f_1 is 32. If $\mathcal{R} = 60$, the amount for f_1 should be adjusted to 48. Further, the time-varying file popularity (q_j) make the problem even more formidable. Therefore, OFRR cannot be simply realized by letting each node distribute replicas of a file until an absolute amount of resource is used.

Solution to Challenge 1: resource competition. OFRR (i.e, Formula (22)) requires that for each file, the sum of its replica nodes' meeting abilities, $\sum_{k=1}^{n_F} V_{Fk}$, is proportional to its priority value P. In other words, OFRR can be shown by

$$P_1 / \sum_{k=1}^{n_1} V_{1k} = P_2 / \sum_{k=1}^{n_2} V_{2k} \cdots = P_F / \sum_{k=1}^{n_F} V_{Fk} \tag{30}$$

where n_j ($j \in [1, 2, \cdots, F]$) represents the number of replica nodes of file j. Then, we can let each file, say file j, periodically compete for the resource with its current $P_j / \sum_{k=1}^{n_j} V_{jk}$. In one competition, the file with the highest $P_j / \sum_{k=1}^{n_j} V_{jk}$ wins and receives resource for one replica. After a file creates a replica, its $P_j / \sum_{k=1}^{n_j} V_{jk}$ decreases. The competition stops when all available resource is allocated and no one can win a competition. Thus, files with larger $P_j / \sum_{k=1}^{n_j} V_{jk}$ win more competitions and receive more resource and files with smaller $P_j / \sum_{k=1}^{n_j} V_{jk}$ only win few competitions and receive less resource. The competition gradually lets each file receive its deserved portion of resource based on OFRR. By enabling file owners to distributively compete for resource for their files, we can realize OFRR without a central server.

Challenge 2: competition for distributed resource. In a MANET, all available resource is scattered among different nodes moving around in the network. This poses three problems. First, different file owners are scattered and can hardly gather together to conduct the resource competition. Second, after a file is replicated to a number of nodes, it is difficult to collect the popularity of the replicas to update the P of the file. Third, since the number of nodes met by a file owner is limited, a single file owner cannot distribute replicas efficiently and quickly. We propose a work-around for this problem. Specifically, we regard a file and its newly created replica as two different files, which participate in further competition independently with evenly split P. However, this brings another challenge: since replica nodes of a file are scattered in the network, how to ensure that the overall

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2014.2308211, IEEE Transactions on Computers

$\sum V_{jk}$ is proportional to the overall P of the file? We solve it in next subsection.

Note the competition used in the description is not to show that resources are very limited. It is only to show the process of resource allocation, which can be viewed as a probability based resource allocation algorithm. Such a solution increases the complexity of the system. However, this is caused by the distributed nature of MANETs. We will investigate how to reduce the complexity in the next step. For example, we can check whether files can reduce the frequency of competition but still get the deserved amount of resources.

Solution to Challenge 2: distributive competition on selective resources. In the solution to Challenge 1, each file periodically competes for resource with its current $P_j/\sum_{k=1}^{n_j} V_{jk}$. However, as previously mentioned, it is a challenge to keep the overall P proportional to the overall $\sum V_{jk}$ while replica holders are scattered. We indirectly resolve this problem by keeping the average V of the replica nodes of a file close to $\overline{V}$. Then, Formula (22) can be re-expressed as

$$n_j * \overline{V} \propto \sqrt{\frac{q_j}{b_j}} \Rightarrow n_j \propto \sqrt{\frac{q_j}{b_j}} \Rightarrow n_j \propto P_j \qquad (31)$$

In such a case, when the number of replicas of each file is proportional to its $P_j = \sqrt{q_j/b_j}$, OFRR is satisfied. To attain this goal, we let each node deliberately select a neighbor node to create replicas of its file so that the average meeting ability of replica nodes of the file is equal or closest to $\overline{V}$. Considering the diverse mobility of nodes in the network, a node should be able to find replica nodes whose average meeting ability equals $\overline{V}$ during its movement. Then, based on Equation (31), each node only needs to consider the P of each file in the resource completion. Upon winning a competition for a file, a node splits the file's P evenly between the file and the replica. After this, the popularity of each file/replica is continuously updated based on the number of requests received for it in a unit time period, which is used to update its priority value P.

When a replica is deleted in the competition, we cannot reverse the process of priority split because it is very difficult to track locations of the holders of the original file in a distributed manner due to the mobility of nodes in MANETs. Fortunately, we can use the querying popularity q to handle this problem. In this case, the qs (or Ps) of other replicas of the file increase since they receive more requests for the file as the total amount of requests is stable. That is, the sum of the replicas' Ps equals the overall P of the original file j (P_i). The increase of priority value caused by the replica deletion can be regarded as the reversed process of priority split. As a result, the number of replicas of each file is proportional to the sum of meeting ability of its replica nodes, realizing Formula (22).

4.2 Design of the File Replication Protocol

The two solutions to handle the challenges in achieving OFRR described above are maximal approximation to realize the OFRR in a distributed manner. Based on the solutions, we propose the Priority Competition and Split file replication protocol (PCS). We first introduce how a node retrieves the parameters needed in PCS and then present the detail of PCS.

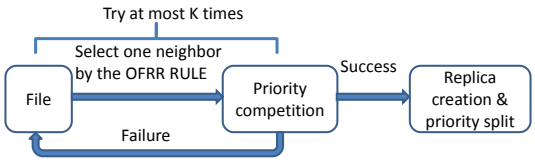


Fig. 2: Replica distribution process.

In PCS, each node dynamically updates its meeting ability (V_i) and the average meeting ability of all nodes in the system ($\overline{V}$). Such information is exchanged among neighbor nodes. We explain the detail of this step in Section 4.3. Each node also periodically calculates the $P_j = \sqrt{q_j/b_j}$ of each of its files. The q_j is calculated by $q_j = u_j/U$, where u_j and U are the number of received requests for the file and the total number of queries generated in a unit of time period, respectively. Note that U is a pre-defined system parameter.

In the solution to Challenge 2, nodes replicate files distributively and select replicate nodes to ensure that the average meeting ability of replica nodes of a file the closest to $\overline{V}$. That is, $\overline{V_{n'_j}} \approx \overline{V}$, where n'_j is the number of created replicas of file j and $\overline{V_{n'_j}}$ is the average meeting ability of these replica nodes. Therefore, each node needs to keep track of n'_j and $\overline{V_{n'_j}}$ of each of its file. After creating a replica, the node increases n'_j by 1 and updates $\overline{V_{n'_j}}$ using the V of the new replica node.

With the above information, we introduce the process of the replication of a file in PCS. Based on OFRR, since a file with a higher P should receive more resource, a node should assign higher priority to its files with higher P to compete resource with other nodes. Thus, each node orders all of its files in descending order of their Ps and creates replicas for the files in a top-down manner periodically. Algorithm 1 presents the pseudo-code for the process of PCS between two encountered nodes. In detail, suppose node i needs to replicate file j on the top of the list, as shown in Figure 2, it keeps trying to replicate file j on nodes it encounters until one replica is created or K attempts have been made. If file j is replicated, its P is split and it is inserted to its new place in the list. Next, the node fetches the file from the top of the list and repeats the process. If file j fails to replicate after K attempts, the node stops launching competition until the next period.

Following the solution to Challenge 2, a replicating node should keep the average meeting ability of the replica nodes for file j around $\overline{V}$. Node i first checks the meeting abilities of neighbors and then chooses the neighbor k that does not contain file j and makes $V_{n'_j}^{new} = (n'_j\overline{V_{n'_j}} + V_k)/(n'_j + 1)$ the closest to $\overline{V}$ as the replica node candidate. It is possible that $V_{n'_j}^{new}$ is far away from $\overline{V}$. Therefore, we set a deviation range r. If creating a replica in the selected neighbor makes $(V_{n'_j}^{new} - \overline{V}) > r$, then the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2014.2308211, IEEE Transactions on Computers

node does not replicate file j in the selected neighbor until it has a different set of neighbors.

In the case that $(V_{n'_j}^{new} - \overline{V}) \leq r$, if the selected neighbor's available storage is larger than the size of file j (S_j), it creates a replica for file j directly. Otherwise, a competition is launched among the replica of file j and replicas already in the neighbor node based on their Ps. The priority value of the new replica is set to half of the original file's P. According to the solution to Challenge 1, the probability that a replica wins the resource competition is proportional to its P, i.e., a replica's probability of being selected to be removed is inversely proportional to its P. Then, suppose there are d replicas in competition, we let each replica be responsible for a range that equals its $1/P$ in range space $[0, \sum_{k=1}^{d} 1/P_k]$. The neighbor node randomly chooses a number in $[0, \sum_{k=1}^{d} 1/P_k]$, and the replica whose range owns the number is selected to be removed. The neighbor node repeats above process until available storage is no less than the size of file j.

If file j is among the selected files, it fails the competition and will not be replicated in the neighbor node. Otherwise, all selected files are removed and file j is replicated. If file j fails, node i will launch another attempt for file j until the maximum number of attempts (K) is reached. The setting of K attempts is to ensure that each file can compete with a sufficient subset of replicas in the system. If node i fails to create a replica for file j after K attempts, then replicas in node i with smaller Ps than file j are unlikely to win a competition. Thus, at this moment, node i stops replicating files until next round. Finally, all available resource in the system is allocated to replicas according to their Ps (i.e., OFRR is realized).

According to the Solution to Challenge 2, we regard file j's replica as a "different" file from file j in PCS. Therefore, if node i successfully creates a replica for file j, it splits the file's P evenly between file j and the new replica. Thus, each file's priority is $P/2$. After the splitting, the two copies of file j involve in further resource competition independently. Note that we do not split files in the PCS algorithm but split the priority value of a file when a replica is created.

The replication for a file stops when the communication session of the two involved nodes ends. Then, the node will continue the replication process for the file again after excluding the disconnected node from the neighbor node list. Since the popularity of files' popularity and Ps and available system resource change as time goes on, each node periodically executes PCS to dynamically handle these time-varying factors. Each node also periodically calculates the popularity of its files (q_j) to reflect the changes on file popularity (due to node querying pattern and rate changes) in different time periods. The periodical file popularity update can automatically handle file dynamism. The popularity of newly added files will be calculated and hence these files will be considered in resource allocation. Similarly, those of deleted files will not be calculated and hence these file will not be considered in resource allocation.

Algorithm 1 Pseudo-code of PCS between node i and k.

```
i.createReplicasOn(k)     //node i tries to create a replica on node k
k.createReplicasOn(i)     //node k tries to create a replica on node i
Procedure createReplicasOn (node)
  nCount ← 0     //initialize a count
  this.orderFilesByP()     //order files by priority value
  For (each file f in current node)     //try to replica each file
      If (node.compete4File(f) == true)     //competition
          node.createAReplica4(f)     //create a replica if win
      else
          nCount ← nCount+1
      If nCount ≥ K     //try at most K times
          Break
end Procedure
Procedure compete4File()     //Compete for file j
  While (nRemainningMem < j.size())
      nSum ← nTotal ← nRandom ← fFile ← 0     //initilization
      For (each file f (including j) in current node)
          nTotal ← nTotal+1/P_f
      nRandom ← generateARandomNumber() % nTotal
      For (each file f (including j) in current node)
          nSum ← nSum+1/P_f
          If (nSum >= nRandom)
            fFile = f  Break     //pick the file
      If (fFile = j)     //j is the picked file, competition fails
          return false
      Else     //win the competition
          select fFile
  delSelectedFiles()     //delete the selected files
  return true
end Procedure
```

4.3 How to Collect Meeting Ability Information

In a MANET, nodes periodically exchange beacon messages to discover neighbor nodes. The frequency of the beacon messages depends on the mobility of nodes. The size of a beacon message usually is several bytes. To save communication cost, the values of V_i and $\overline{V}$ are piggybacked into beacon messages. Since V_i and $\overline{V}$ are only several bytes, the piggybacking only slightly increases the size of the beacon message. In normal MANETs, a node's meeting ability (V_i) is simply measured by the frequency it meets other nodes. In disconnected MANETs, a node needs to know the distribution of different communities to calculate its satisfying ability (Equation (23)). We then let each node piggyback its community ID and the community information it knows in the beacon message. Also, it's hard to collect the satisfying abilities of all nodes in distributed MANETs in a timely manner since nodes are sparsely distributed. We let each node simply use the average meeting ability of all so far encountered nodes as that for all nodes in the system. As nodes meet more and more nodes, the calculated value can generally represent that of all nodes.

4.4 Analysis of the Effectiveness of PCS

In this section, we briefly prove the effectiveness of PCS. We refer to the process in which a node tries to copy a file to its neighbors as one round of replica distribution.

Recall that when a replica is created for a file with P, the two copies will replicate files with priority $P/2$ in the next round. This means that the creation of replicas will not increase the overall P of the file. Also, after each round, the priority value of each file or replica is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2014.2308211, IEEE Transactions on Computers

updated based on the received requests for the file. Then, though some replicas may be deleted in the competition, the total amount of requests for the file remains stable, making the sum of the Ps of all replicas and the original file roughly equal to the overall priority value of the file. Then, we can regard the replicas of a file as an entity that competes for available resource in the system with accumulated priority P in each round. Therefore, in each round of replica distribution, based on our design of PCS, the overall probability of creating a replica for an original file j, denoted by Ps_j, is proportional to its overall P_j. That is:

$$Ps_j \propto P_j \tag{32}$$

Then, suppose total M rounds of competition are conducted, the expected number of replicas, denoted by n_j, for file j is

$$n_j = MPs_j \Rightarrow n_j \propto P_j \tag{33}$$

Therefore, we conclude that the PCS can realize Equation (31), in which the number of replicas of each file is proportional to its P, thereby realizing the OFRR.

We further briefly discuss the security and incentive considerations for PCS in Appendix B .

5 Performance Evaluation in Normal MANETs with the RWP Model

To evaluate the performance of PCS in normal MANETs, we conducted experiments on both the GENI Orbit testbed [37], [38] and the NS-2 [39] simulator. The GENI testbed consists of 400 nodes equipped with wireless cards. We used the Dartmouth real-world MANET trace [34], which provides the mobility trace of 35 laptops moving in an open field, to drive node mobility in both experiments. In order to validate the adaptability of PCS, we used two routing protocols in the experiments. We first used the StaticWait protocol [40] in the GENI experiment, in which each query stays on the source node waiting for the destination. We then used a probabilistic routing protocol (PROPHET) [6], in which a node routes requests to the neighbor with the highest meeting ability. We set a larger TTL for Static Wait since it needs more time to find a file holder. We used 95% confidence interval when handling the experimental results.

We evaluated the performance of PCS in normal MANETs in comparison with several MANET replication algorithms: SAF [10], DCG [10], PDRS [12] and CACHE [9]. The details of these protocols can be found in Section 2. To better validate our analysis, we also compared PCS with Random, which places replicas on nodes randomly, and OPTM, which is a centralized protocol that calculates the ideal number of replicas for each file based on our derived optimal replication rule. OPTM represents the best possible performance can be obtained by the OFRR. In order to evaluate our protocol under different network sizes and node mobilities, we also conducted simulation on the NS-2 with different network sizes and node mobilities synthesized by the modified RWP model. Due to page limit, the results of these tests are shown in Appendix A.

Table 2 shows the parameters used in experiments, unless otherwise specified. The parameters are determined by referring to the settings in [9], [41] and the real trace. According to the works in [9], [42], we determined the file size and storage space on each node. As the work in [33], the probability of originating requests for different files in each node followed a Zipf distribution and the Zipf parameter was set to 0.7. Initially, files were evenly distributed to each node and no replica existed in the system. In the synthesized mobility, the speed of a node was randomly chosen from the range of [$s/2$, $3s/2$], where s is the configured average node movement speed. Since the real trace does not indicate the communication range of each node, we set the communication range to 100m in the simulation and to 60m in the GENI experiment in order to see the influence of different transmission ranges on the performance. We evaluated the performance of PCS with $K = 3$.

We used the following metrics in the experiments:

- *Hit Rate.* This refers to the percent of requests that are successfully resolved by either original files or replicas. This metric shows the effectiveness of replication protocols in enhancing file availability.
- *Average delay.* This is the average delay of all requests. To make the comparison fair, we included all requests in the calculation. For unresolved requests, we set their delays as the TTL. This metric shows the efficiency of replication protocols in terms of file querying delay.
- *Replication cost.* This is the total number of messages generated in creating replicates. This metric shows the overhead of replication protocols.
- *Cumulative Distribution Function (CDF) of the proportion of replicas.* This is the CDF of the proportion of replicas of each file. This metric reflects the amount of resource allocated to each file for replication.

TABLE 2: Simulation parameters.

	Real trace	Synthesized mobility
Environment Parameters	GENI / NS-2	NS-2
Simulation area	$600m \times 300m$	$1000m \times 1000m$
Node Parameters		
Number of nodes	35	60
Communication range	$60m$ / $100m$	$250m$
Average movement speed	-	$6m/s$
The size of a file (kb)	$1 - 10$	$1 - 10$
Number of files in each node	10	10
Storage space for replicas (kb)	50	50
Query Parameters		
Initialization period	$500s$ / $800s$	$200s$
Querying period	$1500s$ / $1200s$	$600s$
TTL of each request	$1000s$ / $200s$	$200s$
Total time for each test	$3000s$ / $3000s$	$1000s$

5.1 Performance in the Trace-Driven GENI experiments

5.1.1 Hit Rate and Average Delay

Table 3 shows the results of each protocol in the trace-driven experiments on GENI. We see that the hit rates in different replication protocols follow Random<CACHE <SAF<PDRS<DCG<PCS<OPTM and the average delays follow a reverse order: Random>CACHE>SAF> PDRS>DCG>PCS>OPTM. We see that OPTM and PCS

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2014.2308211, IEEE Transactions on Computers

lead to higher hit rate and lower average delay than others. This is attributed to the guidance of OFRR, which aims to minimize the average querying delay by considering both storage and meeting ability as resource to enhance overall file availability. PCS generates slightly lower hit rate and around 20% higher average delay than OPTM. This is because OPTM has the knowledge of all information needed in OFRR beforehand, while PCS has to distribute replicas in a fully distributed manner.

On the contrary, other protocols only replicate files locally, creating redundant replicas and failing to achieve high file availability under node mobility. Random has the worst performance on hit rate and average delay. This is because Random only randomly creates replicas for files and fails to assign more resources to popular files, which are queried more frequently by nodes. CACHE only utilizes the storage on intersection nodes, which indicates that it fails to fully utilize storage space in all nodes. Therefore, it cannot create as many replicas as other protocols and exhibits a low hit rate and a hight delay. In SAF, each node replicates its frequently queried files until its memory is filled up. Then, almost all resources are allocated to popular files. Therefore, SAF cannot optimize query delay globally. In PDRS, a node replicates files interested by its neighbors that have less storage resource than itself. However, as the sharing of replicas is not in the whole group, PDRS only renders a slightly performance improvement over SAF. DCG further improves SAF and PDRS by conducting the file replication on a group level. It eliminates duplicate replicas among group members and uses released memory for other replicas, thereby generating higher hit rate and smaller average delay.

We find that the 1st percentiles of the delays of all protocols are 0.01. This is because some requests are immediately satisfied by direct neighbors. The 99th percentiles of the delays of the protocols approximately follow the relationship on average delay. Above results justify that PCS enhances the file searching efficiency by its global optimization of file availability. The fact that Random leads to worse performance than all methods that give priority to popular files when creating replicas also justify that a resource allocation strategy is necessary for file availability optimization.

TABLE 3: Experimental results of the trace-driven GENI experiments.

Protocol	Hit rate	Average / 1% / 99% delay (s)	Replication cost
Random	0.840139	263.176 / 0.01 / 991.9843	13387
CACHE	0.842454	260.469 / 0.01 / 994.2487	0
SAF	0.857341	259.1768 / 0.01 / 997.1095	0
PDRS	0.863074	256.1983 / 0.01 / 991.2384	175140
DCG	0.878559	251.3287 / 0.01 / 993.3947	67549
PCS	0.898823	240.7031 / 0.01 / 990.4522	28983
OPTM	0.910370	195.1776 / 0.01 / 990.1296	14542

5.1.2 Replication Cost

From the table, we find that the replication costs of different protocols follow PDRS>DCG>PCS>OPTM≈Random>SAF=CACHE=0. PDRS shows the highest replication cost because it needs to broadcast each new file to all nodes in the system. DCG incurs moderate replication cost because group members need to exchange information to reduce duplicate replicas. PCS has a low replication cost because each node only tries at most K times to create a new replica for each file it holds. OPTM and Random have a very low cost since nodes only need to communicate with the central server for replica list. SAF and CACHE have no replication cost since they do not need to exchange information among nodes for file replication. However, SAF generates a lot of redundant replicas, and Random and CACHE lead to low performance.

5.1.3 Replica Distribution

Figure 3 shows the CDF of the proportion of resource allocated to each file for replica creation in different protocols. From the figure, we find that PCS exhibits the closest similarity to OPTM while other protocols follow: DCG≈Random≻CACHE≈PDRS≻SAF, where ≻ means closer similarity to OPTM. Combining the results on average delay, we find an interesting phenomenon: except CACHE and Random, a protocol with closer similarity to OPTM has less average delay. This proves the correctness of our theoretical analysis and the resultant OFRR rule expressed in Formula (22). CACHE has a low performance because it does not utilize all storage space, though it exhibits similarity with PDRS. Random creates replicas for each file randomly without considering their popularity, leading to a low performance since popular files are not replicated with priority. We also observe that the CDFs of the proportion of resource allocated to replicas of DCG, CACHE, PDRS and SAF increases to 0.9 quickly. This is because they allocate most resources to popular files, resulting in a lot of replicas for these files. Though these protocols can reduce the delay of queries for popular files, they cannot reduce the delay for unpopular files. PCS is superior over these protocols because it can globally reduce the query delay for all files.

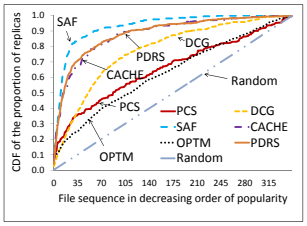


Fig. 3: CDF of the resource allocated to replicas in trace-driven GENI experiment.

5.2 Performance in the Trace-Driven Simulation

5.2.1 Hit Rate and Average Delay

Table 4 shows the results of each protocol in the trace-driven experiments on NS-2. We see the hit rates and average delays of the seven protocols follow the same relationship as in Table 3 due to the same reasons. We find that the average delays of the seven protocols are much less than those in the GENI experiment. This is caused by two reasons. First, the trace-driven simulation adopts the PROPHET for file searching, which can locate files more quickly than the StaticWait searching protocol used in the GENI experiment. Second, the communication range of two nodes (100m) in the simulation is larger than that in the GENI experiment (60m), leading

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2014.2308211, IEEE Transactions on Computers

to shorter searching delay since a node can reach more neighbors. The hit rates of the seven protocols are lower than those in the GENI experiment. This is because the trace-driven simulation used much smaller TTL. The relative performance between different protocols in the simulation matches that in the GENI experiment, which further proves the effectiveness of PCS.

5.2.2 Replication Cost

From Table 4, we find that the replication costs of different protocols follow PDRS>DCG>PCS>OPTM ≈Random>SAF=CACHE=0. This matches the results in Table 3 and the reasons are the same.

5.2.3 Replica Distribution

Figure 4 shows the CDF of the proportion of resource allocated to replicas of each file in the seven protocols. From the figure, we find similar trend as that in Figure 3. That is, except CACHE and Random, a protocol with closer similarity to OPTM has less average delay. This further proves the correctness of our analysis through the trace-driven simulation.

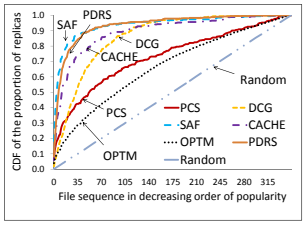


Fig. 4: CDF of the resource allocated to replicas in trace-driven simulation.

TABLE 4: Simulation results of the trace-driven experiments.

Protocol	Hit rate	Average / 1% / 99% delay (s)	Replication cost
Random	0.828652	67.9564 / 0.00175637 / 193.259	4695
CACHE	0.830038	64.6417 / 0.00172859 / 191.703	0
SAF	0.837664	62.1525 / 0.00172887 / 190.896	0
PDRS	0.842982	61.0969 / 0.00172652 / 191.279	246454
DCG	0.848559	59.0611 / 0.00172883 / 189.270	14510
PCS	0.868749	50.2859 / 0.00172885 / 188.550	9846
OPTM	0.878677	41.2282 / 0.00172874 / 188.428	4721

6 Performance Evaluation in Disconnected MANETs with the Community-Based Mobility Model

In order to evaluate the performance of PCS in disconnected MANETs, we conducted event-driven experiments with the MIT Reality project [35] trace and the Haggle project [36] trace. The MIT Reality trace lasts about 2.56 million seconds (Ms), while the Haggle project trace lasts about 0.34 Ms. Both traces represent typical disconnected MANET scenarios. We used the Static Waiting routing protocol [40] in this test.

We evaluated the performance of PCS in comparison with DCG [10], CACHE-DTN [14], OPTM, and Random. CACHE-DTN is a caching algorithm for DTNs. It caches each file in the central node of each network center location (NCL). If a central node is full, its replicas are stored in its neighbor nodes according to their popularity. A higher popular replica is stored closer to the central node. The experiment settings and measurement metrics are the same as in Section 5 unless otherwise specified below. The total number of queries was set to $6000 * R_p$, and R_p is the query rate and was varied in the range of [2, 6]. In the experiment with the Haggle trace and the MIT Reality trace, all queries were generated evenly in the time period of [0.3Ms, 2.3Ms] and [0.05Ms, 0.25Ms], and the TTL of each query was set to 0.3Ms and 0.04Ms, respectively. We again adopted the 95% confidence interval when handling experimental data.

6.1 Hit Rate

Figure 5(a) and Figure 6(a) plot the hit rates of the five methods with the Haggle trace and the MIT Reality trace, respectively. We see that in both scenarios, the hit rates follow OPTM>PCS>CACHE-DTN>DCG>Random. OPTM and PCS achieve higher hit rate than other methods because they follow the deduced OFRR. However, since PCS realizes OFRR in a distributed way, it presents slightly inferior performance compared to OPTM. CACHE-DTN considers the intermittent connection properties of disconnected MANETs and replicates each file to every NCL, leading to high date accessibility, though not optimal. DCG only considers temporary connected group for data replication, which is not stable in disconnected MANETs. Therefore, it has a low hit rate. Random assigns resources to files randomly, which means it cannot create more replicas for popular files, leading to the lowest hit rate. Such a result proves the effectiveness of the proposed PCS on improving the overall file availability and the correctness of our derived OFRR for disconnected MANETs.

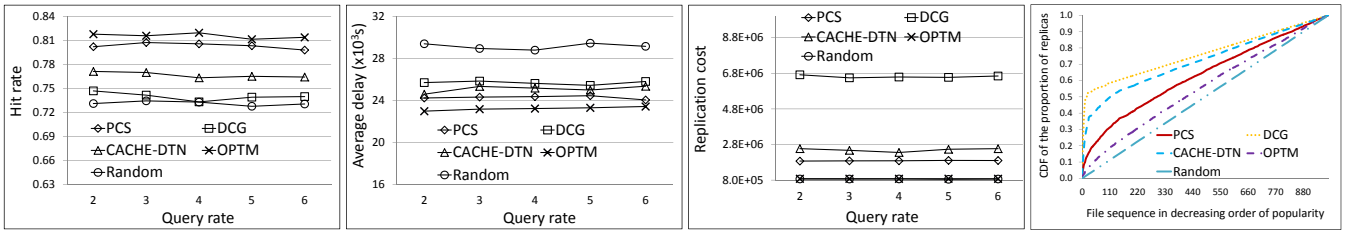
We also see that the hit rates of different methods fluctuate slightly when the query rate increases. This is because the hit rate is not affected by the query rate. Even when the number of query increases, the file availability remains on the same level and leads to similar hit rates, as shown in the two figures.

6.2 Average Delay

Figure 5(b) and Figure 6(b) demonstrate the average delays of the five methods with the Haggle trace and the MIT Reality trace, respectively. We find that with both traces, the average delays follow OPTM<PCS<CACHE-DTN<DCG<Random, which is in reverse order of the relationship between the five methods on hit rate as shown in Figure 5(a) and Figure 6(a). This is because the average delay is reversely related to the overall data availability. As explained in above section, OPTM and PCS have high data availability since they follow OFRR, CACHE-DTN presents higher data availability than DCG because CACHE-DTN distributes every file to different NCLs while DCG only shares data among frequently encounterd neighbor nodes, and popular files in Random only receive equal amount of resources for replicas. Such results further validate the proposed OFRR and PCS in disconnected MANETs.

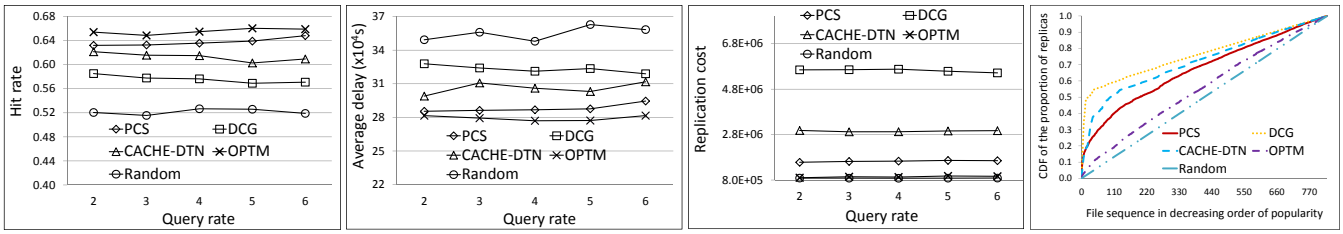
We again find that the average delays of different methods vary slightly when the query rate increases. It is caused by the same reason as explained in above section.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2014.2308211, IEEE Transactions on Computers



(a) Hit rate. (b) Average delay. (c) Replication cost. (d) CDF of allocated resources.

Fig. 5: Performance of the file replication protocols with the Haggle trace.



(a) Hit rate. (b) Average delay. (c) Replication cost. (d) CDF of allocated resources.

Fig. 6: Performance of the file replication protocols with the MIT Reality trace.

6.3 Replication Cost

Figure 5(c) and Figure 6(c) show the replication costs of the five methods with the Haggle trace and the MIT Reality trace, respectively. OPTM and Random have the lowest replication cost while the costs of the other three methods follow PCS<CACHE-DTN<DCG. In OPTM and Random, nodes only need to contact the central server for replica list, leading to the lowest cost. DCG generates the highest replication cost since group members need to exchange a large amount of information to remove duplicate replicas. CACHE-DTN forwards each file to every NCL, leading to moderate replication cost. In PCS, a node tries at most K times to create a replica for each of its files, producing much lower replication cost than CACHE-DTN and DCG. Such a result demonstrates the high energy-efficiency of PCS. Combining all above results, we conclude that PCS has the highest overall file availability and efficiency compared to existing methods, and OFRR is effective in guiding optimal file replication in disconnected MANETs.

6.4 Replica Distribution

Figures 5(d) and 6(d) show the CDF of the proportion of resources allocated to replicas in each protocol in the tests with the Haggle trace and the MIT Reality trace, respectively. We see in both figures, PCS present very close similarity to OPTM and the other two follow CACHE-DTN>PCS. Random also present close similarity on the CDF curve to OPTM. However, the difference between PCS and Random is that PCS assigns priority for popular files while Random gives even priority to all files. Since popular files are queried more frequently, Random still leads to a low performance. For other three methods that favor popular files, we find that the closer similarity with OPTM a protocol has, the better overall performance it has. Such a result also matches what we observed in the test in connected MANETs. This proves the correctness of our theoretical analysis and the resultant OFRR rule for disconnected MANETs.

7 Additional Test

In this section, we conducted additional experiments show the influence of K in PCS on file availability. Specifically, we varied K from 1 to 4 to show its influence on the performance of PCS with both mobility models. We followed the same setting as in Section 5 and 6. In the test with the community-based mobility model, we set the query rate to a median value of 4. The test results are shown in Table 5. In the table, we use Dt, Hg and MIT to denotes the Dartmouth trace, the Haggle project trace and the MIT Reality trace, respectively.

We see that when K increases, both the performance and the replication cost of PCS increase. This is because when K increases, each file has more chances to compete with others to create replicas and gets the portion of resources based on the OFRR rule. Therefore, resources are allocated more strictly following the OFRR rule, leading to better performance. On the other hand, more trials also means more replication costs. Therefore, a balance on the performance and replication cost should be considered when deciding K, which is an interesting topic for our future work.

TABLE 5: Influences of Different Ks.

	Hit rate			Ave. delay (x$10^3 s$)			Rep. Cost (x10^4)		
K	Dt	Hg	MIT	Dt	Hg	MIT	Dt	Hg	MIT
1	0.861	0.799	0.618	0.0515	26.9	294.9	0.4	70	59
2	0.866	0.805	0.628	0.0510	25.8	288.4	0.7	124	113
3	0.869	0.814	0.635	0.0502	24.3	286.8	1.0	188	163
4	0.871	0.821	0.644	0.0497	23.5	284.2	1.2	237	215

8 Conclusion

In this paper, we investigated the problem of how to allocate limited resources for file replication for the purpose of global optimal file searching efficiency in MANETs. We first theoretically analyzed the influence of replica distribution on the average querying delay under constrained available resource under two mobility models, and derived an optimal replication rule to allocate the limited resource to file replicas in order to minimize the average querying delay. Unlike previous

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2014.2308211, IEEE Transactions on Computers

protocols that only consider storage space as resource, we also consider file holder's ability to meet nodes as available resource since it also affects the average querying delay. This new concept enhances the correctness of the deduced rule and the effectiveness of the accordingly developed replication protocol. Finally, we designed the Priority Competition and Split replication protocol (PCS) that realizes the proposed optimal replication rule in a fully distributed manner. Extensive experiments on both real-world GENI testbed, NS-2, and event-driven simulator with real trace and synthesized mobility confirm both the correctness of our theoretical analysis and the effectiveness of PCS in MANETs. In this study, we focus on a static set of files in the network. In our future work, we will theoretically analyze a more complex environment including file dynamics (file addition and deletion, file timeout) and dynamic node querying pattern.

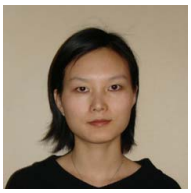
ACKNOWLEDGMENT

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] "Qik," http://qik.com/.
[2] "Flixwagon," http://www.flixwagon.com/.
[3] C. Palazzi and A. Bujari, "A delay/disruption tolerant solution for mobile to mobile file sharing." in *Proc. of IFIP/IEEE Wireless Days*, 2010.
[4] Y. Tseng, S. Ni, and E. Shih, "Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc network," in *Proc. of ICDCS*, 2001, pp. 481–488.
[5] B. Chiara, C. Marco, and et al., "Hibop: A history based routing protocol for opportunistic networks," in *Proc. of WoWMoM*, 2007.
[6] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic routing in intermittently connected networks," *MC2R*, vol. 7, no. 3, pp. 19–20, 2003.
[7] F. Li and J. Wu, "MOPS: Providing content-based service in disruption-tolerant networks," in *Proc. of ICDCS*, 2009.
[8] S. Moussaoui, M. Guerroumi, and N. Badache, "Data replication in mobile ad hoc networks," in *Proc. of MSN*, 2006, pp. 685–697.
[9] L. Yin and G. Cao, "Supporting cooperative caching in ad hoc networks," *TMC*, vol. 5, no. 1, pp. 77–89, 2006.
[10] T. Hara and S. K. Madria, "Data replication for improving data accessibility in ad hoc networks," *TMC*, vol. 5, no. 11, pp. 1515–1532, 2006.
[11] J. Zheng, J. Su, K. Yang, and Y. Wang, "Stable neighbor based adaptive replica allocation in mobile ad hoc networks," in *Proc. of ICCS*, 2004.
[12] H. Duong and I. Demeure, "Proactive data replication semantic information within mobility groups in MANET," in *Proc. of Mobilware*, 2009.
[13] Y. Huang, Y. Gao, and et al., "Optimizing file retrieval in delay-tolerant content distribution community," in *Proc. of ICDCS*, 2009.
[14] W. Gao, G. Cao, A. Iyengar, and M. Srivatsa, "Supporting cooperative caching in disruption tolerant networks." in *Proc. of ICDCS*, 2011.
[15] J. Reich and A. Chaintreau, "The age of impatience: optimal replication schemes for opportunistic networks." in *Proc. of CoNEXT*, 2009.
[16] S. Ioannidis, L. Massoulie, and A. Chaintreau, "Distributed caching over heterogeneous mobile networks." in *Proc. of SIGMETRICS*, 2010.
[17] M. J. Pitkanen and J. Ott, "Redundancy and distributed caching in mobile DTNs," in *Proc. of MobiArch*, 2007.
[18] X. Zhuo, Q. Li, W. Gao, G. Cao, and Y. Dai, "Contact duration aware data replication in delay tolerant networks." in *Proc. of ICNP*, 2011.
[19] X. Zhuo, Q. Li, G. Cao, Y. Dai, B. K. Szymanski, and T. L. Porta, "Social-based cooperative caching in DTNs: A contact duration aware approach." in *Proc. of MASS*, 2011.
[20] Z. Li and H. Shen, "Sedum: Exploiting social networks in utility-based distributed routing for DTNs," *TC*, 2012.
[21] V. Gianuzzi, "Data replication effectiveness in mobile ad-hoc networks," in *Proc. of PE-WASUN*, 2004, pp. 17–22.
[22] S. Chessa and P. Maestrini, "Dependable and secure data storage and retrieval in mobile wireless networks," in *Proc. of DSN*, 2003.
[23] X. Chen, "Data replication approaches for ad hoc wireless networks satisfying time constraints," *IJPEDS*, vol. 22, no. 3, pp. 149–161, 2007.
[24] J. Broch, D. A. Maltz, D. B. Johnson, Y. Hu, and J. G. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," in *Proc. of MOBICOM*, 1998, pp. 85–97.
[25] M. Musolesi and C. Mascolo, "Designing mobility models based on social network theory," *MCCR*, vol. 11, pp. 59–70, 2007.
[26] Http://web.informatik.uni-bonn.de/IV/BoMoNet/BonnMotion.htm.
[27] 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.
[28] M. Musolesi and C. Mascolo, "Car: Context-aware adaptive routing for delay-tolerant mobile networks." *TMC*, 2009.
[29] H. Cai and D. Y. Eun, "Crossing over the bounded domain: from exponential to power-law inter-meeting time in MANET." in *Proc. of MOBICOM*, 2007.
[30] R. Groenevelt, P. Nain, and G. Koole, "The message delay in mobile ad hoc networks." *Perform. Eval.*, vol. 62, pp. 210–228, 2005.
[31] G. Sharma, R. Mazumdar, and N. B. Shroff, "Delay and capacity trade-offs in mobile ad hoc networks: A global perspective." in *Proc. of INFOCOM*, 2006.
[32] L. Kleinrock, *Queueing Systems, Volume II: Coputer Applications*. John Wiley & Sons, 1976.
[33] J. Kangasharju, K. W. Ross, and D. A. Turner, "Optimizing file availability in peer-to-peer content distribution," in *Proc. of INFOCOM*, 2007.
[34] R. S. Gray, D. Kotz, C. Newport, N. Dubrovsky, A. Fiske, J. Liu, C. Masone, S. McGrath, and Y. Yuan, "CRAWDAD data set dartmouth/outdoor (v. 2006-11-06)," http://crawdad.cs.dartmouth.edu/dartmouth/outdoor.
[35] N. Eagle, A. Pentland, and D. Lazer, "Inferring social network structure using mobile phone data," *PNAS*, vol. 106, no. 36, 2009.
[36] A. Chaintreau, P. Hui, J. Scott, R. Gass, J. Crowcroft, and C. Diot, "Impact of human mobility on opportunistic forwarding algorithms," in *Proc. of INFOCOM*, 2006.
[37] "GENI project," http://www.geni.net/.
[38] "Orbit," http://www.orbit-lab.org/.
[39] "The Network Simulator ns-2," http://www.isi.edu/nsnam/ns/.
[40] T. Spyropoulos, K. Psounis, and C. Raghavendra, "Efficient routing in intermittently connected mobile networks: The single-copy case," *ACM/IEEE Transactions on Networking*, 2007.
[41] M. Lu and J. Wu, "Opportunistic routing algebra and its application," in *Proc. of INFOCOM*, 2009.
[42] T. Hara, "Effective replica allocation in ad hoc networks for improving data accessibility," in *Proc. of INFOCOM*, 2001.
[43] Z. Li and H. Shen, "Analysis of cooperation incentive strategies in mobile ad hoc networks," *TMC*, 2012.
[44] B. Chen and M. C. Chan, "MobiCent: a credit-based incentive system for disruption tolerant network." in *Proc. of INFOCOM*, 2010.

Kang Chen Kang Chen received the BS degree in Electronics and Information Engineering from Huazhong University of Science and Technology, China in 2005, and the MS in Communication and Information Systems from the Graduate University of Chinese Academy of Sciences, China in 2008. He is currently a Ph.D student in the Department of Electrical and Computer Engineering at Clemson University. His research interests include mobile ad hoc networks and delay tolerant networks.



Haiying Shen Haiying Shen received the BS degree in Computer Science and Engineering from Tongji University, China in 2000, and the MS and Ph.D. degrees in Computer Engineering from Wayne State University in 2004 and 2006, respectively. She is currently an Assistant Professor in the Department of Electrical and Computer Engineering at Clemson University. Her research interests include distributed computer systems and computer networks, with an emphasis on P2P and content delivery networks, mobile computing, wireless sensor networks, and cloud computing. She is a Microsoft Faculty Fellow of 2010, a senior member of the IEEE and a member of the ACM.