Global Optimization of File Availability Through Replication for Efficient File Sharing in MANETs

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

***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 MANETs 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 sharing with minimum average querying delay. Specifically, current file replication protocols in MANETs 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. A node having a higher meeting frequency with others provides higher availability to its files. 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 that follows the rule. The trace-driven experiments on both the real-world GENI testbed and NS-2 show that our protocol can achieve shorter average querying delay at lower cost than current replication protocols, which justifies the correctness of our theoretical analysis and the effectiveness of the proposed protocol.**

I. Introduction

File sharing applications, like Qik [1] and Flixwagon [2], in mobile ad hoc networks (MANETs) have attracted more and more attention in recent years. Among many feasible techniques, peer-to-peer (P2P) file sharing between nodes in MANETs is promising since it avoids the problem of overloading servers (i.e., base stations) in current client-server based file sharing in the infrastructure wireless networks. Without the dependency on central servers, nodes can freely and unobtrusively access and share files. For example, tourists can share their travel experiences with other tourists or convey emergency information through their digital devices directly even when no base station is available in remote areas.

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 with others only when they are within the communication range. Broadcasting can quickly discover files, but it generates the broadcast storm problem [3] with high energy consumption. Probabilistic routing and file discovery protocols [4]–[6] avoid broadcasting by forwarding a query to a node with higher probability of meeting the destination. But the opportunistic encountering of nodes in MANETs may lead to large file discovery delay.

File replication is an effective way to enhance file availability and reduce the 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 resource. 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 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?*

Recently, a number of file replication protocols have been proposed for MANETs [7]–[11]. In these protocols, each individual node replicates files it frequently queries [7]–[9], or a group of nodes create one replica for each file they frequently query [9]–[11]. In the former, neighboring nodes easily create redundant replicas in the system and waste resource. Though group based file replication solves the problems by sharing replicas to be shared among neighbors, neighboring nodes may separate from each other due to node mobility. 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 resource for replicas, but neglect that a node's ability (i.e., 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 file availability.

In this paper, we introduce a new concept of resource for file replication, which considers both node storage and node meeting ability. The meeting ability of a node is measured by the average number nodes it can meet in a unit time. We theoretically study the influence of resource allocation on the average querying delay and derive an optimal file replication rule that decides the amount of resource for 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 that can approximately realize the rule and achieve 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.

II. Related Work

The topic of file replication for efficient file sharing applications in MANETs has been studied recently. In the proposed file replication protocols [9]–[11], individual or a group of nodes decide the list of files to replicate according to file visiting frequency. Hara [9] 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* [10] proposed to let each node collect access statistics from neighbors to decide the creation or relinquishment of a replica. Duong and Demeure [11] 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 [8] 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 [12] 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 [13] is employed to segment files. Chen [14] 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. Khan *et al.* [15] exploited game theory and derived a payment scheme to solve the negative effects brought about by selfish mobile servers. Moussaoui *et al.* [7] 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 (i.e., the number of hops a request for the file has traveled) to make the replication decision.

III. Theoretical Analysis of Globally Optimal File Replication

A. Node Movement Model

We define the meeting ability of a node in MANETs as the average number of nodes it can meet in a unit time. Different nodes have different meeting abilities due to various reasons (e.g., velocity and active level). Node movement pattern also influences the node meeting ability. We consider a MANET scenario in which the node movement pattern follows the modified random waypoint model (RWP) [16], which has been used in several MANET replication protocol [9], [10], [12]. In RWP, nodes repeatedly select a randomly destination and move to it at a random speed straightly. So the each node has roughly similar meeting ability. We let each node has a fixed speed, which is randomly obtained from a range, in the RWP model. So from the perspectively of one node, it has the same probability of meeting any other node, which means no local gathering exists in the system. Although nodes move randomly in the modified RWP model, they may have different meeting abilities due to different velocities. For example, nodes with faster speed can meet other nodes more frequently (i.e., with short average separation period). We regard our work based on the modified RWP as a fundamental work and will briefly introduce how to adapt our fundamental analysis to other node movement models in the end of this section.

B. Theoretical Analysis

TABLE I: 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 replicas for file j
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 (average $\overline{T_j}$)
$\mathcal{R}$	Total amount of resource in the system
P_j	Priority Value of file j, $P_j = \sqrt{q_j / b_j}$

We first theoretically analyze the influence of the file replica distribution on the overall query efficiency. We assume there is no update on files in the system. Please refer to Table I for the meanings of notations used in our analysis. If a node is able to meet more nodes during a time period unit, it means the node 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 ability to meet nodes (i.e., V_i). Then

$$m_i = \frac{V_i}{\overline{V}N} \tag{1}$$

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

We use vector $(V_{j0}, V_{j1}, \ldots, V_{jk}, \ldots)$ to denote the meeting abilities of a group of nodes holding file j or its replica. 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,

$$p_j = \sum_{i=1}^{N} m_i X_{ij} = \sum_{i=1}^{N} \frac{V_i}{\overline{V}N} X_{ij} = \sum_{k=1}^{n_j} \frac{V_{jk}}{\overline{V}N} \tag{2}$$

where X_{ij} is a zero-one variable that denotes whether node i contains file j or its replica and n_j is the number of file j (including replicas) in the system.

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 node's meeting ability and storage in measuring a node's resource. When a replica is created in a node, ts probability of being met by others is decided by the node's meeting ability. So we regard it consumes both storage resource and 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:

$$\mathcal{R} = \sum_{i=1}^{N} S_i V_i \tag{3}$$

Thus, the total resource allocated to file j is:

$$R_j = b_j \sum_{k=1}^{n_j} V_{jk} \tag{4}$$

where b_j is the size of file j. Based on Equation (4), Equation (2) can be represented as

$$p_j = \frac{b_j \sum\limits_{k=1}^{n_j} V_{jk}}{b_j \overline{V} N} = \frac{R_j}{b_j \overline{V} N} \tag{5}$$

Thus, the probability of encountering file j after k ($k = 1, 2, 3, \cdots$) intervals 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

$$\overline{T_j} = \sum_{k=1}^{\infty} k(1 - p_j)^{k-1} p_j = \frac{1}{p_j} = \frac{b_j \overline{V} N}{R_j} \tag{6}$$

We use q_j to denote the probability of originating a request for file j. Then, the average number of intervals needed to satisfy a request is

$$\overline{T} = \sum_{j=1}^{F} q_j \overline{T_j} = \sum_{j=1}^{F} q_j \frac{b_j \overline{V} N}{R_j} = \overline{V} N \sum_{j=1}^{F} \frac{q_j b_j}{R_j} \tag{7}$$

We aim to minimize the global file querying delay (i.e., $\overline{T}$) by file replication. According to Equation (7), $\overline{T}$ 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 (3) and (7), the problem of optimal resource allocation is expressed by

$$\min(\overline{T}) = \min\{\sum_{j=1}^{F} \frac{q_j b_j}{R_j}\} = \min\{\sum_{j=1}^{F} \frac{B_j}{R_j}\} \tag{8}$$

subject to:

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

Equation (7) also indicates that each R_j should be as large as possible in order to minimize $\overline{T}$. Therefore, we let the sum of all R_j equals $\mathcal{R}$.

$$\sum_{j=1}^{F} R_j = \mathcal{R} \tag{9}$$

By applying Formula (9), Formula (8) is changed to

$$\min(\overline{T}) = \min\{\frac{B_1}{R_1} + \frac{B_2}{R_2} + \cdots + \frac{B_F}{\mathcal{R} - (R_1 + R_2 + \cdots + R_{F-1})}\} \tag{10}$$

Next, we try to find the value of R_j $(1 \leq j \leq F-1)$ that satisfies Formula (10). Specifically, we differentiate $\overline{T}$ 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

$$\frac{B_1}{R_1^2} - \frac{B_F}{\{\mathcal{R} - (R_1 + R_2 + \cdots + R_{F-1})\}^2} = 0 \tag{11}$$

$$\cdots\cdots\cdots$$

$$\frac{B_{F-1}}{R_{F-1}^2} - \frac{B_F}{\{\mathcal{R} - (R_1 + R_2 + \cdots + R_{F-1})\}^2} = 0 \tag{12}$$

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

$$\frac{B_1}{R_1^2} = \frac{B_2}{R_2^2} = \frac{B_3}{R_3^2} = \cdots = \frac{B_{F-1}}{R_{F-1}^2} = \frac{B_F}{R_F^2} \tag{13}$$

According to Equation (9) and Equation (13), we can see that the optimal allocation is

$$R_j = \frac{\sqrt{B_j}}{\sum\limits_{k=1}^{F} \sqrt{B_k}} \mathcal{R} \ (j = 1, 2, 3, \cdots, F) \tag{14}$$

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:

$$R_j \propto \sqrt{B_j} \Rightarrow b_j \sum_{k=1}^{n_j} V_{jk} \propto \sqrt{b_j q_j} \tag{15}$$

That is

$$\sum_{k=1}^{n_j} V_{jk} \propto \sqrt{\frac{q_j}{b_j}} \Rightarrow \sum_{k=1}^{n_j} V_{jk} \propto P_j \tag{16}$$

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

By converting Formula (15) to Formula (16), we convert the double-factor consideration (i.e., both storage and meeting

ability) in the resource allocation to the single-factor consideration (i.e., meeting ability). This is reasonable since once a replica is created, it naturally takes the storage resource and meeting ability resource at the same time.

Based on Formula (16), 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.

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 $P_j = \sqrt{q_j/b_j}$.*

It is interesting to find that OFRR matches the "square root assignment rule" derived by Kleinrock [17] for the link capacity assignment in wireless communication to maximize the network efficiency. It also matches the findings in [18] that when file servers may become unavailable due to node dynamism, the wired P2P content distribution systems can achieve the maximum file hit rate when available storage is allocated in proportional to a constant value plus $ln(q_j/b_j)$ for each file.

C. Extension to Other Node Movement Models

Although above results are obtained based on the modified RWP model in which nodes move randomly and independently, the analysis process can be generalized and adapted to other node movement models. In any kind of node movement model, we first need to figure out the probability that the newly met node is node i (i.e., m_i in Formula (1)), which reflects the meeting ability resource of node i. Then, following similar procedures from Formula (2) to Formula (6), the average number of time intervals needed to meet a specific file, say file j, can be represented as:

$$\overline{T_j'} = \frac{1}{p_j'} = \frac{1}{\sum_{i=1}^{N} m_i' X_{ij}} \tag{17}$$

where p_j' and m_i' represent p_j and m_i under the new movement model. Then, similar to Formula (7), the average number of intervals needed to satisfy a request is

$$\overline{T'} = \sum_{j=1}^{F} q_j \overline{T_j'} = \sum_{j=1}^{F} \frac{q_j}{\sum_{i=1}^{N} m_i' X_{ij}}, \tag{18}$$

where T' represents T under the new movement model. With Formula (18), we can formulate the global optimization problem as minimizing $\overline{T'}$ under limited resource and deduce the optimal resource allocation rule.

However, the calculation of m_i' may be complex and makes the minimization problem non-trivial in complex mobility models. For example, in the recently proposed community based mobility model [19], nodes gather together according to their social relationships. Therefore, nodes in the same community meet with each other more often than with others. Then, a node's probability of meeting node i in the next encountering (m_i') differs from node to node since we need to consider whether the node and node i belong to the same social community. Further exploration of the optimal resource allocation rule under other models is beyond the scope of this paper, and we leave it as our future work.

IV. Distributed File Replication Protocol

A. Challenges to Achieve the Optimal File Replication Rule

Challenge 1: resource allocation without a central server. OFRR and Formula (15) show that in order to realize the globally optimal querying delay, each node needs to know the popularity (q_j) and size (b_j) of all files and the total available resource to decide the portion of resource for each of its files for replica creation. Specifically, suppose there are F files in the system with $b_1 q_1 \cdots b_F q_F$ and total resource $\mathcal{R}$, the resource allocated to file j (R_j) is

$$R_j = \mathcal{R} \times \sqrt{b_j q_j} / \sum_{k=1}^{F} \sqrt{b_k q_k} \tag{19}$$

So, an intuitive way to attain this goal is to setup a central server to collect and distribute required information. However, the nature of the distributed network, node mobility and transmission range constraint become obstacles of building such a central service. 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 in a timely fashion. Thus, a severe challenge is how to enable a node to distributively figure out the proper portion of resource for each of its files without a central server.

Even though each node knows $\sqrt{b_j q_j} / \sum_{k=1}^{F} \sqrt{b_k q_k}$ of each of its files, because of the time-varying available total resource in the system ($\mathcal{R}$) due to node joins and departures and the total number of files (F) due to file deletions and creations, it is 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 is 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. If f_2 is deleted, the amount for f_1 then should be 60. Further, the time-varying file popularity and subsequent change of $b_j q_j$ 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 for the replicas.

Solution to Challenge 1: resource competition. Formula (16) shows that the sum of the meeting ability of replica nodes of each file, $\sum_{k=1}^{n_F} V_{Fk}$, is proportional to the file's priority value P. This also means that the ratio of each file's P to its $\sum_{k=1}^{n_F} V_{Fk}$ has the same value. Therefore, OFRR finally achieves

$$P_1 / \sum_{k=1}^{n_1} V_{1k} = P_2 / \sum_{k=1}^{n_2} V_{2k} \cdots = P_F / \sum_{k=1}^{n_F} V_{Fk} \tag{20}$$

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. Hence, the competition gradually lets each file receive its deserved portion of resource based on OFRR. Therefore, 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 MANETs, all available resource is scattered among different nodes moving around in the network. This poses three problems. First, file owners have limited probability to gather to conduct the resource competition. Second, after a file is replicated to a number of nodes, it is difficult for its owner 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 and a node's capability are both limited, a single file owner cannot distribute replicas efficiently and quickly. We propose an optimized way to solve these problems by regarding a file and its newly created replica as two different files, which participate in further competition independently. However, this brings another challenge: how can the replicas ensure that the $\sum V_{jk}$ is proportional to its P in the resource competition?

Solution to Challenge 2: distributive competition on selective resources. As mentioned, enabling replica nodes to replicate files makes it difficult to keep P proportional to $\sum V_{jk}$. We indirectly resolve this problem by keeping the average V of the replica nodes of a file close to $\overline{V}$ via selectively choosing replica nodes. Formula (16) can then be re-expressed as

$$n_j * \overline{V} \propto \sqrt{\frac{q_j}{b_j}} \Rightarrow n_j \propto \sqrt{\frac{q_j}{b_j}} \Rightarrow n_j \propto P_j \qquad (21)$$

In such a case, when the number of replicas of each file is proportional to its $P = \sqrt{q_j/b_j}$, OFRR is also satisfied. Accordingly, we deliberately select nodes to create replicas so that the average meeting ability of replica nodes equals to $\overline{V}$. Thus, each node competes for resource for its file j only with its P. To allow nodes to replicate files in a distributed manner, upon winning a competition for a file, a node splits the file's P evenly between the file and the replica file. Each file keeps replicating until it fails a competition. Thus, for a file with k replicas, each of its replica's P generally equals P_j/k. The sum of these replicas' Ps equals P_j (P_j is the P of the original file j). In other words, the number of splits each file j has experienced (i.e., the number of replicas of each file) is proportional to its P_j, causing the number of replicas of each file is proportional to the sum of meeting ability of its replica nodes. As a result, Formula (16) is satisfied.

B. Design of the File Replication Protocol

According to the analysis above, we propose the Priority Competition and Split file replication protocol (PCS). We first introduce how a node retrieves the parameters needed in PCS followed by a detailed description of PCS.

Each node needs to know the average meeting ability of all nodes ($\overline{V}$). As nodes move randomly and independently in the network, we can assume that the set of nodes encountered by each node is randomly chosen from the set of all nodes in the network. Then, the average meeting ability of all encountered nodes of a node can generally represent the average meeting ability of all nodes in the network. As a node meets more and more nodes in the system, its calculated $\overline{V}$ converges to the real value. In PCS, each node i periodically calculates its meeting ability (V_i) measured by the frequency it meets other nodes, and exchanges its V_i with its neighbors by piggybacking the information into beacon messages. Each node also periodically calculates the popularity of each of its files (q_j) measured by the number of its received requests for the file in a unit of time period, and calculates the file's $P_j = \sqrt{q_j/b_j}$.

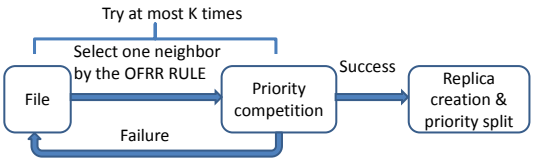


Fig. 1: Replica distribution process.

With above information, a node can order all of its files in descending order of their Ps and creates replicas for the files in the top-down manner. In the Solution to Challenge 2, nodes replicate files in a distributed manner, and each replicating node tries to ensure that the average meeting ability of replica nodes of a file equals to $\overline{V}$. That is, $\overline{V_{n'_j}} \approx \overline{V}$, where n'_j denotes the total number of replicas of file j created by a node and $\overline{V_{n'_j}}$ denotes the average meeting ability of the 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 replicating node increases n'_j by 1 and also updates $\overline{V_{n'_j}}$ using the V of the new replica node. Since the computation only involves simple operations and only one value needs piggybacking, PCS is suitable to the energy-constrained MANETs.

Figure 1 demonstrates the process of the replication of a file in PCS. For example, suppose node i needs to replicate file j. It keeps trying to replicate file j on nodes it encounters until one replica is successfully created or K attempts have been made. To choose a neighbor to replicate file j, node i first checks the meeting abilities of its neighbors. Recall that a replicating node should keep the average meeting ability of the replica nodes for each of its files at $\overline{V}$. Node i finds the neighbor that does not contain file j and has V_k that makes $(n'_j\overline{V_{n'_j}} + V_k)/(n'_j + 1)$ the closest to $\overline{V}$.

If the neighbor's available storage space is larger than the size of file j (S_j), it creates a replica for file j. Otherwise, a competition is launched among file j's replica and other replicas already residing in the neighbor based on their Ps. The priority value of the new replica is set to half of the original file's P. The competition is conducted as a drawing to select one or more replicas to be deleted. According to OFRR, each replica has a probability of being selected to remove, which is inversely proportional to its P. Assume there are d replicas in competition. Each replica is responsible for a range in $[0, \sum_{k=1}^{d} 1/P_k]$ and the length of the range equals its $1/P$. The neighbor randomly chooses a number in $[0, \sum_{k=1}^{d} 1/P_k]$, and the replica whose range owns the number is chosen. If the size of the chosen replica is less than S_j, the neighbor repeats

the same process until available storage is no less than S_j.

If file j is among the selected files, which means file j fails the competition, only file j's replica is deleted. Otherwise, all selected files are removed. Also, if file j fails, node i will launch another attempt for file j until the maximum number of attempts (K) is reached. Each attempt starts with identifying the neighbor to replicate file. The setting of K attempts is to ensure that each file can compete with a subset of sufficient 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 the next time period. Finally, all available resource in the system is allocated to replicas according to their Ps and OFRR is realized.

According to the Solution to Challenge 2, we regard a 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. Without the splitting strategy, files with large Ps will receive more and more resource and starve files with small Ps. As the popularity of files, their Ps and available system resource change as time goes on, each node periodically executes PCS.

C. Analysis of the Effectiveness of PCS

In this section, we prove the effectiveness of PCS through analysis. We refer to the process a node tries to copy a file to its neighbors as one round of replication distribution.

When a replica is created for a file with P, the two copies will replicate files with priority $P/2$ in the next round. After the second round, suppose there is no update of the priority value, the four copies of the file will further replicate files with priority $P/4$, and so on. So, the sum of the Ps of the replicas of each original file is P plus the increase of its priority value, we can regard the replicas of a file as a whole and they compete 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:

$$Ps_j \propto P_j \tag{22}$$

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

$$n_j = MPs_j \Rightarrow n_j \propto P_j \tag{23}$$

Therefore, we can conclude that the PCS can realize Equation (21), in which the number of replicas of each file is proportional to its P, thereby realizing OFRR.

V. Performance Evaluation

We conducted experiments on the GENI Orbit testbed [20], [21], which is a MANET testbed consisting of 400 nodes equipped with wireless cards, and the NS-2 [22] simulator. We used a real-world MANET trace [23] to drive nodes mobility in both experiments. The real trace [23] was obtained through an outdoor project in Dartmouth University and it provides position records of 35 laptop nodes moving randomly and independently across different sections of an open field. 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 as previously indicated. In order to validate the adaptivity of PCS, we used two routing protocols in the experiments. We first used the Static Wait routing protocol [24], in which each query stays on the source node until meeting the destination, in the GENI experiment. We then used the PROPHET probabilistic routing protocol [5], in which a node routes requests to the neighbor with the highest meeting ability, in the simulation. We set a larger TTL for Static Wait since it needs more time to find a file holder than PROPHET.

We evaluated the performance of PCS in comparison with SAF [9], DCG [9], PDRS [11] and CACHE [8]. The details of these protocols can be found in Section II. We also included the performance of the centralized protocol that replicates files according to OFRR, which is denoted as OPTM. OPTM shows the best possible performance of OFRR.

Table II shows the parameters used in experiments, unless otherwise specified. The parameters are determined by referring to the settings in [8], [25] and the real trace. According to the works in [8], [26], we determined the file size and storage space. As the work in [18], the popularity of files 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 set it 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 $K = 3$. Each test was run by 5 times and the average value for each metric is presented. 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 time 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 and shows whether a replication protocol can achieve OFRR in resource allocation.

TABLE II: 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	$1-10$	$1-10$
Number of files in each node	10	10
Storage space for replicas	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	$3000s$ / $3000s$	$1000s$

A. Performance in the Trace-Driven GENI experiments

1) Hit Rate and Average Delay: Table III shows the results of each protocol in the trace-driven experiments on GENI. We see that the hit rates in different replication protocols follow CACHE<SAF<PDRS<DCG<PCS<OPTM. This is because in protocols with longer querying delay, more requests are dropped due to TTL, leading to lower hit rates. The result is supported by the fact that the results on average delay present a reverse order, as shown in the second column of the table. We see that OPTM and PCS lead to lower delays compared with others. This is attributed to the guidance of OFRR, which aims to minimize the average querying delay. Although nodes with high meeting ability may move at a long time scale, they can meet more nodes in a unit time and thereby deliver queries to their destinations more quickly on an average base. Therefore, by considering both storage and meeting ability as resource to enhance file availability, OPTM and PCS optimally allocate all available resource to different files for replication to enhance global file availability and overall file searching efficiency.

On the contrary, other protocols only replicate files locally, consuming resource with redundant replicas and failing to achieve high file availability under node mobility. PCS generates around 20% higher average delay than OPTM. This is because OPTM has the knowledge of all information of each node needed in OFRR beforehand, while PCS has to distribute replicas in a fully distributed manner. The closer performance of PCS to OPTM than others demonstrates the effectiveness of PCS in realizing OFRR in a distributed manner.

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

Protocol	Hit rate	Average / 1% / 99% delay (s)	Replication cost
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	0

We see that the average delays of other four protocols follow CACHE>SAF>PDRS>DCG. 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 the highest delay. Other protocols can fully utilize storage space. In SAF, each node replicates its frequently queried files until its memory is filled up. Since file popularity follows the Zipf distribution, almost all resource is allocated to popular files, leading to large delay for requests querying for unpopular files. Therefore, SAF cannot achieve global optimization of all file queries. In PDRS, a node replicates files interested by its neighbors that have less storage resource than itself, making replicas be shared among neighbors. However, as the sharing of replicas is not in the whole group, PDRS only renders a slightly lower delay than 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 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, leading to very short delay. 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.

2) Replication Cost: From the table, we find that the replication costs of different protocols follow PDRS>DCG >PCS>SAF=OPTM=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 very low replication cost because each node only tries at most K times to create a new replica for each file it holds. SAF, CACHE and OPTM have no replication cost since they do not need to exchange information between nodes for file replication. However, SAF may generate redundant replicas, and CACHE fails to utilize all storage.

3) Replica Distribution: Figure 2 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$\succ$CACHE$\approx$PDRS$\succ$SAF, where $\succ$ means closer similarity to OPTM. Combining the results on average delay, we find an interesting phenomenon: except CACHE, 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 (16). CACHE has large average delay because it does not utilize all storage space for replica creation, though it exhibits similarity with PDRS. We also observe that the CDFs of the proportion of resource allocated to replicas of DCG, CACHE, PDRS and SAF increases to over 0.9 quickly. This is because they allocate most resource to popular files, resulting in more replicas for these files. Though these protocols can reduce the delay of queries for popular files but cannot reduce the delay of queries for unpopular files. PCS is superior over these protocols because it averagely can reduce the delay of queries for both popular and unpopular files.

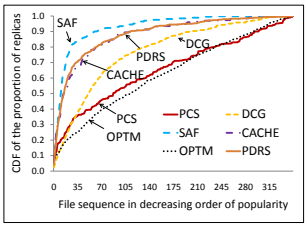


Fig. 2: CDF of the resource allocated to replicas in trace-driven GENI experiment.

B. Performance in the Trace-Driven Simulation

1) Hit Rate and Average Delay: Table IV shows the results of each protocol in the trace-driven experiments on NS-2. We see the hit rates of the six protocols follow the same relationship as in Table III due to the same reasons. We find that the average delays of the six 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 Static Wait 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 to shorter searching delay since a node can reach more neighbors. Also, the six protocols present lower hit rates 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 correctness of our analysis and the effectiveness of the proposed PCS.

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

Protocol	Hit rate	Average / 1% / 99% delay (s)	Replication cost
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	0

2) Replication Cost: From Table IV, we find that the replication costs of different protocols follow PDRS>DCG >PCS>SAF=OPTM=CACHE=0. This matches the results in Table III and the reasons are the same.

3) Replica Distribution: Figure 3 shows the CDF of the proportion of resource allocated to replicas of each file in the six protocols. From the figure, we find similar trend as that in Figure 2. That is, except CACHE, a protocol with closer similarity to OPTM has less average delay. This further proves the correctness of our theoretical analysis through trace-driven simulation.

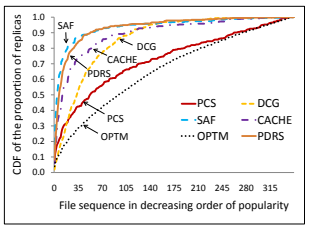


Fig. 3: CDF of the resource allocated to replicas in trace-driven simulation.

C. Performance With Different Network Sizes

In this test, we examined the performance of PCS and other protocols when the total number of nodes varied from 20 to 110 with a 10 increase in each step.

1) Hit Rate: Figure 4(a) plots the hit rates of the six replication protocols. We see the same relationship between different protocols as found in Table III and Table IV with the same reasons. Similarly, the reasons are also supported by the fact that the average delays of the six protocols present reverse order of the hit rate, as shown in Figure 4(b).

2) Average Delay: Figure 4(b) shows the average query delays of the six protocols. We observe the same results as that found in Table III and Table IV. Specifically, at all network sizes, PCS has 10%-15% less average delay than DCG, PDRS, SAF and CACHE, and it shows around 15% - 20% higher average delay than OPTM. CACHE has the largest average delay. Such results are consistent with aforementioned conclusions because of the same reasons. The results in Figure 4(a) and Figure 4(b) confirm the validity of our analysis and the effectiveness of PCS in different network sizes.

More nodes in the network enable a node to have more neighbors and hence more options to forward queries to the file holder. It is interesting to see that in the figures, the hit rate and average delay of each protocol generally remain stable as the number of nodes increases. This is because nodes move randomly and independently in the network. So, the probabilities that a query forwarder meets a file holder are approximately the same in networks with different sizes.

Figure 4(c) plots the 1st and 99th percentiles of the delays of the six protocols. We find that the 1st percentiles of delays of all protocols are all 0.01s. The relationship between the 99th percentiles of the delays of the six protocols is in line with that of the average delays in Figure 4(b), and that of the 99th percentiles in Table III and Table IV because of the same reasons explained previously. The result confirms that PCS is effective in reducing the average querying delay in networks with different sizes.

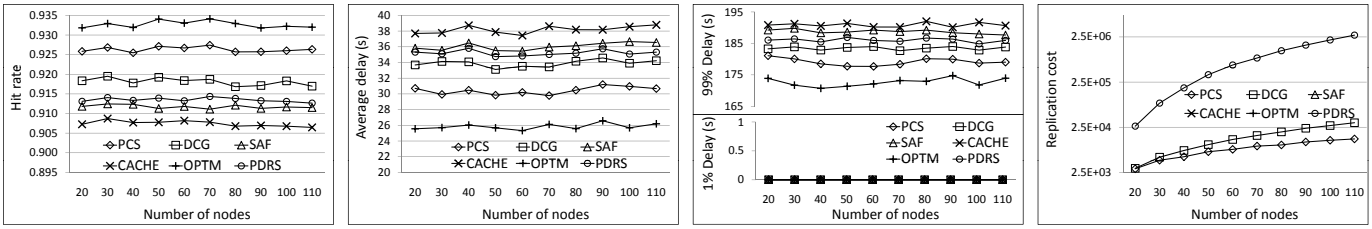
3) Replication Cost: Figure 4(d) illustrates the replication cost of each protocol. The replication costs of SAF, OPTM and CACHE are not shown since they equal 0. We find that PDRS generates high replication cost, DCG shows moderate replication cost, and PCS produces low replication cost. The result is consistent with those in Table III and Table IV because of the same reasons. We also observe that the replication costs of PDRS, DCG and PCS grow as the number of nodes in the system increases. This is because as the number of nodes increases, PDRS generates more messages during the broadcasting process for newly generated files, DCG produces more exchange messages between group members, and PCS replicates more files to neighbors.

4) Replica Distribution: Figures 5(a) and 5(b) show the CDF of the proportion of resource allocated to replicas in each protocol when the number of nodes is 20 and 110, respectively. From the figures, we find that all protocols exhibit similar relationship as Figures 2 and 3. That is, except CACHE, PCS shows the closest close similarity to OPTM and others follow: DCG≻PDRS≻SAF. The results again confirm the correctness of our theoretical analysis with results from different network sizes. We find that Figures 5(a) and 5(b) show similar results, which demonstrates the effectiveness of PCS in different network sizes. Combining above results, we conclude that OFRR can help shorten the average querying delay and PCS can realize it effectively in networks with different sizes.

D. Performance With Different Node Mobilities

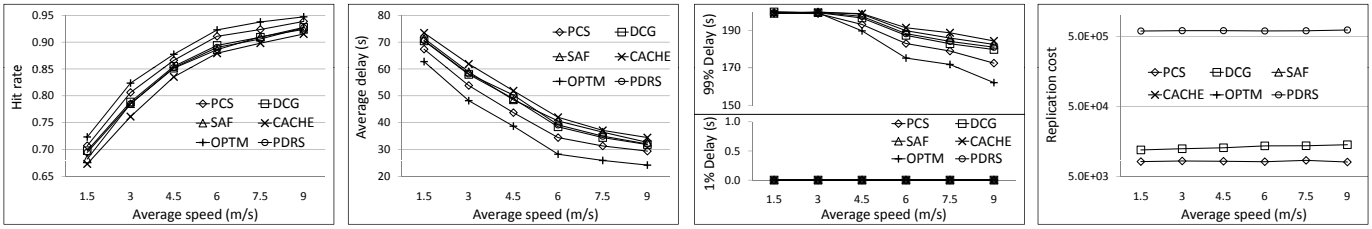
In this test, we examined the performance of the six protocols when the average movement speed of nodes varied from 1.5 m/s to 9 m/s with 1.5 m/s increase in each step.

1) Hit Rate: Figure 6(a) illustrates the hit rates of the six replication protocols. We observe the same relationship between different protocols as those found in Table III, Table IV and Figure 4(a) with the same reasons. We also find that, for



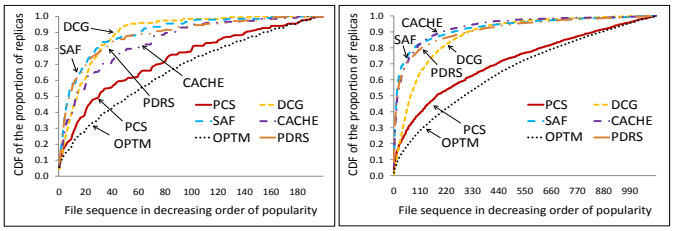
(a) Hit rate. (b) Average delay. (c) The 1% & 99% delays. (d) Replication cost.

Fig. 4: Performance of the file replication protocols with different network sizes.



(a) Hit rate. (b) Average delay. (c) The 1% & 99% delays. (d) Replication cost.

Fig. 6: Performance of the file replication protocols with different node mobilities.



(a) Network size = 20 nodes. (b) Network size = 110 nodes.

Fig. 5: CDF of the resource allocated to replicas with different network sizes.

all protocols, the hit rate is low at slow movement speed and is satisfactory (i.e., >90%) when the average movement speed is higher than 7.5 m/s. This is because a node usually needs long time to encounter requested files when it moves slowly, leading to more dropped requests due to TTL expiration.

2) Average Delay: Figure 6(b) shows the average querying delays of the six protocols. We observe that PCS shows the closest result to OPTM, and it reduces the average delay of SAF, DCG, PDRS and CACHE by about 10%-15%. Again, the result is consistent with those in Table III, Table IV and Figure 4(b) due to the same reasons. The result also shows that the change of node movement speed does not affect the relative performance among different protocols. This is because, as shown in Equation (21), the effectiveness of replication protocol with the same network size and node mobility distribution is only determined by the resource allocation for file replicas. These results confirm the correctness of the OFRR and the effectiveness of the PCS with different node mobilities.

We also observe that the average delays of all protocols decrease as node movement speed increases. When nodes move faster, the average time needed for two nodes to meet with each other is shortened, leading to less average delay. The result implies that the movement speed of a node affects the number of nodes it can encounter in a unit period and hence the availability of its files, which justifies the necessity of considering node meeting ability as resource in file replication.

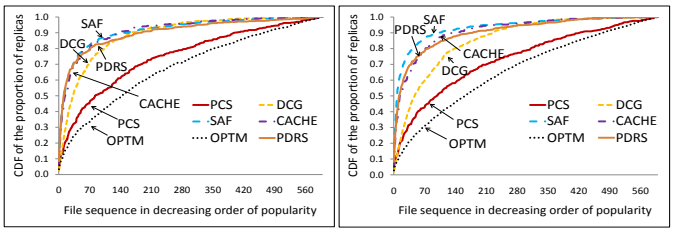
Figure 6(c) depicts the 1st and 99th percentiles of the delays of the six replication protocols. Similar to the results in previous experiments, the 1st percentiles of delays of all protocols are nearly 0 and the 99th percentiles of the delays of these protocols present the same relationship as in Table III and Table IV for the same reasons. When the average speed is slow (i.e., 1.5 m/s and 3 m/s), the 99th percentiles of the delays of all protocols equal the TTL (200s) since we use the TTL as the delay of dropped requests. When nodes move slowly, they averagely need longer time to encounter requested files.

3) Replication Cost: From Figure 6(d), we find that PRDS presents the highest replication cost, DCG has moderate replication cost, and PCS generates low replication cost. We did not plot the replication costs of other protocols in the figure since the results are almost 0. The relationship of the six protocols on replication cost remains the same as those in Table III, Table IV and Figure 4(d) because of the same reasons. We also find that the replication costs of PRDS, PCS and DCG remain stable as the node movement speed increases. The replication costs of DCG and PRDS are only decided by the number of nodes in the system, since the former exchanges information between group members for replication and the latter broadcasts messages through the network for newly created files. For PCS, the number of replica distribution attempts is unrelated to node mobility. So their replication costs remain stable when the node movement speed increases.

4) Replica Distribution: Figures 7(a) and 7(b) show the CDF of the proportion of resource allocated to replicas of each file in each protocol when the average movement speed of nodes is 1.5 m/s and 9 m/s, respectively. We find that all protocols generate similar results as those in Figures 5(a) and 5(b) because of the same reasons. The results again verify the correctness of our theoretical analysis and the effectiveness of PCS in following OFRR in various node mobilities.

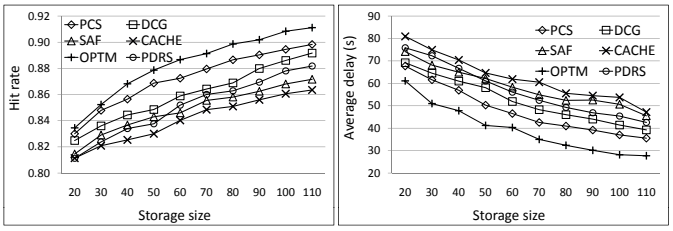
E. Performance With Different Storage Sizes

We also tested the performance of different replication protocols when the storage space for replicas in each node



(a) Average speed = 1.5 m/s. (b) Average speed = 9 m/s.

Fig. 7: CDF of the resource allocated to replicas with different node mobilities.



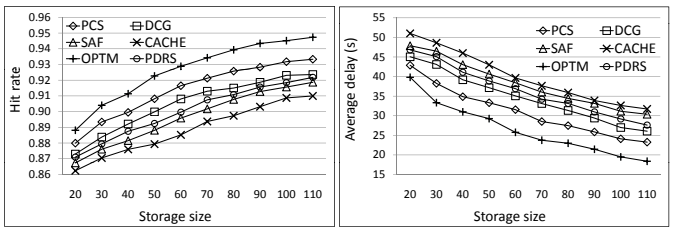
(a) Hit rate. (b) Average delay.

Fig. 8: Performance with different storage sizes with real trace mobility.

ranges from 20 to 110 with 10 increase in each step on the NS-2 simulator. Figure 8 and Figure 9 show the hit rate and average delay of the six protocols in the real trace and synthesized node mobility, respectively. We see that they show the same relationship on their performances with the real trace mobility and with the synthesized mobility. Specifically, as the storage size increases, the hit rates of all protocols increase and their average delays decrease. This is because the number of replicas of each file increases when there is more storage space in the system, leading to higher hit rate and lower average delay. We also find that the performance relationship between the six protocols on the two metrics matches those in Table III and Table IV due to the same reasons. Such results confirm the correctness of OFRR and the effectiveness of PCS with different degrees of storage resource constraint.

VI. Conclusion

In this paper, we investigated the problem of how to allocate limited resource in file replication for global file searching efficiency optimization in MANETs. We first theoretically analyzed the influence of replica distribution on the average querying delay under constrained available resource, and derived an optimal replication rule to allocate the limited resource to file replicas in order to minimize the average querying delay. Unlike previous 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 protocol. Finally, we designed the Priority Competition and Split replication protocol (PCS) that realizes the proposed optimal replication rule in a fully distributed manner. Experiments on both real-world GENI testbed and NS-2 with real trace and synthesized mobility confirm both the correctness of our theoretical analysis and the effectiveness of PCS. In our future work, we will study the effect of PCS in a MANET with other node movement models.



(a) Hit rate. (b) Average delay.

Fig. 9: Performance with different storage sizes with synthesized mobility.

Acknowledgements

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

References

[1] "Qik," http://qik.com/.
[2] "Flixwagon," http://www.flixwagon.com/.
[3] 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.
[4] B. Chiara, C. Marco, J. Jacopo, and P. Andrea, "Hibop: A history based routing protocol for opportunistic networks," in *Proc. of WoWMoM*, 2007.
[5] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic routing in intermittently connected networks," *MC2R*, vol. 7, no. 3, pp. 19–20, 2003.
[6] F. Li and J. Wu, "Mops: Providing content-based service in disruption-tolerant networks," in *Proc. of ICDCS*, 2009, pp. 526–533.
[7] S. Moussaoui, M. Guerroumi, and N. Badache, "Data replication in mobile ad hoc networks," in *Proc. of MSN*, 2006, pp. 685–697.
[8] L. Yin and G. Cao, "Supporting cooperative caching in ad hoc networks," *TMC*, vol. 5, no. 1, pp. 77–89, 2006.
[9] 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.
[10] 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.
[11] H. H. Duong and I. Demeure, "Proactive data replication using semantic information within mobility groups in MANET," in *Proc. of Mobilware*, 2009, pp. 129–143.
[12] V. Gianuzzi, "Data replication effectiveness in mobile ad-hoc networks," in *Proc. of PE-WASUN*, 2004, pp. 17–22.
[13] S. Chessa and P. Maestrini, "Dependable and secure data storage and retrieval in mobile wireless networks," in *Proc. of DSN*, 2003.
[14] X. Chen, "Data replication approaches for ad hoc wireless networks satisfying time constraints," *IJPEDS*, vol. 22, no. 3, pp. 149–161, 2007.
[15] S. U. Khan, A. A. Maciejewski, H. J. Siegel, and I. Ahmad, "A game theoretical data replication technique for mobile ad hoc networks," in *Proc. of IPDPS*, 2008, pp. 1–12.
[16] 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.
[17] L. Kleinrock, *Queueing Systems, Volume II: Coputer Applications*. John Wiley & Sons, 1976.
[18] J. Kangasharju, K. W. Ross, and D. A. Turner, "Optimizing file availability in peer-to-peer content distribution," in *Proc. of INFOCOM*, 2007.
[19] M. Musolesi and C. Mascolo, "Designing mobility models based on social network theory," *MCCR*, vol. 11, pp. 59–70, 2007.
[20] "GENI project," http://www.geni.net/.
[21] "Orbit," http://www.orbit-lab.org/.
[22] "The Network Simulator ns-2," http://www.isi.edu/nsnam/ns/.
[23] 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.
[24] T. Spyropoulos, K. Psounis, and C. Raghavendra, "Efficient routing in intermittently connected mobile networks: The single-copy case," *ACM/IEEE Transactions on Networking*, 2007.
[25] M. Lu and J. Wu, "Opportunistic routing algebra and its application," in *Proc. of INFOCOM*, 2009.
[26] T. Hara, "Effective replica allocation in ad hoc networks for improving data accessibility," in *Proc. of INFOCOM*, 2001, pp. 1568–1576.